A Survey on Network Simulators, Emulators, and Testbeds used for Research and Education

Jose Gomez[a], Elie F. Kfoury[a], Jorge Crichigno[a], Gautam Srivastava[b,c]

[a] *College of Engineering and Computing, University of South Carolina, Columbia, U.S.A*
[b] *Department of Mathematics and Computer Science, Brandon University, Canada*
[c] *Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon*

Abstract

Network operators and researchers constantly search for platforms to evaluate future deployments and test new research ideas. When experimenting, they usually face challenges in deciding on an appropriate platform to validate the advantages and limitations of their proposed system. These challenges include finding an experimentation environment that balances traffic realism, scalability, and cost. An experimenter can evaluate systems, protocols, and security implementations using simulators, emulators, or testbeds to validate the expected behavior of the proposed idea. Simulators and emulators provide a controlled environment to conduct reproducible experiments but lack realism. Testbeds provide realism and scale depending on the available resources. However, real equipment can be costly and unavailable for many experimenters. The inability to test networking ideas in a realistic environment at a large scale presents a barrier for companies, institutions, and network vendors to implement new features, thus, slowing down innovation. In the past few decades, the networking community developed new platforms to test new ideas and deployments at scale, with realism, and at lower costs. These platforms also enable the instruction of networking concepts, cybersecurity, distributed computing, storage systems, and science applications. From the learner's side, practical hands-on experience is required to internalize concepts and improve troubleshooting skills. Learning these concepts can be challenging due to the multidisciplinary nature of networking instruction, where a learner must have a background in several computing areas (e.g., operating systems, programming languages, and computer architecture). This paper presents experimentation platforms used to conduct research in computer networks and evaluates the potential of these platforms for instructing networking courses. This paper examines the literature and presents a taxonomy of network experimentation platforms. It also discusses challenges, analyzes the limitations, and suggests future perspectives by providing an overview of the tools, a description of the underlying resources (i.e., hardware and software), and a summary of the supported experiments. The paper aims to assist experimenters and educators in deciding which platform is more suitable for their experimentation needs and discuss the challenges and future directions related to the network experimentation platforms.

Keywords: Network Emulator, Network Simulator, Network Testbed, FABRIC, GENI, SDN, Programmable Data Planes, P4.

1. Introduction

Testing network ideas requires a systematic methodology to reproduce experiments realistically. Evaluating new network ideas involves a high degree of experimentation before a protocol is adopted or a system is deployed in a production environment. The experimentation process typically consists of testing a system at a sufficient scale to convince operators that the idea reliably addresses their needs or persuade vendors that the protocol can add a competitive advantage to their products [1]. Therefore, using the appropriate platform to validate research proposals is essential to run large-scale experiments at a reduced cost. However, conducting research in computer networks bares some limitations due to the heterogeneous and distributed nature of the network components. Depending on the complexity of the experiment, researchers can find it challenging to access the resources to run high-performance experiments, test distributed algorithms, and program the behavior of the network components. The research community has been developing network experimentation platforms to facilitate experimenters in fulfilling their research demands to address these requirements.

Another critical aspect of network experimentation consists of training experimenters in general and specific areas to operate existing deployments and conduct research efficiently. Learning computer networks requires a balance between theoretical knowledge and hands-on experience. Practical insights into networking concepts are essential for conducting research and troubleshooting. Instructing computer network courses can be challenging due to abstractions that learners must understand and the available tools to apply these theoretical concepts. Unlike programming languages, operating systems, data structures, and other computer science topics, computer network instruction is traditionally delivered with limited practical instruction. Therefore, the learner can experience difficulties applying them in the real world [2]. Topics such as network protocols, cybersecurity, virtualization, blockchain, and Software-Defined

Email addresses: gomezgaj@email.sc.edu (Jose Gomez), ekfoury@email.sc.edu (Elie F. Kfoury), jcrichigno@cec.sc.edu (Jorge Crichigno), srivastavag@brandonu.ca (Gautam Srivastava)

Networks (SDN) require practical exercises to understand how they work and how to build technical solutions with them [3]. Understanding these concepts is important to conduct innovative research and reduce network downtimes and misconfigurations due to human-induced errors [4]. For instance, Internet reliability is a feature that is taken for granted. However, outages, failures, and security breaches are events that can impact the revenue of companies [5–7]. Therefore, network operators must be qualified to understand, troubleshoot, and prevent network disruptions to ensure high availability and reliability. Troubleshooting computer networks requires understanding the configuration of network components and the behavior of the protocols used by those components. These requirements demand that the operator have a deep understanding of networking concepts and the practical expertise to apply those concepts proactively. Hence, during the education process, hands-on exercises can facilitate learners to have better preparation to mitigate network flaws. On the other hand, practical experiences allow instructors and institutions to evaluate the learners' practical skills. It is a challenge for educators to choose the most appropriate methods and tools to produce qualified professionals with problem-solving skills.

The most commonly used platforms to run experiments in network research are simulators, emulators, and virtual testbeds. Simulators are based on numerical models to represent the behavior of network components. They usually reproduce the evolution of the network behavior as a function of discrete events, which can be time or custom events. These events are consistently generated from the user space regardless of the specification of the host. Such characteristics facilitate reproducing results. However, simulations lack realism due to the oversimplification of real-world events. Therefore, a simulation result might differ from a similar system configured in hardware. Network emulators leverage the software primitives provided by the Operating System (OS) to run switches, links, servers, and packets as if they are in an actual network [8]. Events in network emulators occur continuously and use the real protocol stack available in the OS. Emulated networks run with real code rather than discrete models. Therefore, emulators are practical tools to test code that can be easily ported to real deployments. However, network emulators deviate from the expected behavior as the network topology includes many components. This limit varies depending on the hardware resources available in the host device. Network simulators and emulators are agile tools to investigate issues and measure performance before fully deploying novel applications or protocols. On the other hand, virtual testbeds provide the experimenters with realism and high performance, although deploying testing scenarios is slower. Virtual testbeds leverage real hardware to ensure that results are produced with realism. Virtual testbeds focus on research areas such as cybersecurity, routing, machine learning, programmable data planes, and others. Moreover, some testbeds grant access to academic institutions to teach computing concepts [9–12]. These testbeds deliver access to specialized hardware that experimenters can dedicate to their experiments. These hardware resources are isolated to avoid interaction with other experiments. However, it can be hard to reproduce or even produce experiments due to the topology restrictions or resource availability in each testbed. Currently, there are efforts to expand the features of the virtual testbeds to cover topics such as network protocols, the Internet of Things (IoT), SDN, cybersecurity, and other related issues. Moreover, there are scientific instruments that can be accessed via virtual testbeds [13]. Usually, these resources are specialized hardware (i.e., microscopes, telescopes, high-performance computers, and high-speed storage) that produce a large amount of data that must be transferred to a remote location.

1.1. Contributions

To the best of the authors' knowledge, the literature has been missing a survey on network platforms used for research and education (R&E). This paper aims to provide the reader with an overview of the capabilities of these tools to conduct research in different computing areas, such as routing, distributed network protocols, P4-programmable data planes, SDN, distributed computing, machine learning, big data, IoT, and others. This paper also presents the potential use of the surveyed works for education purposes by summarizing the learning outcomes obtained with these tools in academic environments. Finally, it discusses the challenges and the limitations of these platforms proposing future directions that can serve as an improvement when testing new ideas. The main contributions of this survey can be framed as follows:

- Surveying network simulators, emulators, and virtual testbeds used for research and education.
- Providing a taxonomy that categorizes the platform according to key features that can facilitate the decision process of an experimenter.
- Summarizing the type of research conducted and supported by the experimentation platforms.
- Discussing the challenges and limitations of the network simulators, emulators, and testbeds.
- Proposing future directions to improve limitations found in the network simulators, emulators, and testbeds.

1.2. Survey Organization

Figure 1 illustrates the paper roadmap. Section 2 describes the related surveys. Section 3 provides background on network experimentation tools. Section 4 describes the employed methodology and the proposed taxonomy, section 5 surveys the simulation tools, section 6 describes the emulation tools, and section 7 covers the network testbeds. Section 8 discusses the challenges and future works, and section 9 concludes the paper.

2. Related Surveys

Huang *et al.* [14] presented a comprehensive survey on SDN testbeds. The survey explains the design objectives, key technologies, network deployment, and experiments that employ large-scale SDN testbeds, including

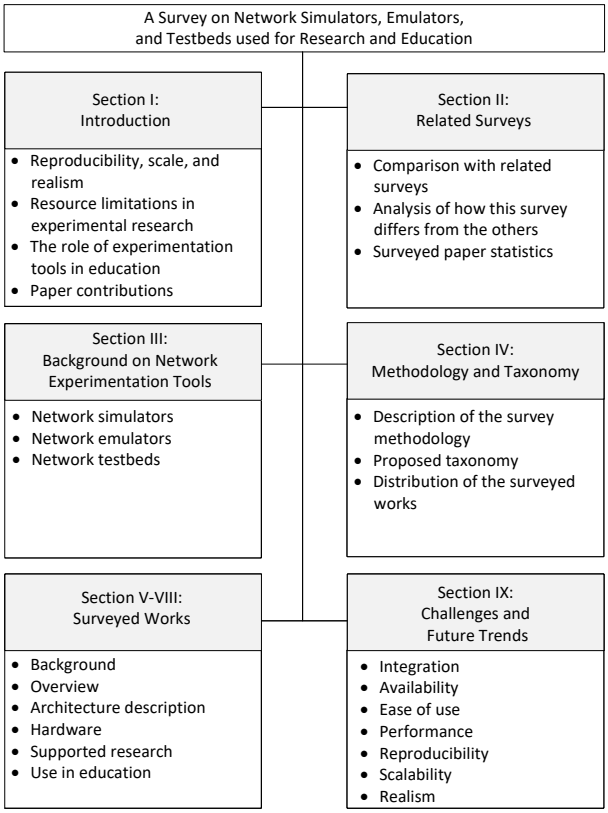

Figure 1: Paper roadmap.

the research challenges. The authors provide a comparison between traditional and SDN testbeds, highlighting the flexibility of SDN technology. Additionally, the survey describes current SDN testbeds available for research and education. The paper concludes with the challenges and future works, including federation, network slicing, tools and deployment, multi-domain, and compatibility limitations.

Tsai *et al.* [15] surveyed the issues associated with developing network emulation testbeds focusing on the control framework. The survey focuses on state-of-the-art architectures, the interoperability of orchestrating resources, and how components are managed. The authors define a general network testbed architecture consisting of hardware infrastructure, control framework, and management application. The main focus is on the control framework, the middleware that manages the hardware infrastructure and updates the management application. The survey highlights the advantages of employing testbeds to conduct network experiments, such as having a secure environment, availability of high-performance computing resources, cost reduction, flexibility in testing new protocols, and network programming. The authors conclude the survey by listing and discussing open issues, including resource orchestration, collaboration, and testbed federation.

Chouliaras *et al.* [16] surveyed cyber range testbeds used for research and education. A cyber range is a specialized testbed used by the military and law enforcement to train cybersecurity personnel by reproducing a threat environment (e.g., to respond to a distributed denial-of-service DDoS, ransomware, and others). The authors conduct a systematic review of cyber range systems and their approaches (e.g., simulation, emulation, and hybrid). The paper presents the current state of the art on testbeds and cyber ranges, the result of interviews conducted with organizations that provide cyber range testbeds, and the challenges and future directions.

Prvan and Ožegović [2] presented a survey of methods and paradigms for teaching computer network courses. In the article, the authors classify the teaching methods into four categories: methods based on visualization (i.e., network simulators), methods based on multimedia applications, methods based on virtualization, methods based on active learning, and methods based on hands-on experiences. Additionally, the authors provide a cross-comparison of each method based on its advantages, disadvantages, and challenges from the perspective of students and educators.

Nussbaum [17] surveyed computer science testbeds, focusing on their capabilities to support reproducible research. The testbeds included in this paper are Chameleon, CloudLab, and Grid‘5000. The author explores the impact of design choices, services, and features to reproduce experiments in big data, cloud computing, and High-Performance Computing (HPC). The paper also identifies the weaknesses of the surveyed testbeds and highlights possible areas for improvement.

Bakni *et al.* [18] developed an evaluation framework to compare the capabilities of network simulators based on standardized criteria. They implemented their methodology by evaluating two simulators used primarily for education purposes. Their findings show the performance, advantages, and disadvantages based on the qualitative (i.e., nature of the tool, supported protocols, user-friendliness) and quantitative (i.e., CPU utilization, memory usage, execution times of discrete events) criteria to show their suitability for researchers in network domains.

Khan *et al.* [19] conducted a comparative analysis of open-source network simulators used in wireless networks. The authors focused on quantitative variables such as CPU utilization, memory usage, computational time, and scalability by simulating Mobile Ad Hoc Networks (MANETs) routing protocols. The final results suggest the most suitable simulators based on the type of research an experimenter aims to conduct.

Nussbaum and Richard [20] performed a comparative study of network link emulators by providing an overview of the main characteristics of Dummynet, NISTNet, and Linux traffic control. The authors focus on the accuracy of the emulators considering the bandwidth and latency and how these metrics are affected by the hosting system (e.g., CPU and memory). The authors discuss the weaknesses of such tools and discuss possible solutions.

Lochin *et al.* [21] surveyed various network emulation platforms to introduce new researchers to the appropriate emulator to conduct experiments. The authors explain and compare the simulation with emulation platforms to highlight the shortcomings of simulation platforms. Therefore, experimenters can observe how the network emulator can reproduce realistic scenarios by using the primitives provided by the operating system. The survey focuses on centralized approaches that enable low-cost and manageable implementation in the context of research labs or industrial developments.

Table 1 highlights the differences between the related

Table 1: Comparison with Related Surveys.

Ref.	Experimentation Platforms			Scope of the Surveys				Discussions		Focus of the survey
	Simulators	Emulators	Testbeds	Research	Education	Prototyping	Interoperability	Challenges	Future directions	
[14]	○	○	●	●	⊙	●	○	⊙	⊙	SDN testbeds
[15]	○	⊙	●	⊙	⊙	⊙	○	●	⊙	Control frameworks
[16]	○	⊙	●	⊙	●	⊙	⊙	⊙	⊙	Security testbeds
[2]	⊙	⊙	⊙	⊙	⊙	⊙	○	●	●	Academic platforms
[17]	○	○	●	●	⊙	⊙	⊙	⊙	⊙	Large-scale platforms
[18]	●	⊙	○	⊙	⊙	⊙	⊙	●	●	Performance analysis
[19]	●	○	○	●	⊙	○	○	○	○	Wireless platforms
[20]	○	●	○	⊙	⊙	⊙	⊙	●	●	Kernel-based platforms
[21]	○	●	⊙	●	●	⊙	⊙	⊙	○	Platform selection
[22]	○	⊙	●	⊙	●	●	●	●	⊙	SDN testbeds
This survey	●	●	●	●	●	●	●	●	●	Network platforms

● Covered in this survey ○ Not covered in this survey ⊙ Partially covered in this survey

surveys and this survey. Previous surveys mainly focus on one experimentation platform and discuss various issues, including performance, educational use, platform selection, and wireless platforms. This survey collects the literature related to network simulators, network emulators, and network testbeds to provide the reader with a broad perspective on the potential of these platforms for conducting research and instructing academic courses. To the best of the authors' knowledge, this work is the first to encompass diverse experimentation platforms and discuss the challenges and future works that can contribute to improving such platforms. Specifically, this survey describes network simulators, network emulators, and network testbeds following a taxonomy. The taxonomy classifies the work to compare and discuss the strengths and weaknesses of the surveyed works. Each category contains the foundational paper of each platform, the published experiments conducted with each platform, and the experiences in using such platforms in an academic environment.

Martini *et al.* [22] discusses the experimentation of SDN and Cloud Orchestration in virtualized testing facilities. The authors present performance results and comparisons between these technologies. The study involves conducting experiments within virtualized testing environments that integrate SDN and cloud orchestration techniques. The main focus is to assess the performance of these technologies and understand how they interact in a virtualized setting. The authors provide detailed insights into the setup and methodology used for the experiments. They measure various performance metrics such as network throughput, latency, and resource utilization to evaluate the effectiveness and efficiency of SDN and cloud orchestration. The results and comparisons presented in the paper highlight the strengths and weaknesses of both SDN and cloud orchestration, especially when deployed together in virtualized testing facilities.

3. Background

This section provides a background on simulators, emulators, and testbeds used to conduct network experiments. It defines, describes, and compares each platform's purpose, features, and limitations.

3.1. Network Simulators

Network simulators use a set of discrete models to reproduce the behavior of real hardware [23–29]. Experiments conducted in network simulators run in virtual time due to the simulated event, which can occur simultaneously and be paused or executed at a faster pace. The underlying simulator runs in user space and produces the same result each time, independently of the machine's specifications on which the experiment runs. Consequently, simulation results are considered easy to reproduce. However, results obtained from simulations might not be realistic due to the simplified behavior determined by the model. This simplification can lead to discrepancies compared to the same experiment on real hardware.

Network simulators are classified into time-based simulators and discrete-event simulators. Time-based simulators produce an output as an incremental progression of time slots. Events are executed during each time slot as the simulation advances. Figure 2(a) shows the flowchart of a time-based simulator. Examples of time-based simulations include IoT applications, transport protocols (e.g., TCP, UDP), routing protocols (e.g., BGP, OSPF), and congestion control algorithms (e.g., CUBIC, Reno, BBR). On the other hand, discrete-event simulators are governed by events different than a constant clock, which can be the activation of a node, the variance of energy levels, the expiration of a route in the routing table, and others. The simulation time progresses after each scheduled event is executed. Figure 2(b) shows the flowchart of a time-based simulator. Experiments that can be modeled using a discrete-event simulator typically include optical

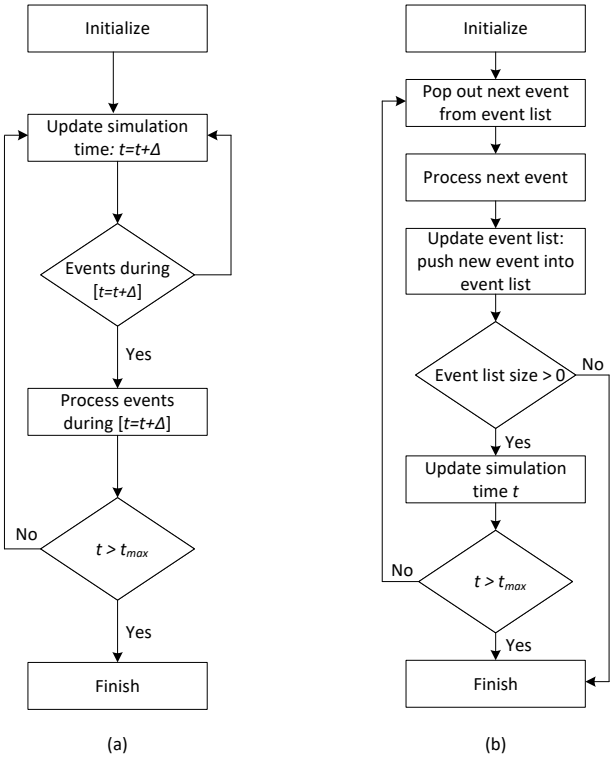

Figure 2: Type of network simulators. (a) Time-based simulator. (b) Event-based simulator [35].

networks [30, 31] and IoT protocols such as ZigBee [32], Z-Wave [33], Near Field Communication (NFC) [34], among others.

3.2. Network Emulators

Network emulators process events in continuous time and run code rather than a discrete event model. Typically, emulators isolate system resources such as the kernel data structures, the file system, and the network protocol stack to implement emulated hosts, switches, and other network appliances. There are two main categories of network emulators. The first category is known as Full-System Emulation [36] consisting of independent VMs coupled by virtual switches [37]. In this emulation environment, each VM represents a host, a switch, or a network appliance such as a firewall. The second category is known as Container-Based Emulator (CBE). It isolates resources by performing a lighter way of virtualization called process-level virtualization. This isolation mechanism allows sharing of resources from the host VM. A CBE achieves better scalability than the Full-System Emulation by running only a group of user space processes. The cost of adding a new host is the same as adding a new process. The main advantage of network emulators is that the code used for an experiment can be ported to a real server and obtain similar results [1].

3.3. Network Testbeds

Network testbeds provide a collection of real and virtualized hardware to conduct experiments in diverse research areas. The research community [10, 14, 38] provides openly available testbeds with high-performance hardware. These testbeds can be shared across a community of users or be specific to one project. The main advantage of real hardware testbeds is that they produce repeatable results and provide resource isolation. This feature prevents other users from affecting the performance of each other. Another relevant aspect is that testbeds produce realistic results. However, each testbed have its own characteristic which carries limitations such as topology restrictions, hardware compatibility, and resource configuration. This survey defines network testbeds as virtualized network facilities. These facilities are essentially laboratories constructed upon a coordinated federation of testbeds that incorporate various types of hardware. Consequently, the terms network testbed and virtual testbed are used interchangeably.

3.4. Comparison between Simulators, Emulators, and Testbeds

Network simulators are flexible tools to run custom and large-scale topologies at a low cost. They are based on discrete models that enable the simulation of various devices. However, network simulators lack realism when comparing the behavior of simulation models against real hardware.

Network emulators isolate kernel components into isolated instances, making it possible to scale experiments and run real code. This feature makes it possible to port the configuration from an emulated experiment to real hardware and obtain similar performance. However, emulators can deviate from a realistic behavior if the allocated hardware resources are insufficient to support all the running processes.

Network testbeds provide more realistic testing scenarios than emulators and simulators. They play an essential role in designing and validating network protocols and distributed systems [39]. However, network testbeds can lack scalability, and their results are hard to reproduce in other environments. Another limitation is resource availability, which constrains experimenters from scaling up topologies to test new ideas. Once the topology is defined, it is hard to perform changes. Thus limiting the experiment's flexibility. Besides the experimentation constraints, the configuration overhead is a barrier to porting an experiment to another testbed or reproducing it in the future.

4. Methodology and Taxonomy

Figure 3 depicts the proposed taxonomy. The taxonomy was designed to cover relevant works on the network platforms used for research and education. This paper categorizes the surveyed works based on various high-level disciplines. A reader interested in a specific topic can move to the corresponding section and find the necessary background to understand the section's content. Each high-level category is further divided into subcategories. For instance, works related to "Full Virtualization" fall under the "Network Emulators" category. Each section has a table that summarizes the relevant characteristics of each work discussed in that section. Following the literature review, each section introduces the tools, describes the underlying architecture and hardware, discusses the type of research that can be conducted using each platform, and analyzes the potential use for instructing network courses. At the end of each section,

the relevant outcomes are discussed, highlighting the surveyed schemes' strengths and weaknesses.

5. Network Simulators

With the increasing growth and complexity of networked systems, there is a need for simulation technologies to conduct accurate and scalable tests. Simulators are helpful tools for network research that enable reproducibility, rapid prototyping, and education. Simulation-based studies allow experimenters to visualize events in scenarios involving large-scale models, custom applications, and dynamic environments. Typically, network simulators leverage discrete-event models, where the simulation runs in discrete steps, and events are used to communicate between simulation entities to produce a state transition [71]. These states change based on events that trigger a transition and generate output results [72].

The development and testing of networking protocols strongly depend on simulation support. Simulations are an efficient way to run tests when scalability and cost are constraints for experimentation. Another essential advantage of simulations is that the environmental conditions can be varied and repeated many times. These characteristics enable running simulations that compare the performance of different protocols in various scenarios. In real-world tests, minor disturbances can change the outcome of experiments. Thus, simulations can be used to test new systems whose behavior is uncertain when running on real hardware.

Another key motivation for using network simulators is that they provide a controlled and repeatable environment for experimentation. By varying network conditions and parameters, researchers can examine how protocols perform under different scenarios and stress test their resilience to failure. This helps to identify weaknesses and potential improvements that can be made to enhance the overall performance of a system.

Network simulators are flexible tools that can be used to teach computer network courses and other computer science topics. Unlike programming courses that usually require a computer [2], teaching computer networks demands additional resources such as networking equipment which can both be costly and take up a lot of space [73, 74]. Therefore, network simulators are practical tools that can be used in education to introduce learners to foundational and advanced concepts.

5.1. Protocol Evaluation

This category includes network simulators used in general experimentation, such as protocol modeling, network performance, routing, and wireless communications. Typically, these simulators aim at covering various types of network experiments by allowing users to create libraries to describe custom scenarios. These resources add flexibility to the platforms at the expense of not simulating model-specific behaviors.

Network simulators are powerful tools that can be used to evaluate the performance and behavior of network protocols in a controlled environment. These simulators allow researchers, developers, and network engineers to test new protocols, compare different design choices, and identify potential issues before deployment in a real-world network.

One of the main motivations for using network simulators is that they can save time and resources compared to testing protocols on a physical network. With a simulator, researchers can easily recreate complex network topologies, traffic patterns, and various network conditions that are difficult or impossible to replicate in a physical network. This allows for more comprehensive and accurate evaluations of protocols without the need for expensive equipment, lengthy setup times, and potential disruptions to real network traffic.

Furthermore, network simulators can also provide insights into the scalability, interoperability, and security of network protocols. Researchers can simulate large-scale networks and evaluate the protocol's ability to handle a high volume of traffic and diverse devices. They can also test the protocol's compatibility with different operating systems, hardware, and software configurations. Finally, network simulators can be used to simulate various security attacks and evaluate the protocol's resilience to such attacks.

Network simulators provide a powerful tool for evaluating network protocols. They offer a controlled, repeatable, and cost-effective environment for researchers, developers, and network engineers to test new protocols, compare different design choices, and identify potential issues before deployment in a real-world network.

5.1.1. OMNeT++

The Objective Modular Network Testbed in C++ (OMNeT++) [40, 41] is an object-oriented discrete event simulation environment to test communication protocols, multicore applications, and other distributed systems. OMNeT++ implements a framework approach that supports the basic machinery and tools to write simulations rather than directly providing simulation components for computer networks, queueing theory, and other domains. With OMNeT++, experimenters can create their own models that target a specific type of experimentation (e.g., simulating the behavior of Active Queueing Management (AQM) algorithms, congestion control algorithms, and other protocols that involve traffic dynamics). OMNeT++ supports large-scale simulations and visualization tools that facilitate interpreting results and debugging.

OMNeT++ leverages modules that communicate using message passing to create simple and compound models. Simple modules are active elements written in C++ using a simulation class library. A group of simple modules creates a compound module. Figure 4 represents a network of simple and compound modules. Messages can be exchanged between simple and compound modules. Experimenters can define complex modules consisting of a collection of simple and compound modules known as module types. When a module type is invoked in a simulation, there is no distinction if it is a simple or compound module, allowing experimenters to split a module into several simple modules in a compound module.

Modules are components that allow the incremental deployment of new functionalities without interfering with the existing ones. Modules can communicate using messages which contain timestamps and arbitrary

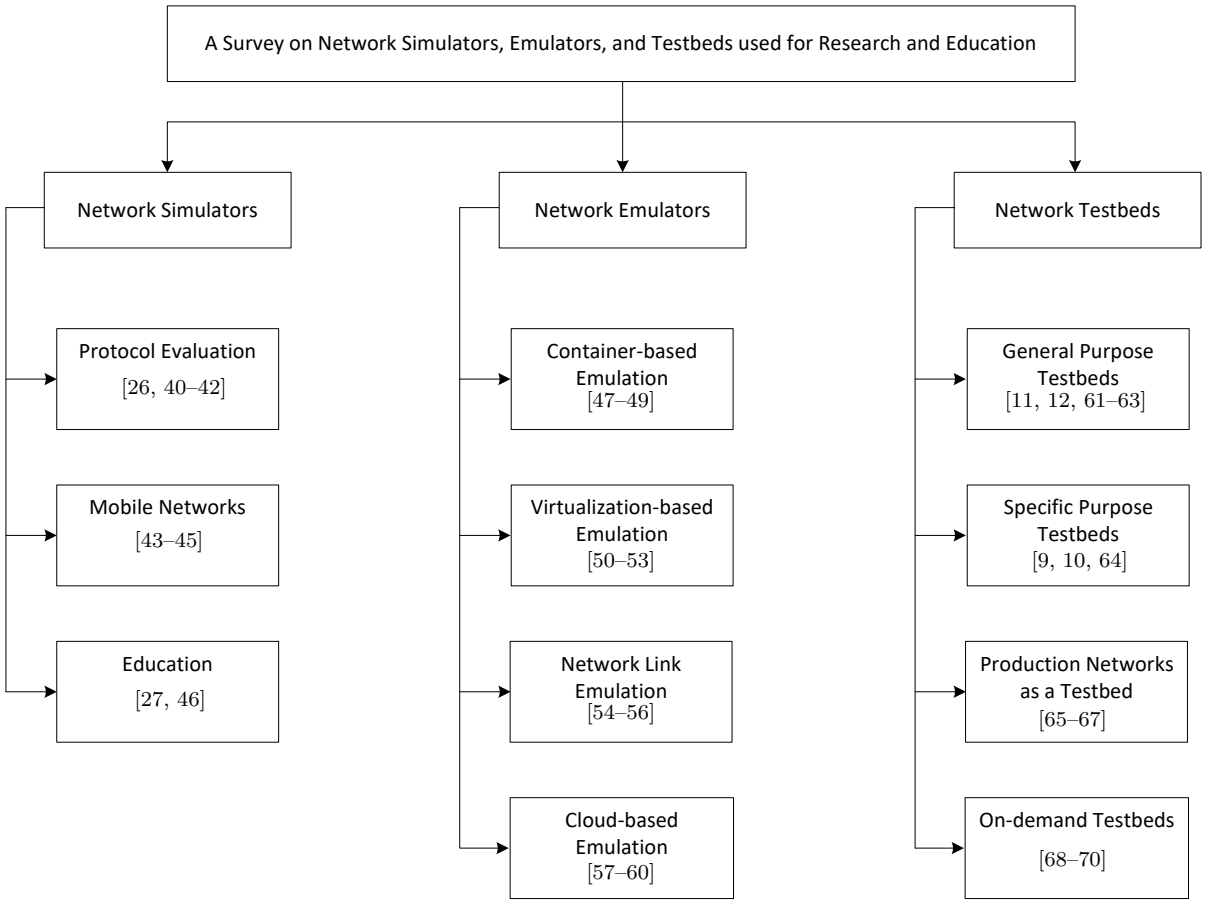

Figure 3: Taxonomy of network simulators, emulators, and testbed used for research and education.

data. Typically, messages are exchanged via gates, which are the input and output interfaces. Linked gates create connections that can only be used within the same hierarchy, meaning that the modules comprising a compound module cannot communicate with a simple external model.

5.1.2. ns-3

The Network Simulator 3 (ns-3) [26] is an open-source network simulation environment that aims at improving realism compared to its predecessor, the Network Simulator 2 (ns-2) [72]. The ns-3 simulator was developed from scratch addressing the shortcomings of ns-2, which included scalability, outdated code design, and difficulties in using the simulator [75]. Ns-3 includes useful resources such as scalability tools, cross-layer features, and real-world integration features. This last feature allows researchers to read and parse packet capture (pcap) files provided by tools such as Wireshark [76] and tcpdump [77]. The ns-3 simulator associates each event with its execution time governed by temporal increments. This process involves associating every event to a point in simulation time where events are initiated and triggered consecutively, simulating discrete increments from one event to another.

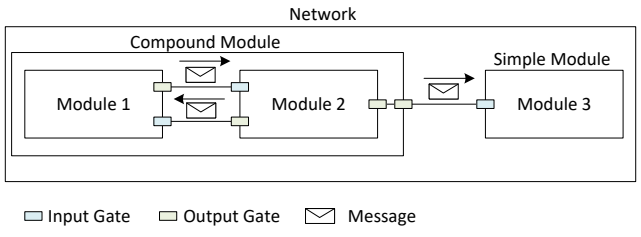

Figure 4: Model architecture in OMNeT++ [40].

Experimental research with ns-3 shows that the simulator can reproduce realistic network models in wired and wireless systems. Mezzavilla *et al.* [78] created a lightweight model in ns-3 to provide an accurate link performance metric at a low computational cost. The model relies on the knowledge of the Signal to Interference + Noise Ratio (SINR) and of the modulation and coding scheme in LTE communications. The authors integrated the model into the official distribution of the ns-3 simulator. Other authors [79–83] also evaluated and integrated wireless communication models in ns-3. Alberro *et al.* [84] used ns-3 to reproduce a data center environment for testing routing protocols. The authors integrated the Free Range Routing (FRR) protocol suite, which supports multiple routing protocols. The authors conducted comprehensive control plane experiments over fat-tree topologies, obtaining scalability when running the simulation on a single host environment. The proposed solution demonstrated that the customized ns-3 simulator could be effectively used for experimenting in data center environments.

The ns-3 simulator provides models for network elements such as network nodes (i.e., end hosts, routers, switches, and hubs), Ethernet and wireless links, and communication protocols (e.g., IPv4, IPv6, TCP, UDP), including their corresponding headers. An experimenter describes a network topology using C++ or Python.

Then, the program is compiled and linked with the library of network models.

5.1.3. OPNET

Optimized Network Engineering Tool (OPNET) [42] is a comprehensive development environment for specifying, simulating, and evaluating the performance of communication networks. It has a vast set of functionalities combined with an intuitive and straightforward user interface to model wired and wireless communication networks. The platform comprises a discrete-event simulator for modeling the behavior of network components following a hierarchical structure. Each level of the hierarchy defines the aspects of the model being simulated. Specialized libraries support existing protocols and allow experimenters to modify them and create customizable libraries. OPNET models are compiled into executable code, which can be debugged or executed, producing output data. The library packages include tools to allow experimenters to specify detailed models, identify the elements of interest in the model, run the simulation, and analyze the generated results.

Li *et al.* [85] proposed a Zigbee-compliant simulation model using the OPNET simulator. This new proposal improved the existing AODV routing algorithm to support node mobility, which is compatible with Zigbee protocols. The authors conducted a performance evaluation considering the proposed model and the Zigbee model in OPNET standard libraries. Results show that the proposed simulation model performs better than the original one. Rukmani and Ganesan [86] evaluated five types of scheduling algorithms (i.e., First-In-First-Out (FIFO), priority queueing, Weighted Fair Queuing (WFQ), Class-Based Weighted Fair Queuing (CBWFQ), and Low Latency Queuing (LLQ) Algorithm) using OPNET. Simulation results show that low LLQ improves the overall performance of the real-time applications more than all the other algorithms.

5.1.4. Comparison, Discussion, and Limitations

OMNeT++ is a widely used network simulator primarily used for discrete event simulations. It is widely used in academic and reserch environments due to its extensibility and modularity. One of its key strengths is the ability to model complex networks with various protocols effectively. OMNeT++ supports multiple programming languages, such as C++, allowing researchers to implement custom models easily. Additionally, it offers a graphical user interface (IDE) that simplifies the simulation setup process for users. OMNeT++ stands out for its strong community support, well-documented libraries, and regular updates. Its visual representation of simulations through the IDE aids in understanding and debugging network behaviors. However, OMNeT++ may be challenging for beginners due to its steep learning curve, especially for those not familiar with C++. The graphical IDE can also become less efficient with large-scale simulations, potentially impacting performance. Xian *et al.* [87] conducted a comparative analysis of OMNeT++ with other simulators to evaluate WSN systems. The authors implemented directed diffusion protocols to compare execution times between ns-2 and OMNeT++. Results showed that the simulation scenario in OMNeT++ is more scalable than in ns-2.

ns-3 is another widely used network simulator that focuses on realism and scalability. It provides a detailed and precise modeling approach, making it suitable for research and industry applications. As a primarily C++-based simulator, it offers good performance and allows users to create complex simulations. ns-3 emphasizes realism, aiming to accurately represent real-world network scenarios. This feature makes it suitable for evaluating new protocols and technologies realistically. However, the increased focus on realism might result in longer simulation times compared to other simulators. The learning curve for ns-3 can also be challenging for newcomers, especially those without prior programming experience. Zugno *et al.* [88] discussed the limitations of ns-3 to reproducing cellular networks involving LTE and mmWave deployments. The authors also provided future directions to fill the gaps for the main issues influencing the system's behavior. The recommendations focus on improving the channel model, antennas, applications, obstacles, and mobility models in ns-3. The authors anticipate that new cellular technology will rely on communication at high-frequency bands where the communication is strongly dependent on the spatial characteristic of the surrounding environment. Current models implemented in ns-3 do not consider these characteristics, especially when modeling multi-antenna systems.

OPNET differentiates from OMNet++ and ns-3 due to its simulation speed and graphical capabilities. It is widely used in industry settings for network planning and optimization. OPNET's strength lies in its ease of use and a visually intuitive graphical interface, which simplifies the creation and visualization of network scenarios. A major limitation of OPNET is its cost, as it is a commercial tool and may not be affordable for individual researchers or small academic institutions. Additionally, OPNET's closed-source nature restricts users from modifying the underlying simulation engine, limiting customizability. Hammoodi *et al.* [89] evaluated OPNET based on availability, reliability, response time, and other QoS parameters. The authors used the ZigBee protocol in three common topologies (i.e., star, mesh, and tree). The results suggested improvements related to modeling security protocols, predicting the behavior of unicast and multicast traffic, and estimating energy consumption.

5.2. Mobile Networks

Network simulators are increasingly being used for the evaluation of mobile networks, which include cellular networks such as 3G, 4G, and 5G, as well as other wireless networks like Wi-Fi, Bluetooth, and Zigbee. Mobile networks are complex systems that involve multiple layers of protocols, dynamic network topologies, and a diverse range of mobile devices. Network simulators provide an effective way to evaluate the performance, behavior, and scalability of mobile networks in a controlled and repeatable environment.

One of the main motivations for using network simulators in mobile networks is to reduce the time and cost of evaluating new technologies, protocols, and applications. Mobile network simulators can easily replicate complex network topologies, including radio access networks, core networks, and backhaul networks, allowing researchers

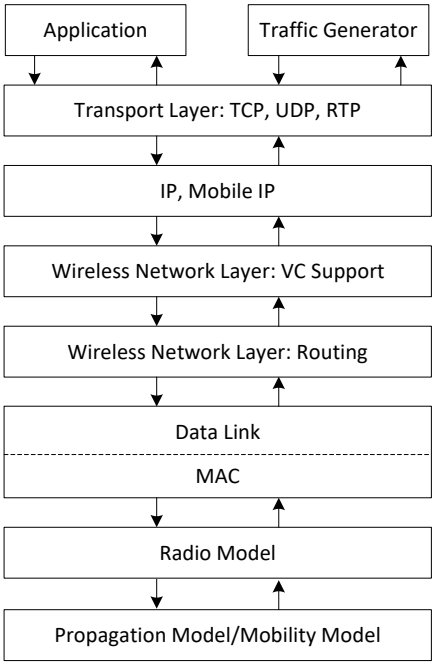

Figure 5: GloMoSim Architecture [43].

and developers to test their new technologies and protocols in a controlled environment. This can significantly reduce the time required for testing and evaluation and enable faster development and deployment of new technologies.

Another key motivation for using network simulators in mobile networks is to evaluate the network's performance under various network conditions, such as different traffic loads, mobility patterns, and environmental conditions. Network simulators can simulate a wide range of network conditions, allowing researchers to study the impact of these conditions on network performance, such as data rates, latency, and network capacity. This can help in the design of efficient and reliable network protocols that are adaptable to different network conditions.

Additionally, network simulators can be used to evaluate the impact of network failures and disruptions on the network's performance. Researchers can simulate various network faults, such as node failures, link failures, and network congestion, to evaluate the resilience and fault tolerance of the network. This can help in the design of robust network protocols that can recover from network failures and ensure uninterrupted communication.

Finally, network simulators can also be used to evaluate the security of mobile networks. Researchers can simulate various security threats, such as attacks on user data, identity theft, and network intrusion, to evaluate the security of the network and identify potential vulnerabilities. This can help in the design of secure network protocols and in the development of security measures that can protect the network from potential attacks.

This category includes the network simulators specialized for modeling mobile networks, which include Wireless Sensor Networks (WSN) and mobile networks. These simulators provide insights into the efficiency of a protocol, the energy consumption of wireless devices, and the methods that can optimize existing protocols. Experimenters can also simulate physical events such as the

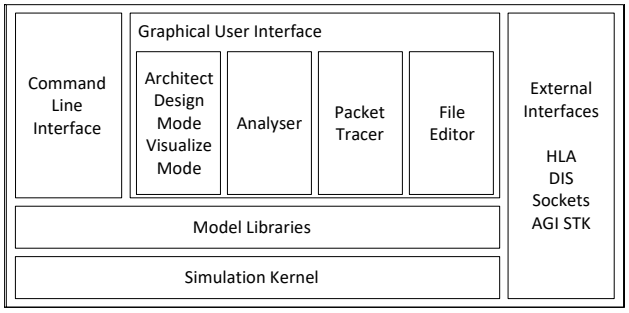

Figure 6: QualNet Architecture [92].

permeability of the environment, the dynamics of a moving sender/receiver, and the interferences produced by different sources.

5.2.1. GloMoSim

The Global Mobile Information System Simulator (GloMoSim) [43] is a simulator for wireless networks based on sequential and parallel models. The simulator comprises library modules that simulate wireless communication in the protocol stack. The libraries in GloMoSim rely on the Parallel Simulation Environment for Complex Systems (PARSEC) [90], a parallel simulation language based on C. PARSEC uses a message-based approach to implement discrete-event simulation, where physical processes are modeled by simulation objects called entities. Events are represented by the transmission of time-stamped messages among corresponding entities. Experimenters can also create and include custom libraries to test with new protocols due to the composability and extensibility of GloMoSim's design. Simulations in GloMoSim can be implemented on both shared memory and distributed memory computers. GloMoSim can execute experiments with several synchronization protocols, such as the null message [91] and conditional event algorithms.

Figure 5 shows the architecture of GloMoSim for ad hoc networks which consist of a networking stack decomposed into several layers. Each layer implements a set of models which executes the actions corresponding to their protocol stack. For instance, the data link layer can implement several protocols, such as Carrier Sense Multiple Access (CSMA), Multiple Access Collision Avoidance (MACA), Floor Acquisition Multiple Access (FAMA), and other protocols that have been modeled in the GloMoSim library. The network layer implemented flooding protocol and flat distance vector routing protocol (DSDV) [43]. Moreover, GloMoSim's library also contains models for the physical layer (i.e., propagation model/mobility model) that includes a free space model that calculates signal strength based only on the distance between every source and receiver pair, an analytical model for computing signal attenuation using a logarithmic normal distribution, a fading channel model that reproduces the effect of multipath, shadowing, and fading to calculate the signal strength. The simulator also incorporates as built-in libraries the Free BSD implementation of the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Real Time Protocol (RTP). GloMoSim provides a standardized API set to exchange messages between adjacent layers.

Ahvar and Fathy [93] used GloMoSim to evaluate the energy consumption of routing protocols in high-density ad hoc networks. The authors performed their analysis as a function of different network loads, mobility, and size. The paper concludes by reporting a detailed critique of the three protocols, focusing on the differences in their dynamic behaviors that can lead to performance differences. A similar study was conducted by Kumari *et al.*, which compared the performance of MANET routing protocols such as AODV, Location Aided Routing 1 (LAR1), and Wireless Routing Protocol (WRP).

5.2.2. QualNet

QualNet [44] is a commercial network simulation tool used for modeling and analyzing communication networks. The QualNet simulator has a modular architecture, which allows the user to customize the simulator by selecting different modules and configuring their parameters. QualNet is a discrete-event simulator for heterogeneous networks and distributed applications. It enables users to test custom protocols, develop prototypes, and run large-scale networks by providing a simulation suite. The company supporting the simulator [94] also provides an emulation tool for running cybersecurity tests. QualNet is based initially on GloMoSim and supports operationally accurate contexts. It provides a suite of tools for configuring, tracing, and analyzing traffic events. Figure 6 shows QualNet's architecture. QualNet comprises the components to build comprehensive network models and generate statistics that reflect actual or projected performance. It includes high-fidelity models for a wide variety of network devices and from all protocol stack layers. Moreover, QualNet allows experimenters to include dynamic interactions that involve the Human-in-the-Loop (HITL) interface, which allows manipulating modules and modifying traffic through the HITL environment.

The QualNet simulator architecture consists of four major components. First, a scenario modeler specifies network topologies, node properties, and traffic patterns. The user can also import external data sources, such as terrain maps and traffic traces, to create more realistic scenarios. Second, a network emulator provides a virtual environment where the network scenario is executed. The network emulator is responsible for managing the network traffic, simulating the behavior of network devices, and enforcing the network protocol standards. Third, the performance evaluator collects performance metrics during the simulation, such as throughput, delay, and packet loss. The user can customize the performance metrics to be collected based on their research needs. Fourth, the analyzer provides tools to analyze the simulation results and visualize the network behavior. The user can create custom reports and graphs to present the simulation results in a meaningful way. The modular architecture of the QualNet simulator enables flexibility in creating and analyzing network scenarios, making it a powerful tool for network researchers and engineers.

Shuaib [95] conducted a performance evaluation of the IEEE 802.16e WiMAX [96] standard using QualNet. The study focused on mobile WiMAX to deliver broadband wireless access to mobile users. The experiment considered various scenarios with different levels of interference modeled with the QualNet simulator. Results showed the impact of factors such as load and mobility. The authors showed how these factors affect the performance of WiMAX in a single-cell environment where the performance measurements (i.e., throughput) are affected by the end-to-end delay and jitter. Goyal *et al.* [97] used QualNet to analyze the performance of routing protocols in wireless sensor networks. They considered ad hoc protocols such as Ad-hoc On-Demand Distance Vector (AODV), Dynamic MANET on-demand Routing Protocol (DYMO), Optimized Link State Routing Protocol (OLSR), and IERP and measured performance metrics such as the average jitter, throughput, end-to-end delay, error rate, and queue length. Results show that with the random waypoint mobility model, the Interzone Routing Protocol (IERP) protocol gives the highest throughput compared to DYMO, AODV, and OLSR. Latkoski *et al.* [98] extended the QualNet simulator environment by introducing a novel simulation technique using the Specification Description Language (SDL). This solution overcomes the difficulties of re-coding QualNet's components. The proposed methodology based its approach on developing the wireless heterogeneous network standard IEEE 802.21 [99]. The authors present some use cases, which include data acquisition techniques, dynamic network parameters recalculation, and buffer management methods.

5.2.3. COOJA

COOJA [45] is a sensor network simulator that runs on the Contiki OS, a portable operating system designed to model resource-limited devices. This event-driven simulator supports pre-emptive multithreading on a per-process basis. The platform also supports a full TCP/IP stack implementation that can be extended using interfaces representing a sensor node property such as the position, a button, or a radio transmitter. New interfaces can be easily created and added to the simulation environment, such as custom radio mediums to forward network data and underlying hardware platforms that reproduce heterogeneous networks. A Java interface connecting with the underlying OS allows real application code to run on COOJA models without requiring modifications. The platform also provides a GUI that facilitates defining topologies and configuring the parameters of the simulated devices.

Finne *et al.* [100] proposed a multi-level data trace generator that enables data logging at different levels while maintaining a global time. This system aims to test intrusion detection systems (IDS) in wireless multi-hop networks with no entity that can overhear all packets. Results show that the system generates traces fast enough to serve as input to user-defined algorithms used in IDS. The authors also highlight that the proposed approach is a valuable contribution to the research community as it generates time-synchronized logs at different levels. Jabba and Acevedo [101] designed and implemented an application that allows the creation of other energy estimation models. The system runs on top of COOJA and shows the real-time behavior and heat map of energy consumption traces. The authors sustain that their approach can help researchers to monitor real-time topology deployments, node disconnections, and battery depletion. Zenalabdin *et al.* [102] employed

the COOJA simulator to evaluate IoT protocols over dense and sparse network topology, including application layer protocols, namely, Message Queueing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), and transport layer protocols such as UDP and TCP. The authors evaluated performance metrics, including power intake, radio responsibility cycle, and average inter-packet arrival time.

5.2.4. ns-3 LENA

ns-3 LENA [103] is an advanced tool designed for modeling and evaluating next-generation networks. Their architecture and design principles encompass a robust framework, employing various modeling techniques to simulate complex networking scenarios accurately. This tool offers a wide range of features and functionalities, including support for multiple frequency bands, diverse network elements like base stations and user equipment, and integration with 5G technologies. The protocol stack covers standard protocols, mobility management, and Quality of Service (QoS) mechanisms, enabling researchers to assess their simulation accuracy. Researchers have utilized ns-3 LENA for performance evaluation, studying metrics like throughput, latency, and energy efficiency in various network settings. Bojović *et al.* [104] designed and implemented a beamforming technique using a Sounding Reference Signal (SRS)-based channel estimate in the ns-3 5G-LENA module. The study focuses on enhancing the accuracy of beamforming simulations in next-generation networks. The researchers proposed a novel approach that utilizes SRS-based channel estimates to improve the realism of beamforming models in the ns-3 5G-LENA module. By incorporating real-world channel characteristics, the simulation outcomes better reflect actual network behavior.

5.2.5. WiGig module

The WiGig module [105] in the ns-3 network simulator is a tool designed to model and simulate 60 GHz wireless networks (WiGig). WiGig is a high-speed wireless communication technology operating in the 60 GHz frequency band, capable of delivering multi-gigabit data rates over short distances. The WiGig module in ns-3 provides support for various WiGig functionalities, including beamforming, spatial reuse, and channel bonding. Researchers can use this module to investigate the performance of WiGig networks under different scenarios and evaluate the impact of various parameters on network performance. The module supports both single-node and multi-node simulations, enabling the analysis of WiGig networks with multiple devices. Additionally, it incorporates realistic channel models, ensuring accurate representations of real-world wireless propagation environments. Assasa *et al.* [106] employ the WiGig module to evaluate the implementation of the IEEE 802.11ay standard. The authors focus on the specifics of IEEE 802.11ay operations such as the 802.11ay frame structure, channel bonding, new beamforming training procedures, quasi-deterministic MIMO channel support, and single-user MIMO and multi-user MIMO beamforming training.

5.2.6. Comparison, Discussion, and Limitations

GloMoSim is a scalable network simulator designed for large wireless networks. It aims to simulate mobility scenarios accurately and supports various communication protocols. Its strengths lie in modeling wireless networks with a large number of mobile nodes effectively. GloMoSim focuses on simulating large-scale wireless networks, making it suitable for research in mobile communication systems. However, GloMoSim's development has been discontinued, and it may lack some features and updates available in more recent simulators. Khan *et al.* [107] discussed the limitations of GloMoSim to recreate large-scale WSN experiments compared to other simulators. The authors conducted a comparative analysis that included accessibility, user support, availability of modules, extensibility, and scalability. Although GloMoSim scales well and has a wide range of modules implemented, the tool is outdated and only simulates wireless networks.

QualNet is a commercial network simulator known for its extensive library of pre-built models and a user-friendly graphical interface. It provides a wide range of communication and network models, making it suitable for various scenarios. QualNet provides a large collection of pre-built models, which simplifies simulation setup for users. The main limitation of QualNet is its cost, as it is a commercial tool and may not be affordable for all users, especially individual researchers or small institutions. Tan *et al.* [108] compared and analyzed experimental results between QualNet and ns-2. The authors found discrepancies in modeling the physical layer and antenna diversity. Results indicate that QualNet has an optimistic channel, whereas ns-2 presents more conservative settings. They proposed compensation procedures to mitigate the discrepancies between both simulators.

COOJA is a network simulator specifically designed for WSNs. It is an integral part of the Contiki OS development environment and allows researchers to evaluate and test WSN protocols effectively. COOJA's strength lies in its focus on Wireless Sensor Networks, providing an environment tailored to WSN simulations. However, COOJA may lack some advanced features available in more general-purpose network simulators. Sundani *et al.* [109] analyzed the limitation of the COOJA simulator by conducting a comparative study. The authors found that COOJA presents relatively low efficiency while simulating many nodes with several interfaces. In this scenario, the tool incurs many calculations, especially when plug-ins are started and registered as observers to interfaces. Moreover, their findings show that COOJA only supports a limited number of simultaneous node types resulting in a system restart when the number of nodes is exceeded.

ns-3 LENA is an extension of the ns-3 simulator, focusing on modeling Local and Enterprise area Networks (LANs and EANs). It aims to provide a detailed representation of these networks, allowing researchers to evaluate LAN/EAN-specific scenarios and protocols. ns-3 LENA's strength lies in its specialization for LAN and EAN simulations, which is beneficial for research in these areas. The limitation of ns3-LENA is its narrow scope, as it may not be suitable for simulating other types of networks.

The WiGig module is an extension for the ns-3 simulator that specifically models Wireless Gigabit

communication technology. WiGig operates in the 60 GHz frequency band and provides high-speed wireless data transfer. The WiGig module's strength lies in its ability to accurately model WiGig communication and evaluate high-speed data transfer scenarios. The limitation of the WiGig module is its focus solely on WiGig technology, making it less suitable for simulating other types of networks or protocols.

5.3. Education

This category comprises the simulators used to deliver training and hands-on materials to guide learners on network configuration and troubleshooting. These simulators are mostly used in an academic environment. The key feature of these simulators is their usability, which reduces the learning curve on how to use the tool and immerses the learner in the simulation environment. These simulators typically comprise a user-friendly GUI to permit a high level of customization.

5.3.1. Cisco Packet Tracer

Cisco Packet Tracer [46] is a discrete-event simulator used for educational and training purposes. Packet Tracer is a Cisco product that supports teaching in networking courses and training certification tracks. Packet Tracer comprises network simulation, visualization, and collaboration capabilities to help the learner understand computer network principles and improve the learner's practical skills. Its simulation environment provides a wide variety of features and functions to enable students and educators to create and test complex networking scenarios. Packet Tracer has a graphical user interface that allows users to drag and drop networking devices (e.g., routers, switches, PCs, servers, IoT devices, and others), connect devices via networking cables (e.g., coaxial, copper straight-through, serial DTE/DCE), and support a command line interface for configuring the network devices. Hence, this simulator has commonly been used as the preferred tool in educational initiatives to instruct computer network concepts and troubleshooting procedures.

Noor *et al.* [110] evaluated the effectiveness of the Cisco Packet Tracer as a learning tool. The authors focused on three aspects: the student's perception of the simulator's capabilities to learn networking concepts, the ease of use the tool provides, and the degree of confidence the student acquires in learning routing concepts. Results reported that Cisco Packet Tracer facilitated students in learning routing protocols. Moreover, the simulator also helped the students to compare the behavior and performance of different routing protocols. Gwangwava *et al.* [111] used Cisco Packet Tracer to simulate IoT systems which include smart things—home, smart city, industrial control, and power grids. The authors focused on designing a generic IoT-based control system to monitor chemical variables such as carbon and sulfur emissions. The system also integrated the models of human-machine interfaces that facilitate operators to verify the measurable parameters. Allison [112] discussed the impact of Cisco Packet Tracer on undergraduate education, identifying the benefits and challenges of the tool. The paper reports practical implementations of the tools in the literature and provides practical recommendations to help educators and curriculum designers to create more effective and interactive networking sessions.

5.3.2. GNS-3

The Graphical Network Simulator 3 (GNS3) [27] is a network experimentation framework that supports multi-vendor models. The platform also supports the emulation of physical devices and the simulation of models available in GNS3's marketplace. GNS3 supports different network operating systems developed to run on hardware appliances. The platform also provides a hardware-independent interface for the operating systems, allowing virtual machines on the local host. Moreover, the tool allows experimenters to install custom images into a component to virtualize its operations. GNS3 has a built-in GUI and can easily inter-operate with other network applications such as Wireshark [76], and Oracle VirtualBox [113]. The main purpose of GNS3 is to train learners for certification exams and reduce exam preparation costs.

GNS3 is also a tool used by experimenters to validate their proposed systems. Velieva *et al.* [114] developed a mathematical model to validate the Random Early Detection (RED) Active Queue Management (AQM) in GNS3. They used a Cisco router image and a traffic generator to observe the interaction of network traffic with the proposed RED AQM implementation. Gil *et al.* [115] evaluated the performance of GNS3 for conducting network experiments in a classroom. The authors developed a set of hands-on sessions that included basic configuration, routing, and data transfer experiments. The feedback shows that students understood the basic topics as if they were in a real computer laboratory. Emiliano and Antunes [116] proposed an extension for GNS3 to generate valid configuration files for network devices automatically. Their implementation could automatically produce an initial IP and routing configuration of all the Cisco virtual equipment using the GNS3 specification files. Mihaila *et al.* [117] employed GNS3 as a testbed to detect misconfigurations in network devices. Castillo *et al.* [118] simulated a portion of the GEANT backbone in GNS3 to offer consulting services to ISP companies.

5.3.3. Comparison, Discussion, and Limitations

Cisco Packet Tracer is a network simulator developed by Cisco Systems. It is primarily designed for educational purposes, providing a user-friendly interface to simulate network configurations and troubleshoot issues. It is often used in networking courses and certifications. Cisco Packet Tracer is a time-based simulator that is primarily used for education. Although some used Packet Tracer to conduct research, the platform is mostly used as an entry-level training resource for teaching classrooms, preparing for certification exams, and developing troubleshooting skills. Packet Tracer is simple and easy to use, making it ideal for beginners and educational settings. It offers a large library of Cisco devices and provides a visual representation of network configurations. However, Packet Tracer has some limitations in terms of scalability and support for non-Cisco devices and protocols. It may not fully represent complex network scenarios and might not be suitable for advanced research or professional network simulations.

GNS3 is an open-source network simulator that offers

Table 2: Simulators comparison.

Ref.	Simulator	Description	Interface	License	Parallelism	WSN Support	GUI	Custom Modeling
[40]	OMNeT++	OMNeT++ is an object-oriented discrete event simulation environment to test communication protocols, multicore applications, and other distributed or parallel systems.	C++/Python	Open source	No	Yes	Limited	Supported
[26]	ns-3	ns-3 is an open-source environment based on discrete-event simulation. It allows importing and creating complex simulation models using object-oriented programming languages such as Python and C++.	C++/OTcl	Open source	No	Yes	Yes	Supported
[42]	OPNET	OPNET is a discrete-event simulator leveraged by object-oriented and hierarchical modeling, debugging, and analysis. It supports hybrid simulation, analytical simulation, and parallel simulation, among other features.	Proto-C	Proprietary	Yes	Yes	Yes	Supported
[43]	GloMoSim	GloMoSim is a simulator that leverages parallel processes to reduce the execution time of detailed high-fidelity models of large-scale wireless networks.	PARSEC	Open source	Yes	Yes	No	Supported
[44]	QualNet	QualNet is used to model large heterogeneous networks and distributed applications. It enables the parallel execution of simulated models and facilitates the analysis of post-simulation and statistical variables via its GUI.	C++/NED	Proprietary	MPI/PVM	No	Yes	Supported
[45]	COOJA	COOJA is a WSN event-driven simulator designed to model resource-limited devices. The simulator supports pre-emptive multithreading on a per-process basis.	Contiki OS	Open source	Yes	Yes	Yes	Supported
[103]	ns3-LENA	ns3-LENA integrates various modeling techniques to simulate 5G environments. The tool integrates a simulated channel and physical layer model with a full stack implementation.	C++/OTcl	Open source	Yes	Yes	No	Supported
[105]	WiGig module	The WiGig module is an ns-3 wireless interface controller based on the WLAN IEEE 802.11ad/ay standards.	C++/OTcl	Open source	Yes	Yes	No	Supported
[46]	Cisco Packet Tracer	Cisco Packet Tracer is a simulator for wired and wireless networks primarily used for education. It facilitates building complex topologies and reproduces inter-protocol scenarios. Its user-friendly GUI helps the user to visualize the experimentation scenario.	Cisco IOS	Academic	No	Yes	Yes	Not Supported
[27]	GNS-3	GNS3 is a network experimentation framework that supports multi-vendor models. The platform also supports the emulation of physical devices and the simulation of models.	Dynamips	Open source	No	Yes	Yes	Not Supported

more advanced capabilities compared to Packet Tracer. It allows users to create complex network topologies and supports a wider range of devices and operating systems, including Cisco devices. Its integration with VMs enables users to create custom images and test more specific systems. Cisco Packet Tracer is a network simulator that is mainly used to design, configure and test networks for training purposes. GNS3, on the other hand, is used to design and simulate complex networks at the professional level. Cisco Packet Tracer is a user-friendly tool that is easy to use and does not require any prior knowledge of programming or command-line interface (CLI). GNS3, on the other hand, presents more complexity by requiring some knowledge of CLI and programming. Cisco Packet Tracer has a limited range of network devices that can be simulated, such as routers, switches, and hubs. GNS3, on the other hand, can simulate a wide range of network devices, including virtual machines, routers, switches, and firewalls. Cisco Packet Tracer is free for networking academy students and instructors, but there are limitations to its functionality. GNS3 is a completely open-source software that is free for everyone to use without any limitations. However, GNS3's complexity may be challenging for beginners, and it requires a good understanding of networking concepts and configurations. Setting up GNS3 correctly may also be more involved compared to the straightforward setup of Packet Tracer.

5.4. Summary and Lessons Learned

Network simulators are essential tools to evaluate the feasibility and effectiveness of novel designs, architectures, and algorithms for network systems. They allow experimenters to monitor the overall system's performance in a controlled environment, with different scenarios and parameter settings, before implementing

Table 3: Most common research conducted using the surveyed simulators.

Research Topic	OMNeT++	ns-3	OPNET	GloMoSim	QualNet	COOJA	ns3-LENA	WiGig module	Cisco Packet Tracer	GNS3
Routing Protocols	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Transport Protocols	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Link Layer Protocols	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
IoT Protocols	✓	✓	✓	✓	✓	✓	×	×	✓	✓
5G	✓	✓	✓	✓	×	×	✓	✓	✓	✓
WSN	✓	✓	✓	✓	✓	✓	×	×	✓	✓
Network Attacks	✓	✓	✓	✓	✓	✓	×	×	✓	✓
SDN	✓	✓	×	×	×	×	✓	×	×	✓
ML-based Applications	✓	✓	×	×	✓	×	✓	×	×	✓

them in real hardware. Table 2 provides a summary of the main characteristics of the surveyed simulators.

Each simulator serves a specific purpose and caters to different research areas. For example, OMNeT++ and ns-3 are versatile and widely used for general network simulations, while COOJA focuses on Wireless Sensor Networks. Understanding the specialization and flexibility of each simulator helps researchers select the most suitable tool for their research needs. OMNeT++ and ns-3 stand out as open-source simulators, offering accessibility, customization, and a strong user community. These simulators are particularly valuable for researchers with limited budgets and those seeking to contribute to open-source projects. OPNET and QualNet are commercial simulators with comprehensive features and user-friendly interfaces. They may require licensing costs but provide extensive support and pre-built libraries, making them ideal for industrial and educational applications. GloMoSim, COOJA, ns3-LENA, and the WiGig module are specialized in wireless communication and mobility scenarios. Researchers focusing on wireless technologies should consider these simulators for their evaluations. Researchers should be aware of each simulator's limitations, such as limited device support, scalability constraints, or specialized focuses that may not cover all research requirements.

Table 3 shows the most common research topics that used the surveyed simulators. These topics are derived from papers that validated their proposed systems, protocols, and applications. The table shows that OMNeT++, ns-3, and GNS3 support all the research topics listed in the table. Some works use OMNeT++ to research a wide range of topics. This simulator is preferably used for discrete-event simulations that include mobile networks, 5G, Wireless Sensor Networks (WSN), and other distributed applications. ns-3 is a simulator employed for modeling scenarios that involve most of the surveyed research topics. Papers collected from 2005 to 2022 show that the most popular network simulator in wireless networks is ns-3.

On the other hand, GNS3 capabilities enable running experiments in all the surveyed research topics. However, there are not too many papers that used GNS3 to validate their proposed systems. This is due to the purpose of GNS3, which is education, troubleshooting, and preparing students to obtain certifications. However, the ease of use, the wide range of emulated hardware available in its marketplace, and the integration of VMs with simulated models make GNS3 an excellent tool for conducting applied research. Although some research projects used Cisco Packet Tracer [111, 119–121], the tool is mainly used for teaching classrooms [112, 122, 123]. The user-friendly interface, available training material, and support of the major network vendor make Cisco Packet Tracer a suitable tool for students that want to learn networking concepts.

6. Network Emulators

Network emulation is widely used for rapid prototyping and testing the behavior of applications under various network conditions. It is an alternative approach to network simulation and a cost-effective option compared to real hardware. Emulation allows interaction with real network traffic and modifies it based on software implementations. Network emulators facilitate the creation of network experimentation scenarios that involve bandwidth limitation, packet delay, jitter, loss, duplication, and reordering. Emulators must reproduce the desired network conditions accurately to prevent unreliable experiments with misleading results. Figure 7 shows the layers that comprise the most typical network emulation architectures. These architectures are full virtualization, which consists of VMs running on guest OS or in a bare-metal hypervisor. The other type of virtualization implements containers to isolate OS resources into independent instances that are part of the OS. This architecture consumes fewer resources than full virtualization, scaling up the instances running in a single OS.

6.1. Container-based Emulation

Container-based emulation has become an increasingly popular approach for emulating network topologies and testing network protocols and applications. The use of container-based emulation is primarily driven by the need for more efficient and scalable emulation solutions that can support the development and testing of complex networking scenarios. One key advantage of container-based emulation is its ability to provide lightweight and isolated network environments that can be easily replicated and scaled up or down as needed. This makes it easier to test network protocols and applications under a variety of conditions and scenarios, including complex multi-domain networks and distributed systems. Another

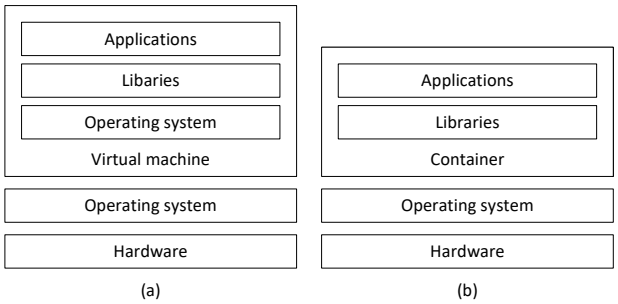

Figure 7: Network emulation architectures. (a) Virtualization-based emulation. (b) Container-based emulation.

advantage of container-based emulation is its flexibility and ease of use.

Containers can be quickly and easily deployed on any host system, making it easy to set up and tear down emulation environments as needed. This makes it ideal for developers who need to quickly iterate on their network designs and test their applications in different network environments. Container-based emulation also provides greater control over the network environment, allowing developers to isolate and modify specific components of the network stack. This enables them to test the impact of different network configurations and protocols on the performance and behavior of their applications. With this approach, researchers can significantly reduce the cost and complexity of setting up and managing large-scale network testbeds. By using containerization technology, developers can create virtual networks that are more scalable, efficient, and easier to manage than traditional physical testbeds. The motivation for using container-based emulation is driven by the need for more efficient, scalable, and flexible network emulation solutions that can support the development and testing of complex network scenarios. With its lightweight, scalable, and flexible nature, container-based emulation has become an increasingly popular approach for network developers and researchers. This category includes schemes that use container-based emulation to reproduce the behavior of network components. Schemes using this abstraction instantiate fewer resources than the ones using a single VM to build network topologies. OS resources used to implement container-based emulators can be Linux Containers (LXC) [124], Docker containers [125], and other types of OS-level virtualization tools.

6.1.1. Mininet

Lantz *et al.* [47] introduced Mininet, a network emulator for prototyping large networks using process-level virtualization, a lighter form of virtualization in which OS-level system resources are shared. Initially, Mininet was developed for rapidly prototyping large networks (i.e., up to 1000 end hosts) on devices with constrained resources. The use of network namespaces allows scaling up to a large number of end hosts, switches, and controllers. Network topologies in Mininet are deployed in seconds and can be defined using programming languages such as Python. This feature also facilitates sharing topologies with other experimenters who can modify them according to their research needs. Additionally, the code used in Mininet is the same as in production networks. Mininet shares system resources such as page tables, kernel data structures, and the file system. Mininet achieves better scalability than VM-based systems, permitting a more significant number of small virtual hosts on a single system. Moreover, Mininet provides an environment for quickly implementing and testing a network feature on a large topology using application traffic. This feature is enabled by combining lightweight virtualization with an extensible CLI and APIs. Figure 8(a) shows an example of a topology created in Mininet with two hosts, an OpenFlow switch, and a controller. In Figure 8(b), it is observed that the topology instantiates two network namespaces to represent the hosts, where a virtual Ethernet pair (Veth) emulates a wire connecting two virtual interfaces (e.g., h1-eth0 and s1-eth0). Packets are sent from one interface to another, and the hosts can see these interfaces as fully functional Ethernet ports. The virtual Ethernet pairs are bridged to an OpenFlow switch. However, they can also be bridged to other software switches or interfaces supporting hardware NICs. Hosts h1 and h2 are network namespaces that act as containers for network states. These instances support kernel-level processes with exclusive ownership of interfaces, ports, and routing tables (e.g., ARP and IP tables).

Network namespaces isolate resources, meaning that Mininet hosts can emulate two servers listening to private interfaces on the same transport protocol port, such as port 80. From the user perspective, a Mininet host is a simple shell instance that shares the host OS file system. The OpenFlow switch emulates the forwarding logic as it will occur in a hardware switch. Finally, Mininet can support a wide variety of controllers as long as they can support IP-level connectivity. For example, the controller can run locally in the host VM, on another server, or in the cloud.

Network topologies in Mininet can be defined using the CLI, a Python script, or MiniEdit, a GUI that facilitates interconnecting components in a drag-and-drop manner. After running a topology, an experimenter can control and orchestrate tests using a script that instantiates the network components. In the script, the experimenter can invoke commands used in servers running on a production network and reproduce network conditions using the traffic control (tc) commands available in Linux distributions [126]. The traffic control in Linux offers a rich set of functions, including queueing disciplines (qdiscs) such as Token Bucket Filter (TBF), Network Emulator (NetEm), Active Queue Management (AQM) algorithms, packet filters, and others. These tools allow the implementation of the network conditions observed in production networks. Additionally, the experimenter can collect network measurements from a controller or the OS. Lastly, the most relevant limitation of Mininet is the lack of performance fidelity with high loads. This limitation is directly related to the number of CPUs and memory available in the host running Mininet. Typically, the Linux scheduler multiplexes CPU resources over time which does not ensure that packets will be forwarded at the same rate by all switches. Mininet uses software switches whose forwarding rate is less deterministic than hardware switches when managing high traffic loads. For example, when installing new routes, software switches implementations perform linear lookups

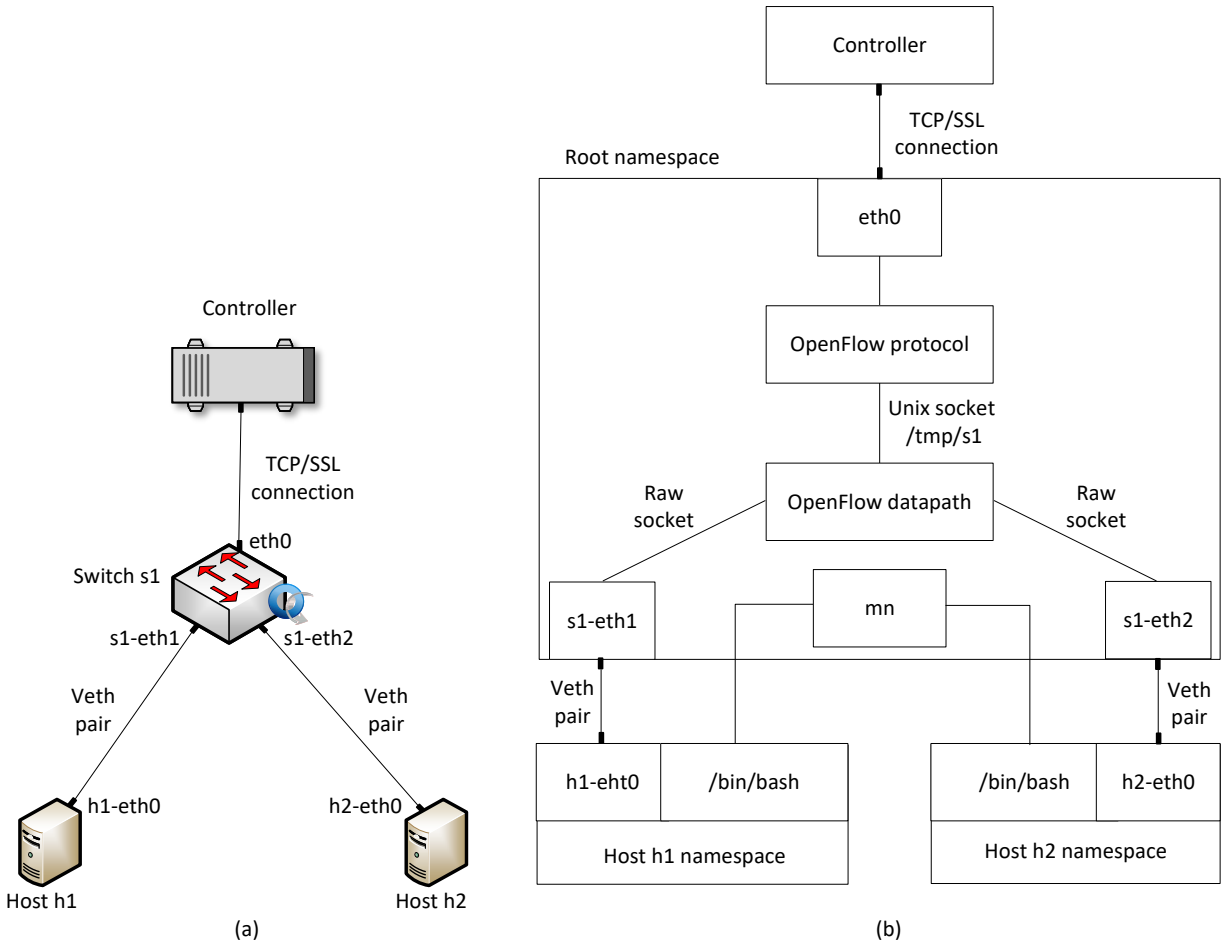

Figure 8: Mininet components and connections of a topology with two hosts, an OpenFlow switch, and a controller. (a) Logical topology. (b) Namespaces representation and virtual Ethernet pairs (Veth).

(i.e., $O(n)$), whereas hardware switches usually rely on Ternary Content Addressable Memories (TCAMs) to implement a constant lookup (i.e., $O(1)$).

In order to mitigate the performance discrepancies observed in Mininet, Heller [1] presented Mininet-HiFi to enable a wide variety of network experiments that are realistic, verifiable, low-cost, and reproducible. The proposed system combines the 2010 version of Mininet with a fidelity monitor that tracks the network invariants, which are performance metrics that evaluates the realism of a topology running on Mininet. Mininet enables processes to have an isolated view of the system's resources (i.e., process ID, user names, file systems, and network interfaces while running on the same kernel) by employing Linux network namespaces. When the user creates a topology in Mininet, a network namespace is associated with a container that has a virtual network interface, a file system with its associated data that includes ARP caches and routing tables. A network invariant is referred to a timing property that is consistent in any network that comprises wired links and output-queued switches. Network invariants should be independent of the topology, traffic patterns, application behavior, and transport protocol. Mininet-HiFi aims to provide an emulation tool that replicates the behavior of real hardware. The fidelity monitor keeps track of the network invariants to measure the faithfulness of the network measurements. Results show that Mininet-HiFi can quantitatively reproduce hardware results of previously published papers [127]. The experiments include queueing disciplines, transport protocols, routing protocols, and performance evaluation using simple (e.g., two-way, fork out single-SW, and dumbell) and complex topologies [128, 129]. The author highlights research papers should include the experimentation platform (i.e., the VM with Mininet and the scripts to run the experiments) to facilitate result validation, enhance collaboration, and enable future works on current experiments.

6.1.2. Containernet

Peuster *et al.* [48] presented Containernet, an open source Network Function Virtualization (NFV) prototyping platform that supports the creation and local execution of complex Service Function Chains (SFCs). Containernet is a Mininet fork that integrates Docker containers. With Containernet, experimenters can create prototypes that explicitly support hybrid SFCs composed of container-based and VM-based VNFs combined in a single chain. The emulator fully integrates VMs with the existing Mininet and Containernet APIs, which allows a user to add a fully-featured VM to an emulation topology with a single Python command that expects the path to the VM image to be used as an additional parameter.

Crichigno *et al.* [57] employed Containernet to deliver virtual hands-on training in topics such as SDN, P4-programmable data planes, Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), Multi-Protocol Label Switching (MPLS), and others. The authors used Containernet on top of VMs hosted by an

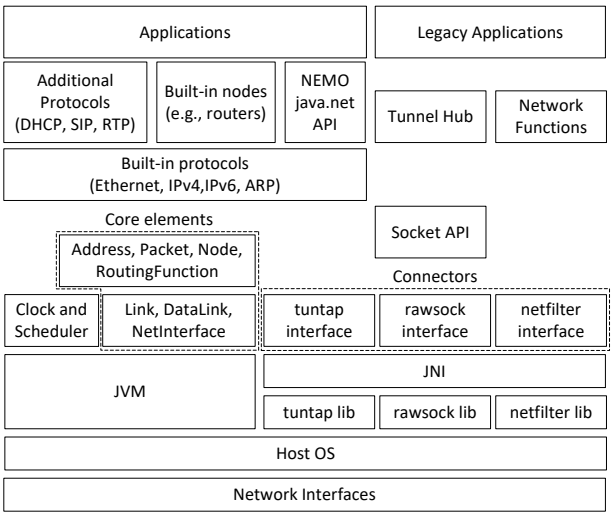

Figure 9: NEMO architecture [49].

academic cloud. The users remarked on the high quality of the lab manuals and the consistency of the steps with the results observed in the VM. Moreover, instructors that used the virtual labs indicated that they would likely adopt the training material in their institutions. Sen *et al.* [130] used Containernet to create a testbed to test cyberattacks on smart grids. The platform facilitates scenarios to test defensive and offensive use mechanisms, where an attacker can perform network scans, find vulnerabilities, exploit them, gain administrative privileges, and execute malicious commands. The authors present a built-in Intrusion Detection System (IDS) from these scenarios that analyze generated network traffic using ML-based anomaly detection approaches.

6.1.3. NEMO

Veltri *et al.* [49] developed NEMO, a Java-based network emulator used to experiment with a single link, a portion of a network, or an entire network. NEMO can work in real and virtual environments depending on the experiment requirements and goals. It can run as a single instance in a VM or multiple VMs. This feature allows scaling up experiments and supporting resource-intensive distributed applications. NEMO can also virtualize the execution of third-party Java applications on top of virtual nodes. These nodes can be attached to an emulated or external network. This network emulator addresses the limitations of similar platforms that are too specific for supporting experiments or hard to modify. NEMO aims to maximize simplicity, usability, and flexibility by providing a complete fresh re-implementation of the IP stack. Therefore, this platform supports general-purpose experiments that involve IP-based networks and standardized protocols. Figure 9 shows NEMO's architecture, which is designed as a highly structured network emulator comprising modules to facilitate the change or creation of new components. The emulator's core includes basic network components such as packets, network addresses, network interfaces, nodes, links, and routing functions. The platform also provides many built-in protocols, such as IPv4, IPv6, ARP, ICMP, UDP, TCP, and others.

Amoretti *et al.* [131] used NEMO to validate a publish/subscribe framework for industrial IoT. The authors proposed a communication framework based on Message Queuing Telemetry Transport (MQTT) broker bridging, enabling dynamic interoperability across different assembly lines or industrial sites in an Industrial IoT scenario. The system aims at ensuring a higher degree of isolation and control over the information flows, thereby increasing the overall security of the communication among nodes. The proposed solution also provides security features to support dynamic authentication and authorization. This feature is implemented and evaluated in a small-scale industrial IoT testbed containing PLCs, IoT gateways, and MQTT brokers. Results show a linear time complexity for all the broker implementations and bridging modes. The access token encapsulation techniques used in the prototype demonstrate a minimal overhead compared to standard MQTT brokers.

6.1.4. OAI

The OpenAirInterface (OAI) [132, 133] initiative encompasses several projects aimed at developing open-source solutions for wireless communication networks. These projects focus on various aspects of wireless technology, enabling researchers and developers to create and experiment with advanced wireless systems. Some of the key projects included in the OAI Initiative are the OAI 5G Radio Access Network (5G RAN), the OAI 5G Core Network (5G CN), and the MOSAIC5G (M5G). The 5G RAN centers on the implementation of 5G radio access network functionalities, including base stations and user equipment models. The 5G CN focuses on developing open-source core network components, such as the Evolved Packet Core (EPC) for 4G and Next-Generation Core (NGC) for 5G. The M5G project group aims to transform radio access (RAN) and core networks (CN) into agile and open network-service delivery platforms.

OAI utilizes a software radio frontend architecture hosted on a personal computer. It provides a comprehensive and open-source implementation of LTE and LTE-Advanced features, fully compliant with 3GPP standards. OAI covers both essential components of the LTE system architecture: the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and the Evolved Packet Core (EPC). The platform spans the entire protocol stack, from the physical to the networking layer, and offers realistic emulation modes. This enables diverse applications, including indoor/outdoor field experimentation and controlled/scalable evaluations with emulated wireless links. Leveraging the success of the OpenAirInterface initiative, which boasts more than 3000 members, the OAI platform also serves as an open-source facility for remote experimentation with LTE technology.

6.1.5. srsRAN

The srsRAN project [134] is an open-source initiative that focuses on creating a software-defined radio (SDR) solution for Radio Access Networks (RAN). The project aims to develop a flexible and customizable RAN software stack that is compatible with various wireless communication standards. srsRAN provides a complete and standards-compliant implementation of various RAN technologies, including 4G LTE, and 5G. It allows researchers, developers, and operators to experiment

with and deploy RAN solutions using off-the-shelf hardware and software-defined radio platforms. The project emphasizes performance optimization and scalability, making it suitable for diverse deployment scenarios, ranging from small-scale private networks to large-scale public networks.

6.1.6. openLEON

OpenLEON [135, 136] is an open-source software project that focuses on developing an emulation framework for wireless networks, particularly targeting 4G LTE and 5G technologies. The project aims to provide researchers, developers, and practitioners with a flexible and extensible platform to study and evaluate the performance of wireless communication systems. OpenLEON offers a comprehensive emulation environment that encompasses various network components, including base stations, user equipment, and core network elements. It allows users to customize network parameters, mobility models, and network topologies, enabling them to conduct detailed and realistic emulations. openLEON combines the capabilities of current emulators designed for data centers and mobile networks, such as Containernet and srsLTE. By doing so, it enables the assessment and verification of research concepts concerning all the elements of a complete mobile edge architecture in an integrated manner.

6.1.7. Comparison, Discussion, and Limitations

Mininet is an open-source network emulator that creates virtual networks on a single machine. It is widely used for testing and developing SDN (Software-Defined Networking) applications. Mininet allows users to emulate networks with multiple hosts, switches, and controllers. Mininet's strength lies in its simplicity and ability to create lightweight network topologies for SDN experimentation. However, Mininet is limited to emulating networks on a single machine, which may restrict scalability and performance for larger simulations. Heller [1] studied Mininet's limitations and proposed Mininet-HiFi to address them. This approach tracks the system's invariants, namely, network invariants and host invariants. Network invariant consists of queueing delay, transmission delay, propagation delay, forwarding delay, and transmission gap, whereas host invariants comprise preempt-to-schedule latency and CPU capacity. Mininet-HiFi tracks these invariants and compensates them for providing a realistic environment.

Containernet is an extension of Mininet that adds support for Docker containers in network emulation. Moreover, Containernet supports adding or removing Docker containers from the emulated network at runtime [137]. This feature is not available in Mininet and allows the reproduction of a cloud infrastructure where the administrator can start or stop instances at any time. It allows users to integrate virtualized applications within network topologies and simulate more realistic SDN environments. Containernet improves upon Mininet by enabling the incorporation of Docker containers, offering a more comprehensive emulation environment for SDN experiments. Similar to Mininet, Containernet may also face limitations in terms of scalability and performance when dealing with larger, more complex simulations.

NEMO is a Java-based network simulator focused on mobility scenarios. It allows researchers to simulate and evaluate mobile networks, including scenarios involving moving nodes and handovers between different network access points. Thank you for pointing that out. NEMO stands out for its specialization in mobility simulations, making it suitable for research in the field of mobile communication. However, NEMO's capabilities might be limited when compared to more versatile network simulators that cover a wider range of scenarios.

OAI is an open-source network simulator designed for 4G and 5G wireless communication systems. It provides a flexible platform for testing and developing radio access network protocols and technologies. OAI focuses on 4G and 5G wireless communication, providing researchers with a specialized tool for these networks. As a specialized simulator, OAI may not be suitable for simulating other types of networks or technologies.

srsRAN is an open-source software suite that emulates radio access networks for 4G and 5G systems. It focuses on radio aspects and is commonly used for testing and research in radio communications. srsRAN offers researchers the ability to evaluate radio-specific scenarios and technologies. Similar to other specialized simulators, srsRAN may have limitations when it comes to simulating other network aspects beyond radio communications.

openLEON primarily focuses on satellite communication networks. It provides a platform for evaluating satellite communication protocols and scenarios. As a specialized simulator, openLEON may not be suitable for simulating other types of networks unrelated to satellite communication.

6.2. Virtualization-based Emulation

This section describes the platforms that integrate different virtualization technologies to build a test environment where the user can run experiments. These platforms aim to provide a consistent and reproducible environment that enables hands-on learning of computer network concepts, cybersecurity principles, and troubleshooting procedures. The approaches described in this section rely on VMs to implement a testing scenario. One key advantage of using VMs to create network testbeds is their ability to provide a high degree of isolation and security. VMs are completely isolated from the host operating system and from other VMs, providing a secure environment for testing and evaluating network protocols and applications. This makes it easier to test the impact of different network configurations and protocols on the performance and behavior of applications without compromising the security of the host system. VMs also provide greater control over the network environment, allowing developers to modify and customize any aspect of the network stack, including hardware, operating systems, and network protocols. This enables them to test the impact of different network configurations and protocols on the performance and behavior of their applications.

6.2.1. CORE

Ahrenholz *et al.* [50] presented the Common Open Research Emulator (CORE), a real-time network emulator focused on the rapid instantiation of hybrid topolo-

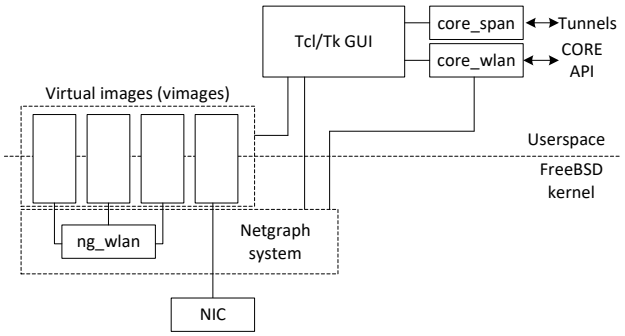

Figure 10: CORE architecture [50].

gies. CORE facilitates integrating virtualized instances with real hardware using the FreeBSD network stack to extend physical networks for planning, testing, and development. The platform aims to reduce the need for expensive hardware to run network experiments. The authors highlight the scalability of CORE in instantiating over a hundred virtual emulated nodes. These nodes can run on a regular server achieving a performance that involves sending and receiving over 300,000 packets per second (i.e., ∼4Gbps). Live-running emulations with CORE can be connected to physical networks and routers in real-time. Figure 10 shows CORE's architecture. CORE relies on the Integrated Multi-protocol Network Emulator/ Simulator (IMUNES), open-source software that allows multiple lightweight virtual instances to inter-operate. These virtual instances are via FreeBSD's Netgraph kernel subsystems. The user can interact with the emulator via a Tcl/Tk-based GUI that allows the rapid development of X11 user interfaces. In recent years, CORE included Python APIs as they are in Mininet. These APIs facilitate complete control over all aspects of the emulations, thus creating a rich programming environment.

Joy *et al.* [138] validated a network coding scheme in a scenario where the network connectivity fluctuates intermittently. They implemented their scheme using the CORE emulator and measured the file delivery ratio, latency, and network overhead. The authors found that although Transmission Control Protocol (TCP) provides reliable block transfer, it cannot compete with the broadcast capability obtained using User Datagram Protocol (UDP). Lunardi *et al.* [139] implemented an appendable-block blockchain framework to support different consensus algorithms through modular design. The authors used CORE to evaluate the performance and study the impact of modifying the number of devices and transactions as a function of the consensus algorithm. Results indicated that the latency to append a new block is less than 161 milliseconds under heavy load, whereas the delay for processing a new transaction is less than seven milliseconds. Their findings suggest that their approach is an efficient and scalable solution for IoT devices. Tomsett and Bent [140] used the CORE emulator to demonstrate how Vector Symbolic Architectures (VSA) can be employed for distributed semantic discovery and orchestration of remote devices. The authors tested the system with different workflows and considered that the participating nodes did not know the IP location of the adjacent devices. Results show that the system performs well without a central control point with multiple coordinated sub-workflows.

6.2.2. Cisco CML

Cisco Modeling Labs (CML) [51], previously known as Cisco Virtual Internet Routing Lab (VIRL), is an emulation platform that enables operators, engineers, network designers, and architects to design Cisco-based networks using virtual versions of Cisco operating systems. CML includes a server and a client, providing a sandbox environment that quickly and efficiently facilitates the design, configuration, visualization, and simulation of network topologies rapidly and efficiently. The platform reproduces testing scenarios by orchestrating VMs, representing the devices as nodes and the connection between them as links [141]. These instances are translated into commands that instantiate the network VMs and the links transparently, providing the experimenter with a modeling framework to design and build network experiments. The main components of CML are the server, the client, and the virtual images. The CML server is a shared resource containing mechanisms to instantiate topologies using virtual images. On the other hand, the CML client provides a GUI that simplifies topology creation and initial device configuration in the CML server. The CML server is a Linux distribution that runs on VMware Open Virtual Appliance (OVA) files. This shared resource run backend functions such as router bootstrap configuration, spinning up routers to operate with designated operating systems, and modifying and testing configuration. The CML server framework comprises three main components: OpenStack [142], AutoNetkit [143], and a services topology director. OpenStack is a cloud computing configuration manager used to coordinate the operations of a large set of VMs. AutoNetkit is a tool to generate the configuration files used to link the VMs in a CML-based topology. The services topology director produces OpenStack calls for the generation of VMs and links based on the XML topology defined in the CML client.

Zorello *et al.* [144] employed CML to design and implement a Hybrid SDN-based network application to provide dynamic services that combine centralized and distributed control with traditional Virtual Private Network (VPN) protocols. The system performs a flexible policy-based routing by selecting the access technology according to the QoS requirements and network conditions. Results show that the proposed approach enables the services to be efficiently supplied without over-provisioning the resources. Al-Musawi *et al.* [145] described the operations of a Border Gateway Protocol (BGP) time stamp replay tool that enables operators and researchers to improve security issues. The tool performs functions by replaying past BGP events into a controlled testbed. BGP is the most used inter-domain routing protocol used on the Internet. It manages network reachability information between Autonomous Systems (ASes) with guarantees of avoiding routing loops. The authors evaluated the system's functionalities with CML to emulate the behavior of Cisco routers. They conducted experiments by injecting a stream of BGP updates into the network. This generated data set represents a series of announcements and withdrawals of IPv4 and IPv6 prefixes for 100 seconds. Results show that the tool facilitates understanding BGP behavior at the BGP speaker level, classifying

BGP updates, and debugging BGP behavior in different routers' images.

6.2.3. IMUNES

Puljiz and Mikuc [52] presented the Integrated Multiprotocol Network Emulator/Simulator (IMUNES), a distributed network emulator based on lightweight virtualization. IMUNEs works on a modified FreeBSD [146] kernel distribution to enable emulated nodes using UNIX applications. IMUNES provides scalability, realistic performance, and high fidelity. The tool achieves high scalability by using a lightweight VMs model that does not drain the resources of the hosting machine. Scalability occurs by handling packets from one VM to another by passing packet pointers. IMUNES keeps all VMs residing in the kernel to provide high fidelity. This feature is based on real system calls processed in only one context switching, which creates a user process for the kernel. IMUNES provides a management utility responsible for mapping GUI-level objects to kernel-level objects. The kernel-level objects used for emulation are VMs and netgraph nodes. IMUNES provides a GUI to create network topologies consisting of two basic units: nodes and links. Nodes can exist independently, whereas links always connect two different nodes.

Kuman *et al.* [147] used IMMUNES to create an Industrial Control System (ICS) testbed. The authors created a honeynet consisting of honeypots that emulate a network of devices instead of a single device. Results show that emulated topologies give owners of ICSs more options for securing their systems. Along with potentially improving the security of an ICS, the author highlights that the tool could serve as a template to build other kinds of honeynets. Salopek *et al.* [148] conducted a comparative analysis of emulation testbeds with IMUNES. The authors measured key performance metrics to determine the efficiency of the surveyed tools. The scenarios considered for the evaluation included IPsec peers and IPv6 to IPv4 translation. Their evaluation showed that IMUNES offers excellent flexibility for adapting to the precise requirements of various test scenarios beyond its limitations.

6.2.4. SEED Labs

The SEcurity EDucation (SEED) labs [53] is a collection of more than 30 hands-on exercises covering various cybersecurity topics. These exercises are distributed as prebuilt VMs images that learners download and run on their computers, facilitating the adoption and distribution of the content. The lab environment leverage full virtualization to run Ubuntu Linux and Minix [149] operating systems. Most labs use Ubuntu Linux, and a smaller set uses Minix to perform kernel-level coding. The SEED labs do not require a dedicated computer laboratory. Instead, learners can run them using their laptops or renting computing resources in the cloud. Learners can download the VM images related to software security, network security, web security, system security, cryptography, blockchain, and mobile security. The labs also provide instructor material that academic institutions can adopt to instruct cybersecurity courses.

6.2.5. Comparison, Discussion, and Limitations

CORE is an open-source network emulator that allows users to create complex network topologies for testing and research. It provides a user-friendly interface and supports a variety of network devices and protocols. CORE is praised for its ease of use and versatility, making it suitable for a wide range of network emulation scenarios. However, CORE may have limitations in terms of scalability for very large network simulations and may not offer as extensive device support as some commercial emulators. Ahrenholz [150] evaluated the performance of the CORE emulator in various scenarios. The author reported that the emulator fairly utilizes the host's resources compared to the Netgraph and Linux Ethernet bridging.

Cisco CML is a commercial network emulator that provides a robust environment for simulating Cisco network devices and complex network topologies. It is suitable for professional network testing and certification preparation. Cisco CML's strength lies in its extensive support for Cisco devices and its accuracy in representing real-world network behaviors. Tyler and Viana [151] used Cisco CML to create a framework for assisting healthcare organizations in transitioning to a zero-trust network architecture. The authors found a zero-trust architecture could secure medical devices by placing firewalls directly in front of them. Although the approach increases latency, it blocks all unnecessary traffic on the rest of the network resulting in a performance boost. On the other hand, the authors reported that Cisco CML does not support Multi-Factor Authentication (MFA), which is an essential feature for reproducing a zero-trust architecture.

IMUNES is an open-source network emulator designed for testing and evaluating communication networks. It offers a visual interface and supports various network protocols. IMUNES' strength lies in the ease of visually representing network topologies, which aids in understanding and debugging complex configurations. However, IMUNES may not have as large a user community or extensive documentation compared to other well-established emulators. Zec and Mikuc [152] analyzed the performance of the IMUNES emulator in several scenarios. The authors found that the achievable throughput in IMUNES depends on the number of emulated hops packets traverse. Additionally, they observed another scaling issue related to the size of the network topology that IMUNES can support on a physical node. This size depends on the number of entries in the routing table and the user space application running on each node.

SEED Labs is an educational platform that provides a series of hands-on labs for learning about computer security and networking. While not a dedicated emulator, it offers network simulation exercises to enhance practical understanding. SEED Labs' strength lies in its educational focus, providing learners with interactive, hands-on experiences in network-related concepts. The limitation of SEED Labs is that it is primarily intended for educational purposes and may not offer the same level of features and complexity as other dedicated network emulators.

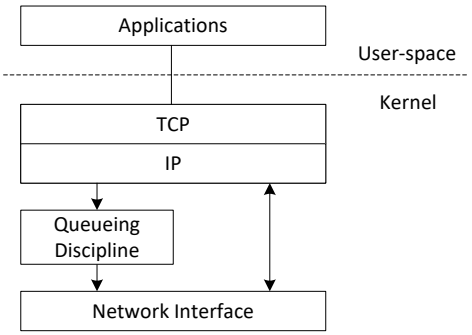

Figure 11: Linux queueing discipline. NetEm is implemented in Linux distributions as a queueing discipline [54].

6.3. Network Link Emulation

This section explains network link emulators, the simpler tools to add delay, packet drops, and other disturbances into a network link. Network link emulators operate in the network interface to alter the behavior of incoming and outgoing packets.

6.3.1. NetEm

Hemminger [54] developed Network Emulator (NetEm), one of the most widely adopted link emulators used for testing the performance of applications over virtual networks. NetEm can reproduce long-distance WANs in environments that include VMs, Docker containers, and bare-metal servers. Scenarios created with NetEm facilitate testing and evaluating protocols and devices from the application layer to the data-link layer under various conditions. NetEm enables modifying parameters such as delay, jitter, packet loss, duplication, and re-ordering of packets. It consists of two portions: a kernel module that implements queuing disciplines and a command line utility to enable interaction with the experimenter. Figure 11 shows the basic architecture of NetEm. NetEm is implemented as a Linux traffic control queueing discipline. The Linux traffic control subsystem provides the methods to control the transmission of packets in the Linux kernel. The queuing disciplines reside between the IP protocol output and the network device. A queuing discipline is a simple object composed of two interfaces. One interface queues outgoing packets, and the other interface releases incoming packets. The default queuing discipline is the FIFO queue which can be modified using the Linux traffic control commands. The queuing discipline implements policies that decide which packets to send, delay, and drop.

Kfoury *et al.* [153] conducted an emulation based on the Bottleneck Bandwidth and Round-trip Time version 2 (BBRv2) congestion control. A main component of TCP is its congestion control mechanism, which dictates how a sender will inject packets into the network and react when congestion is experienced. To reproduce wired broadband scenarios, the authors employed NetEm to emulate latency and packet loss over emulated links. Results show that BBRv2 has better coexistence when interacting with CUBIC than its predecessor, BBRv1. Gomez *et al.* [154] evaluated the performance of the BBRv2 congestion control. The authors used NetEm to create Round-Trip Time (RTT) unfairness between competing flows, explore the accumulative effects of BBRv2 flows joining a current flow, and analyze the behavior of BBRv2 under changing network conditions. The authors report that BBRv2 presents less disruptive behavior than BBRv1 when interacting with CUBIC flows. Lübke *et al.* [155] evaluated the accuracy and performance of NetEm in scenarios including hardware and software solutions. Experiments conducted using different packet sizes reported that NetEm is the network emulator that presents smaller deviations from the preset configuration in terms of bandwidth, delay, and packet losses.

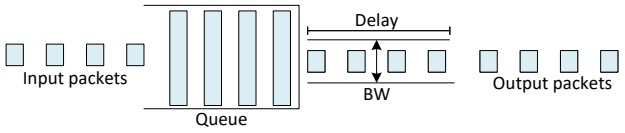

Figure 12: The structure of a Dummynet pipe. The bandwidth, delay, and queue size are configurable parameters [156].

6.3.2. Dummynet

Dummynet [55] is a link emulator developed to run experiments in user-configurable network environments. It supports emulating moderately complex network setups on unmodified operating systems to provide reliable and reproducible results. The main factors contributing to the popularity of Dummynet in the research community are availability, learning curve, and feature set [156]. Most operating systems support Dummynet. Hence, researchers could find it readily available to run their experiments. Moreover, there are bootable disk images to create Dummynet-enabled bridges using existing PC hardware without disrupting existing software installations. The user interface of Dummynet is based on a set command that facilitates the creation of network topologies in a few lines. Since its first version, Dummynet has been significantly improved to support flexible packet classifiers, queue management schemes (FIFO, RED, GRED), and the ability to create per-flow emulated links. Dummynet can emulate complex topologies and reproduce multipath effects. The authors also added support for dynamically loadable packet schedulers, including Deficit Round-Robin (DRR), Weighted Fair Queueing Plus (WF2Q+), and other queueing disciplines. The emulator also supports third-party extensions to introduce programmable or trace-driven packet dropping or alterations.

Dummynet comprises several components: pipes, packet classifiers, multipath and multihop networks, packet dropping policies, queue management algorithms, packet scheduling agents, and media access control (MAC) intermediate layer effects. Figure 12 shows the basic structure of a pipe in Dummynet. This element combines a finite-size queue, a fixed bandwidth communication link, and a programmable propagation delay. A numeric identifier defines each pipe and is constrained by available memory. The user interacts with pipes via a CLI to manage the sending rate, delay, and available bandwidth. A packet traversing a pipe is enqueued into a buffer. This buffer is then drained at a rate corresponding to the link's bandwidth configured by the user.

Szilágyi and Bordán [157] employed Dummynet to test the performance of a Generic Routing Encapsulation

(GRE) tunnel over a Wide Area Network (WAN). Measurement results showed that the solution could efficiently aggregate the performance of physical connections in an emulated WAN environment. The authors evaluated the effects of different network parameters, such as packet delay, jitter, and loss rate, on data transfers and CPU utilization. The emulation confirmed that the worst performance is experienced with a 1% of packet losses. Noda and Ito [158] employed Dummyet to reproduce scenarios for testing the behavior of Multi-Path TCP (MPTCP). The authors employed the emulator to create various environments for testing the Quality of Service (QoS) for a web service that uses MPTCP. They also proposed an improvement to the scheduler to improve the trade-off between the QoS fluctuation and the throughput. Results show that the proposed scheduler suppresses the variance of RTT manifested by the default scheduler in all testing scenarios. Al-Saadi and Armitage [159] presented an experimentation framework that used Dummynet to evaluate Active Queue Management (AQM) algorithms. The authors implemented in Dummynet running on a FreeBSD [146] distribution the AQMs Controlled Delay (CoDel), Proportional Integral controller Enhanced (PIE), and Fair-Queueing CoDel (FQ-CoDel). The authors experimentally compared their implementation against the ones implemented in other Linux distributions. Results show that their implementation behaves similarly to equivalent Linux kernel implementation. They also demonstrated that their proposed variation of PIE (i.e., FQ-PIE) is a promising alternative to FQ-CoDel.

6.3.3. Mahimahi

Mahimahi [56] is a network emulation tool that enables researchers and developers to recreate different network conditions in order to test and evaluate the performance of their systems and applications. Mahimahi is designed to be highly flexible, user-friendly, and effective, with a wide range of features that can accommodate the needs of a diverse set of users. Mahimahi works by leveraging the Linux traffic control (tc) subsystem to simulate various network conditions, such as latency (DelayShell), packet loss (LossShell), and bandwidth restrictions (LinkShell). It supports a wide range of network topologies, including single-link, star, and tree topologies, and can be configured to simulate different types of networks, such as wired or wireless networks, 3G or 4G mobile networks, and satellite or long-haul networks. The tool provides users with a command-line interface that is easy to understand and offers a wide range of options for customizing network conditions and topologies. Users can specify the amount of latency, packet loss, or bandwidth to introduce, as well as the number of network devices and links to include in their simulations. They can also configure different types of traffic shaping and control techniques, such as token bucket filtering, RED (Random Early Detection), and traffic scheduling.

In addition to its network emulation capabilities, Mahimahi also includes advanced features for traffic capture and replay, automated configuration of network conditions based on user input, and integration with other tools like Wireshark and Chrome DevTools for network analysis and debugging. This makes it a powerful and versatile tool for evaluating the behavior of networked systems and applications under different network conditions. Mahimahi is a highly effective and flexible network emulation tool that is widely used in research and development to evaluate the performance of networked systems and applications. Its ease of use, configurability, and support for a wide range of network topologies and conditions make it a popular choice for researchers and developers who need to simulate different network scenarios and evaluate the behavior of their systems or applications under these conditions.

Netravali [160] presented a comprehensive study of web page loading times on modern networks, focusing on the impact of different factors such as network characteristics, web page design, and user behavior. The author performed experiments using real-world networks, including home and mobile networks, and analyze the performance of a large set of web pages from popular websites. The findings show that the performance of web page loading is highly variable, with significant differences in loading times between different web pages and between different networks. The study identified several factors that contribute to this variability, including the size and complexity of the web page, the number of requests and connections required, and the latency and bandwidth of the network. Additionally, the author found that user behavior, such as scrolling and clicking on links, can affect the perceived loading time of the web page. To improve web page loading times, the author proposed several optimizations that can be applied to web page design and network protocols. The first recommendation is minimizing the number of requests and connections required, reducing the size of web pages, and optimizing the use of caching and compression. The study also suggested improvements to network protocols such as TCP and HTTP that can reduce latency and improve bandwidth utilization. The author validated their proposed optimizations through experiments and simulations, demonstrating that they can significantly improve web page loading times on real-world networks. The study concluded that a combination of optimizations to web page design and network protocols is necessary to improve web page loading times and provide a better user experience.

Zhang *et al.* [161] used the Mahimahi emulator to evaluate the performance of low-latency HTTP-based streaming players, focusing on the impact of different design choices on latency and player stability. The authors analyzed the behavior of several popular streaming players, including the Apple HTTPS Live Streaming (HLS), Adobe HTTP Dynamic Streaming (HDS), and Moving Picture Experts Group Dynamic Adaptive Streaming over HTTP (MPEG-DASH) protocols, and study the performance of these players under different network conditions. The authors found that low-latency streaming players can achieve latencies as low as a few seconds, but the latency is highly dependent on the design choices made by the player. They identified several design choices that can affect the latency, including the size of the video chunks, the number of chunks preloaded by the player, and the way the player handles rebuffering events. The authors also investigated the stability of low-latency streaming players under adverse network conditions, such as network congestion and packet loss.

They found that low-latency players are more prone to stability issues than traditional players due to their reliance on small video chunks and frequent requests for new content. To improve the performance and stability of low-latency streaming players, the authors proposed several optimizations that can be applied to player design and network protocols. They suggested increasing the size of video chunks to reduce the number of requests required, optimizing the use of caching and preloading, and improving error handling and recovery mechanisms. The authors validated their proposed optimizations through experiments and simulations, demonstrating that they can significantly improve the performance and stability of low-latency streaming players on real-world networks. They concluded that a combination of optimizations to player design and network protocols is necessary to achieve low-latency streaming while maintaining player stability and providing a high-quality user experience.

6.3.4. Comparison, Discussion, and Limitations

NetEm is a network emulator built into the Linux kernel. It allows users to introduce delay, loss, and other network conditions to emulate real-world network scenarios. NetEm is commonly used for testing and evaluating network applications under different network conditions. NetEm's strength lies in its integration with the Linux kernel, providing a straightforward way to introduce network impairments for testing purposes. However, NetEm may have limitations in terms of fine-grained control over complex network conditions, and its configuration can be less intuitive for users without Linux expertise. Moe [162] addressed a limitation of NetEm, which is its inability to emulate network bandwidth. NetEm does not have a built-in bandwidth emulator. Thus, it relies on other Linux traffic control queueing disciplines, such as TBF, to limit the bandwidth. The author observed that configuring NetEm with other Linux traffic control can present non-deterministic results. Therefore, the proposed system extended NetEm capabilities to emulate rate and delay with high accuracy.

Dummynet is a network emulator primarily used on FreeBSD and other BSD-based systems. It is designed to control and limit the bandwidth of network connections, enabling researchers to simulate bandwidth-constrained environments. Dummynet's strength lies in its ability to accurately limit bandwidth, making it suitable for testing applications under constrained network conditions. However, Dummynet may have limitations when it comes to emulating other network impairments like delay and loss, which are important factors in many real-world network scenarios. Lübke *et al.* [155] compared the performance of Dummynet against other Link emulators, including NetEm. The author found that NetEm experience packet reordering at high rates, whereas Dummynet presents inaccurate bandwidth and delay values in the same scenario.

Mahimahi is a network emulator developed by Stanford University. It focuses on emulating network conditions with fine-grained control over delay, loss, and bandwidth. Mahimahi is commonly used for research in networking and particularly in web applications. Mahimahi stands out for its precise control over network impairments, providing researchers with a flexible tool for studying the effects of various network conditions. One limitation of Mahimahi is its specialized focus on network conditions and may not have the same level of device support or versatility as other general-purpose emulators.

6.4. Cloud-based Emulation

A cloud-based emulator is a virtual environment hosted in the cloud that allows users to emulate and test various network conditions, such as network latency, bandwidth limitations, and packet loss, in a controlled and scalable manner. Cloud-based emulators typically use virtual machines or containers to create these environments and provide users with the ability to configure and control various network parameters, as well as capture and analyze network traffic.

Cloud-based emulation provides several key advantages for testing and developing networked systems. One key motivation is the flexibility and scalability that cloud-based emulators offer. With cloud-based emulation, users can easily spin up and tear down virtual environments as needed, without the need for expensive hardware or complex infrastructure. Cloud-based emulation also provides a cost-effective solution for testing and developing networked systems. With cloud-based emulators, users can pay for the resources they need on a pay-as-you-go basis, without the need for expensive hardware or complex infrastructure. This allows experimenters to save on costs while still providing a reliable and scalable testing environment. Cloud-based emulation can enable collaboration and sharing among geographically dispersed teams. With cloud-based emulators, users can easily share testing environments and collaborate on development projects, regardless of their physical location, leading to more efficient and effective development processes.

6.4.1. Netlab Academic Cloud

The Netlab academic cloud is a distributed platform that provides computing resources to support virtual laboratories. These computing resources are distributed across four data centers in the United States. The platform dynamically provides the resources that maximize the learner's experience in running a virtual laboratory. The University of South Carolina, in cooperation with the Stanly Community College and the Network Development Group (NDG) [163], deployed the academic cloud platform in January 2020. As of January 2022, it has served over 100,000 learners [57]. Figure 13 illustrates the academic cloud architecture. Servers are provisioned with a large number of resources: CPUs (32 to 40 cores per server), RAM (∼1TB per server), and storage (∼3.5TB per server). On average, each data center has approximately ten servers used to host virtual pods. A pod is a collection of resources such as VMs, virtual switches, virtual links, physical Tofino-based switches, and others) orchestrated by the academic cloud platform to deliver virtual laboratory experiments. Some virtual laboratories require more than one VMs to set up an experiment. As Netlab+, the academic cloud is implemented with the server virtualization software VMware vSphere [164]. The vSphere components include a collection of bare-metal

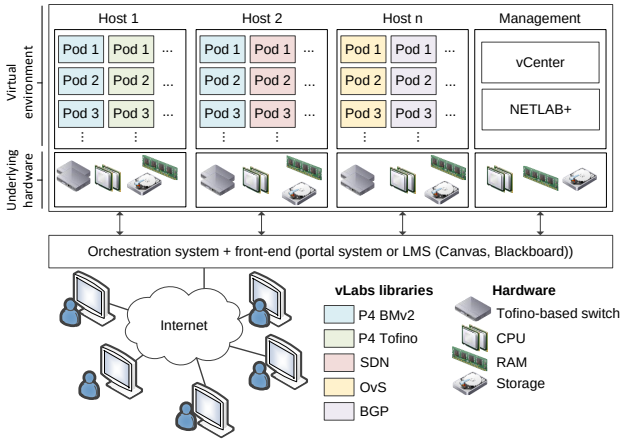

Figure 13: The academic cloud architecture [165].

hypervisors (ESXi), a management server (vCenter), and a VM running NETLAB, a customized management application developed by NDG. The academic cloud functions include aggregating resources from the four data centers, implementing a calendar interface for scheduled access to equipment pods, and routing pod reservations to the nearest data center containing the requested pod type.

6.4.2. AWS

Amazon Web Services (AWS) [58] provides a comprehensive set of cloud-based services that can be leveraged as a network testbed. These features are scalability, cost-effectiveness, flexibility, security, and accessibility. AWS offers virtually unlimited scalability, allowing users to simulate a wide range of network scenarios without the need for expensive hardware investments. This makes it an ideal platform for testing large-scale network deployments. AWS is a pay-per-use service where users only pay for the resources they consume, making it an economical option for network testing. Additionally, the ability to spin up and tear down resources on demand further reduces costs. Moreover, AWS offers a wide range of services and tools that can be used for network testing, including Virtual Private Cloud (VPC), Elastic Load Balancing (ELB), and Amazon Elastic Compute Cloud (EC2). This flexibility provides a wide range of experiments. AWS is a secure cloud platform that complies with various industry standards and regulations. This allows users to test network security in a safe and controlled environment. Finally, since AWS is a cloud-based service, it can be accessed from anywhere with an internet connection, making it an ideal platform for collaborative network testing.

Gomez-Sanches *et al* [166] used AWS Elastic Compute Cloud (EC2) as a testbed infrastructure for High-Performance Computing (HPC) applications. The authors conducted experiments on AWS EC2 instances to test different I/O configurations for HPC applications, specifically focusing on the impact of buffer sizes and network bandwidth on application performance. The results showed that AWS EC2 can be used as a reliable and cost-effective testbed for HPC application I/O system configuration, with the ability to simulate a range of different network bandwidths and buffer sizes. The paper provides insights into the use of cloud-based infrastructure for HPC testing and highlights the potential benefits of using AWS EC2 as a testbed for I/O system configuration testing. Ruth and Cevik [167] presented an experimental study on the use of AWS Direct Connect (DX) with Chameleon, ExoGENI, and Internet2 Cloud Connect. The authors evaluated the performance of AWS DX for data transfer and communication with different cloud and research infrastructures. They found that using AWS DX with Chameleon and ExoGENI resulted in improved network performance and reduced data transfer time, compared to using the public internet. Additionally, the study showed that using Internet2 Cloud Connect with AWS DX can provide even better performance for specific use cases, such as transferring large amounts of data. The authors highlight the potential benefits of using AWS DX in combination with other cloud and research infrastructures and provide insights into how AWS DX can be used to optimize network performance for data-intensive applications. Jackson *et al.* [168] conducted a study on the performance of HPC applications on the AWS cloud. The authors evaluated the performance of different HPC applications on AWS EC2 instances and compared the results with those obtained from traditional HPC clusters. They found that AWS EC2 instances can provide comparable performance to traditional HPC clusters for certain applications, but may be less efficient for others due to differences in hardware and network architectures. The study also highlighted the importance of optimizing application parameters and selecting appropriate instance types to achieve optimal performance on AWS EC2. The authors provided insights into the performance characteristics of HPC applications on AWS EC2 and the factors that can affect performance, which can be useful for researchers and practitioners considering cloud-based HPC solutions.

6.4.3. GCE

Google Compute Engine (GCE) [59] is a cloud-based infrastructure that provides virtual machines on demand for a wide range of computing workloads. GCE can be an attractive option for researchers and practitioners as a testbed infrastructure for several reasons. GCE offers a highly flexible and customizable environment that can be configured to meet specific testing requirements. GCE provides access to a large number of virtual machine types, ranging from small to very large, with a variety of processing, memory, and storage capabilities. GCE provides a robust and reliable infrastructure with high availability and scalability, which can be crucial for large-scale testing scenarios. GCE offers a pay-per-use pricing model, allowing users to optimize their testing costs based on the specific needs of their experiments. Finally, GCE integrates well with other Google Cloud Platform services, such as storage, networking, and data analysis tools, which can provide a comprehensive testing environment. GCE can be an attractive option as a testbed infrastructure for researchers and practitioners looking for a flexible, scalable, and cost-effective platform to conduct experiments and test their applications.

Ruan *et al.* [169] used GCE to conduct a study on the performance of containers in a cloud computing environment. The authors compare the performance of containers to that of traditional virtual machines (VMs)

in terms of resource utilization, overhead, and scalability. The study was conducted using the Docker container platform and a popular VM platform, KVM, on a cloud infrastructure based on OpenStack. The authors found that containers have lower overhead and better resource utilization compared to traditional VMs. Specifically, the study showed that containers have lower CPU overhead, memory overhead, and disk space overhead compared to VMs. Additionally, the study found that containers are more scalable and have faster startup times compared to VMs. However, the study also highlighted some limitations of containers, such as the lack of isolation between containers running on the same host, and the potential security risks associated with shared kernel environments. The paper provides insights into the performance characteristics of containers in a cloud computing environment and highlights the potential benefits and limitations of using containers for cloud-based applications. Kratzke and Quint [170] investigated the impact of microservice architecture on network performance using GCE. The authors sustain that microservice architecture has become increasingly popular for designing cloud-based applications, but the impact of this architecture on network performance is not well understood. The study was conducted by implementing a microservice-based system using Docker containers and measuring the network performance of the system. The authors compared the network performance of the microservice-based system to that of a monolithic system with similar functionality. The study found that the microservice-based system had higher network latency and lower network throughput compared to the monolithic system. The authors attribute this to the overhead of inter-service communication in the microservice architecture, as well as the additional network hops required for service discovery and load balancing. However, the study also found that the microservice-based system had better fault tolerance and scalability compared to the monolithic system. Overall, the paper provides insights into the trade-offs between network performance and other aspects of system design when using microservice architecture, and highlights the importance of carefully considering the impact of architecture choices on network performance.

6.4.4. Microsoft Azure

Microsoft Azure [60] is a cloud computing platform that offers a wide range of services and tools for building, deploying, and managing applications and services in the cloud. It provides a flexible and scalable environment for organizations to meet their computing needs, from small-scale applications to large-scale enterprise solutions. Azure offers a variety of services for computing, storage, analytics, and networking, all of which can be accessed and managed through a centralized dashboard. Its compute services include VMs, container instances, and serverless computing options like Azure Functions and Azure Logic Apps. Azure's storage options include block, file, and object storage, as well as backup and disaster recovery solutions. Azure also offers a variety of data and analytics services, including SQL databases, NoSQL databases, and Big Data processing solutions. Additionally, it provides networking services such as load balancing, virtual private networks (VPNs), and domain name system (DNS) hosting. One of the key benefits of Azure is its support for a wide range of programming languages and frameworks, including .NET, Java, Python, and Node.js, among others. This allows developers to build and deploy applications using the tools and languages they are most comfortable with. Another benefit of Azure is its support for hybrid cloud scenarios, allowing users to integrate their on-premises resources with their Azure resources. This makes it possible to build and deploy applications across multiple environments, while still maintaining a centralized management system. Azure is known for its reliability, scalability, and security features. It provides a 99.95% uptime Service Level Agreement (SLA) for its compute services and offers various options for disaster recovery and backup. Additionally, Azure's security features include network security groups, firewalls, and identity and access management solutions. Azure's pricing model is based on a pay-as-you-go model, which allows users to only pay for the resources they use. Azure also provides various support options, including self-service support, developer support, and enterprise support. Microsoft Azure is a comprehensive and flexible cloud computing platform that provides a wide range of services and tools for building, deploying, and managing applications and services in the cloud. It is known for its reliability, scalability, and security features, and is used by a wide range of organizations, from small startups to large enterprises.

Persico *et al.* [171] investigated the variability of network throughput in the Microsoft Azure cloud environment. The authors conducted experiments using a variety of network measurement tools and different Azure regions and analyze the resulting data to understand the sources and patterns of network throughput variability. The study found that network throughput in Azure exhibits significant variability, both over time and across different Azure regions. The authors attribute this variability to a number of factors, including network congestion, server load, and inter-VM interference. They also found that the source and pattern of variability depended on the measurement tool used and the specific Azure region being tested. The study concluded that network throughput variability is an important consideration for users of Azure, particularly those who require consistent and predictable network performance. The authors suggest that users should carefully monitor and manage their network resources in order to mitigate the effects of variability and that Microsoft should continue to improve its network management and monitoring tools in order to provide better visibility and control over network performance. Hassan *et al.* [172] explored the performance of HPC applications on the Microsoft Azure cloud platform. The authors conducted experiments to evaluate the scalability and communication performance of HPC applications on Azure, using a variety of benchmark applications and metrics. The study found that Azure can support HPC applications with good scalability and communication performance, particularly for embarrassingly parallel workloads. The authors attribute this to Azure's use of InfiniBand networking technology, which provides low-latency and high-bandwidth communication between compute nodes. However, the study also found that the performance of HPC applications on Azure is highly de-

Table 4: Emulators comparison.

Ref.	Enulator	Description	Nodes	Purpose	Resource consumption	Programmable data planes	Integrates w/ hardware	Cost
[47]	Mininet	Mininet is a container-based emulator used to run a wide variety of experiments. It provides realism and scalability needed to test protocol, systems, and reproduce attack-defense scenarios.	< 1000	General	Medium	Supported	Yes	Free
[48]	Containernet	Containernet is a Mininet fork that allows the integration with Docker containers. This characteristic expands the type of experiments supported by the emulator.	< 1000	General	High	Supported	Yes	Free
[49]	NEMO	NEMO is an OS-independent network emulator used for large-scale testing. The platform facilitates capturing traffic traces using standard tools.	> 1000	Research	High	Not supported	No	Free
[132]	OAI	OAI aims at developing open-source solutions for wireless communication networks. It focuses on various aspects of advanced wireless systems.	> 1000	Research	Medium	Not supported	No	Free
[134]	srs-RAN	srsRAN provides SDN-based wireless solutions to experiment with different wireless communication standards such as 4G LTE and 5G.	> 1000	Research	Medium	Not supported	No	Free
[135]	openLEON	OpenLEON emulates integrated wireless networks targeting 4G LTE and 5G technologies.	> 1000	Research	Medium	Not supported	No	Free
[50]	CORE	CORE is a real-time network emulator focused on the rapid instantiation of hybrid topologies. It also integrates with real hardware.	< 1000	Research	Medium	Not supported	Yes	Free
[51]	Cisco CML	CML is a proprietary emulator to run experiments with Cisco virtualized devices. The platform orchestrates VMs to reproduce scenarios rapidly and efficiently.	< 100	General	High	Not supported	No	Free in the cloud
[52]	IMUNES	IMUNES is a lightweight emulator that focuses on TCP/IP-based testing. The tool provides a GUI that facilitates creating topology and instructing networking classes.	< 100	General	High	Not supported	No	Free
[53]	SEED Labs	The SEED Labs is a collection of more than 30 hands-on exercises covering various cybersecurity topics. These exercises are distributed as prebuilt VMs images that learners download and run on their computers.	< 1000	Education	Medium	Not supported	No	Free
[54]	NetEm	NetEm is a Linux-based emulator used to alter the properties of virtual links to reproduce various network conditions.	> 1000	General	Low	Not supported	Yes	Free
[55]	DummyNet	DummyNet is a link emulator to reproduce mid-scale networks across multiple OSes. The tool integrates with real hardware to support complex experiments.	< 1000	Research	Medium	Not supported	Yes	Free
[56]	Mahimahi	Mahimahi simulates latency, packet loss, and bandwidth restrictions. It supports a wide range of network topologies, including single-link, star, and tree topologies, and can be configured to simulate different types of networks, such as wired or wireless networks, 3G or 4G mobile networks, and satellite or long-haul networks	< 1000	Research	Medium	Not supported	No	Free
[70]	Netlab Academic Cloud	Netlab Academic Cloud is a distributed system that provides computing resources to support virtual labs on networking, cybersecurity, virtualization, and other topics. The platform dynamically allocates the resources to enable learners to perform hands-on experiences.	< 100	Education	High	Supported	Yes	Paid
[58]	AWS	AWS is a cloud computing platform that offers a wide range of cloud-based computing services, including computing power, storage, and databases, among others.	> 1000	Research	Flexible	No	Yes	Paid
[59]	GCE	GCE provides virtual machines running on a global infrastructure. GCE allows users to easily deploy and manage virtual machines and associated resources, such as storage and networking, and provides a range of cloud-based services, including machine learning, data analytics, and Internet of Things (IoT) services.	> 1000	Research	Flexible	No	Yes	Paid
[60]	Microsoft Azure	Azure provides a range of management and monitoring tools, including automated scaling and load balancing, as well as security and compliance features. The platform also supports a wide range of operating systems and programming languages.	> 1000	Research	Flexible	No	Yes	Paid

pendent on the specific configuration and deployment of the application. The authors identified a number of best practices for configuring and deploying HPC applications on Azure, including the use of specialized Azure virtual machine types, the optimization of communication libraries and protocols, and the use of tools for monitoring and managing network performance. The study demonstrated that Azure can be an effective platform for running HPC applications, particularly for embarrassingly parallel workloads. However, the authors caution that users should carefully consider their specific application requirements and deploy their applications in a manner that maximizes performance and scalability.

6.4.5. Comparison, Discussion, and Limitations

Netlab is a flexible tool for supporting IT education and running repeatable and scalable experiments. The main limitation is that it is a proprietary solution that incurs upfront costs. Although the academic cloud subscription-based model can facilitate access to more users, using it as a research environment requires complete control of the management software. On the other hand, the SEED labs are open to anyone who wants to gain more insights into cybersecurity topics. It relies on full virtualization, where the VMs used for delivering the labs do not demand many resources from the host machine. However, the SEED labs only cover cybersecurity topics more suitable for computer science courses rather than training IT professionals. These two platforms enable educators to provide interactive hands-on experiences, guiding students through essential cybersecurity domains like ransomware detection [173], phishing prevention, data breach mitigation, cryptography, and various others.

On the other hand, AWS, GCE, and Microsoft Azure are three of the most popular cloud computing platforms in the market. While they offer similar services and features, there are some differences that set them apart. For instance, AWS, GCE, and Azure offer different pricing models and rates. AWS is generally considered to have the most complex pricing structure, with many different pricing options and discounts available. Azure is often considered to have the most straightforward pricing, with a focus on pay-as-you-go and predictable costs. GCE falls somewhere in between, with a range of pricing options and discounts available. Regarding services, AWS, GCE, and Azure all offer a similar range of cloud services, including computing, storage, networking, and security. However, there are some differences in the specific services and features offered by each platform. For example, AWS has a broader range of computing options, including specialized instances for machine learning and GPU computing. Azure has a strong focus on hybrid cloud solutions, with tools for integrating on-premises infrastructure with Azure services. GCE has a strong focus on containerization and Kubernetes orchestration. All three platforms offer web-based interfaces for managing cloud resources, as well as APIs and command-line tools for automation. However, there are some differences in the ease of use of these interfaces. AWS is often considered to have a steep learning curve, due to its complex interface and extensive documentation. Azure is often considered to be the most user-friendly, with an intuitive interface and strong integration with other Microsoft products. GCE falls somewhere in between, with a relatively simple interface but some complexity in managing containerized workloads. AWS is the largest cloud computing platform in terms of market share, with over 30% of the market. Azure is the second largest, with around 20% of the market. GCE is a smaller player, with around 10% of the market. In summary, AWS, GCE, and Azure are all strong cloud computing platforms that offer similar services and features but differ in pricing, services, ease of use, and market share. Users should consider their specific requirements and preferences when choosing a cloud platform.

6.5. Summary and Lessons Learned

Network emulators are essential tools for simulating and testing network scenarios, protocols, and technologies. Each network emulator serves a specific purpose and target audience. For example, Mininet and Containernet are tailored for SDN experiments, while Dummynet focuses on bandwidth constraints. Researchers should choose an emulator that aligns with their simulation needs and research focus. Emulators such as IMUNES and Mahimahi offer advanced features and fine-grained control for experienced researchers seeking more precise network emulation. On the other hand, OAI and srsRAN specialize in wireless communication scenarios. openLEON caters to researchers studying satellite communication networks, demonstrating the value of specialized emulators for specific research areas. Mahimahi and NetEm stand out for their fine-grained control over network impairments, while AWS, GCE, and Microsoft Azure offer cloud-based environments with advanced capabilities for scalable and distributed network simulations.

Network emulation systems are available for both open-source and commercial routing platforms. While such network emulation platforms offer limited data-plane forwarding performance or ASIC-level fidelity of production network hardware, they provide realistic control plane and management-plane behavior. Network VMs are not intended to emulate the hardware characteristics of physical network equipment or to validate aspects such as convergence time, as timings can differ. Moreover, network emulators are a flexible solution to deliver educational content and orchestrate reproducible experiments. Platforms such as Mininet, Containernet, and VMs used in the SEED labs, can integrate with Netlab to scale up and create a hybrid experimentation environment. Finally, the P4 virtual labs available in the academic cloud can fulfill the increasing interest in P4-programmable data planes. Considering that creating an appropriate environment to run a P4 software switch is a complex process, learners can use these virtual labs to reduce such overhead when learning P4 [165]. These virtual labs cover the fundamentals of P4-programmable data planes in an incremental approach.

Network emulators play a crucial role in various research and educational settings, catering to a wide range of network simulation needs. Researchers should consider factors such as specialization, user-friendliness, advanced control, scalability, cost, and the focus of the emulator when selecting the most appropriate tool for their specific

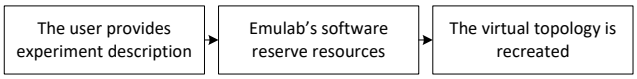

Figure 14: Emulab's experiment lifecycle. Steps for recreating a virtual network topology within an Emulab-based testbed [174].

research requirements. The choice of network emulator should align with the research objectives, expertise level, and available resources, enabling effective experimentation and evaluation of network protocols and technologies.

7. Network Testbeds

This section describes the network testbeds currently used to run large-scale experiments in several areas, such as cloud computing, cybersecurity, programmable networks, distributed computing, and other fields. First, this section covers the general-purpose testbeds available to conduct research and instruct networking courses. Second, it explains the specific purpose of testbeds that explore areas such as cybersecurity, source routing, and cloud-based application. Then, this section analyzes the testbeds that use real traffic from production networks to validate new systems. Lastly, the section provides the comparison, discussion, and limitations found in the surveyed works, followed by the lessons learned.

7.1. General-Purpose Testbeds

The testbeds in this category support a diverse set of experiments, including networking, machine learning, cybersecurity applications, IoT, blockchain, and others. The hardware that experimenters can use ranges from GPUs, NetFPGAs to specific instruments such as telescopes, microscopes, and other appliances available only at specific locations. Moreover, many of these testbeds facilitate users to attach their devices to extend the capabilities of existing research testbeds.

7.1.1. Emulab

Emulab [11] is a network testbed used to recreate a wide range of experimentation environments in which users can develop, debug, and evaluate complex networked systems. With Emulab, experimenters can interact with a cluster of resources, including physical machines, virtual machines, containers, and storage clusters. The platform supports large-scale experiments allowing users to control several nodes. Emulab's management allocates resources according to the experiment's demands. For instance, if the experiment aims to achieve fidelity, the testbed will consist of bare-metal hardware. On the other hand, if the resources are used for research that does not demand high fidelity (e.g., during software development, education, or reliability studies), the hardware constraints can be reduced and moved to a hybrid or a completely emulated environment. For this purpose, Emulab leverages virtualization as the main tool to efficiently allocate these resources.

Siaterlis *et al.* [174] conducted a rigorous review of the Emulab platform focusing on experiment fidelity, repeatability, measurement accuracy, and interference. As a result, they produced a tutorial to help experimenters to

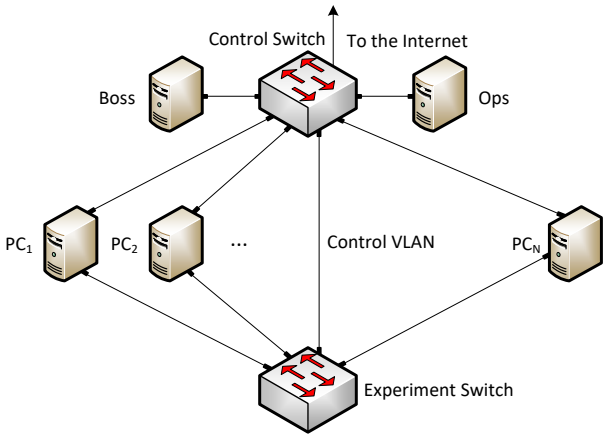

Figure 15: Emulab hardware [177].

decide whether Emulab is a suitable platform to conduct their experiments. This tutorial could serve as a complement to the user guide available on Emulab's website [175]. In the tutorial, the authors described the experiment lifecycle in Emulab, summarized in Figure 14. First, the experimenter provides the experiment script, which consists of a description of the topology with a script [176]. The experiment script contains the components (e.g., switches and routers) and instances (e.g., virtual machines and monitoring interfaces) needed for the experiment. The script is also useful to port and shares the topology with other experimenters. Second, the Emulab software instantiates the components and allocates the required resources on bare-metal servers. This process is known as swap-in, which establishes the beginning of the experiment, as opposed to swap-out, which indicates the end of an experiment. Emulab comprises a pool of resources that are commonly available to experimenters. Therefore, if the requested resources are unavailable, the experimenter must redefine the topology. Third, the Emulab software isolates the links described in the topology by configuring them in different VLANs. Finally, the monitoring tools are activated to let the experimenter run regular tests.

Currently, the Emulab cluster at the University of Utah hosts approximately 500 experimentation nodes connected by thirteen Ethernet switches. The cluster has added new nodes in large homogeneous batches

Table 5: Emulab Hardware.

Node	Hardware	Description
d430	CPU	Intel E5-2630v3 8-Core at 2.4 GHz (Haswell)
	Memory	64GB DDR4 ECC at 2133MT/s
	Storage 1	200 GB 6G SATA SSD
	Storage 2	1 TB 7.2K RPM 6G SATA HDDs
	NIC 1	Intel I350 1GbE NICs
	NIC 2	Intel X710 10GbE NICs
d820	CPU	Intel E5-4620 8-Core at 2.2 GHz (Sandy Bridge)
	Memory	128GB DDR3 at 1333MHz
	Storage 1	250 GB 7.2K RPM 3G SATA HDD
	Storage 2	600 GB 10K RPM 3G SAS HDDs
	NIC	Intel X520 10GbE NICs
d710	CPU	Intel Xeon E5530 4-Core at 2.4 GHz (Nehalem)
	Memory	12GB DDR2 ECC at 1066MHz
	Storage 1	500 GB 7.2K RPM SATA HDD
	Storage 2	250 GB 7.2K RPM SATA HDD
	NIC	Broadcom BCM5709 1GbE
pc3000	CPU	Intel Xeon single-core at 3.0 GHz (Nocona)
	Memory	2GB DDR2 ECC at 400MHz
	Storage	146 GB 10K RPM SCSI HDD
	NIC 1	Intel 1GbE
	NIC 2	Intel 10/100

and kept the old ones. Maintaining a large number of homogeneous node groups achieves a high degree of flexibility and ensures reproducibility. Old and new nodes are compatible to enable more freedom in resource assignment. The rationale for keeping old nodes is based on the assumption that network researchers are interested primarily in the network. Therefore, some experiments can run on older, low-powered hosts.

Emulab's physical topology consists of a control switch and an experiment switch attached to the experiment PCs (i.e., nodes), as shown in Figure 15. Additionally, the hardware setup includes control nodes (i.e., Boss and Ops) responsible for user access control, fileservers, and other management tasks. Table 5 summarizes the characteristics of the servers in the Emulab cluster at the University of Utah. This cluster contains around 500 servers which include four types of CPU generations. There are 160 d430 nodes, 16 d820 nodes, 160 d710 nodes, and 160 pc3000 nodes. All the nodes are connected to a control network and an experiment network. The control network is connected to the Internet for remote access and experiment management. The experiment network connects the nodes and layer two Cisco, Arista, and Dell (Force 10) switches. Each host has at least four interfaces on this network.

7.1.2. GENI

The Global Environment for Networking Innovation (GENI) is a distributed virtual laboratory to run experiments in network science, services, and security [12]. GENI started in 2004 as a global experimental playground and testbed for research and education. The main goal of GENI is to address the inherent problems of Internet ossification, where the rigid nature of Internet protocols limits innovation. GENI provides the resources for network experimentation in real hardware at small, medium, and large scales. These resources are geographically distributed, allowing researchers to reproduce WAN conditions. This capability enables researchers to start with small experiments that can be scaled to test them in more realistic scenarios. GENI allows users to experiment with bare-metal servers and interconnected VMs, which can reproduce distributed network scenarios and run various software. A particular organization does not own the GENI testbed. Instead, its resources are independently provided, owned, and controlled by individual contributing organizations.

Figure 16 shows the GENI network architecture consisting of a control plane and a data plane. The control plane is used to reserve, discover, access, program, and manage the testbed compute and communication resources. The data plane provides the resources for individual experiments. These resources define the experimental topology, compute resources connected to the network, link bandwidth, programmable switches, and controllers used to implement custom packet-forwarding algorithms. Networks supported by GENI are isolated using Ethernet Virtual Local Area Networks (VLANs). These isolated units are known as slices, and they guarantee traffic and performance isolation among experimenters.

Moreover, the data plane in GENI implements deep programmability, allowing programmers to process and perform actions on custom headers using OpenFlow-capable software switches. GENI is integrated with the Internet2 network [179], which provides national data plane connectivity and allows experimenters to program the software switches available in their network. This capability enables the testing of custom network protocols and controllers that operate in large-scale scenarios.

GENI relies on virtualization and programmable layer two networks to reserve and isolate computing resources. Virtualization technology provides the illusion of exclusive ownership of shared resources, whereas layer two programmable OpenFlow switches slice the network into isolated segments. Moreover, GENI implements deep programmability by allowing the experimenter to customize the behavior of the OpenFlow switches. With this capability, part of the computing, storage, routing, and forwarding are performed in the network. Figure 17 shows three types of racks projects available in GENI. Figure 17(a) depicts the ExoGENI rack that provides flexible virtual networking topologies solutions, including OpenFlow. This type of rack is typically deployed in campus networks supporting multi-site cloud applications. Figure 17(b) shows the InstaGENI rack components. The InstaGENI rack can be deployed in campus networks at a large scale to deliver Internet cloud application support using OpenFlow switches to slice the network. InstaGENI racks are deployed outside the organization's firewall. Figure 17(c) depicts the OpenGENI rack architecture. This rack supports GENI-compliant equipment in a Dell rack, where the data plane switch is a production OpenFlow [186].

GENI provides isolated and dedicated resources called slices. Experimenters can design and customize network typologies within a slice without interfering with other projects. However, reserving resources for an experiment is subject to their availability. Therefore, an experimenter follows the process depicted in Figure 18, which shows the lifecycle of an experiment in GENI. The stages in the figure suggest that there is no satisfactory network setup for an experimenter. GENI addresses this issue by providing an open application programmer interface (API) to facilitate the development of multiple experimenter tools. The testbed aims to build a community of interoperating experimenter tools with this approach. These tools are user interface software (i.e., Graphical User Interfaces (GUIs) and Command-Line Interfaces (CLI)) that facilitate reserving resources in the testbed. If the required resources are unavailable, the user redesigns the experiment to use the available resources. The experimenter uses the experiment management tools to automate and maintain the configuration process in the execution stage. Then, the topology is ready to run experiments and produce results. Before finalizing the experiment execution, the experimenter can reserve more resources to run the experiment at a larger scale.

GENI users can access resources from other testbeds using their accounts. In this way, they can add more resources to their GENI slice to run experiments that span multiple testbeds worldwide. These testbeds are CloudLab [187], Chameleon Cloud [69], Emulab [11], ORBIT [188], WITest [189], w-iLab.t [190], and PlanetLab [191].

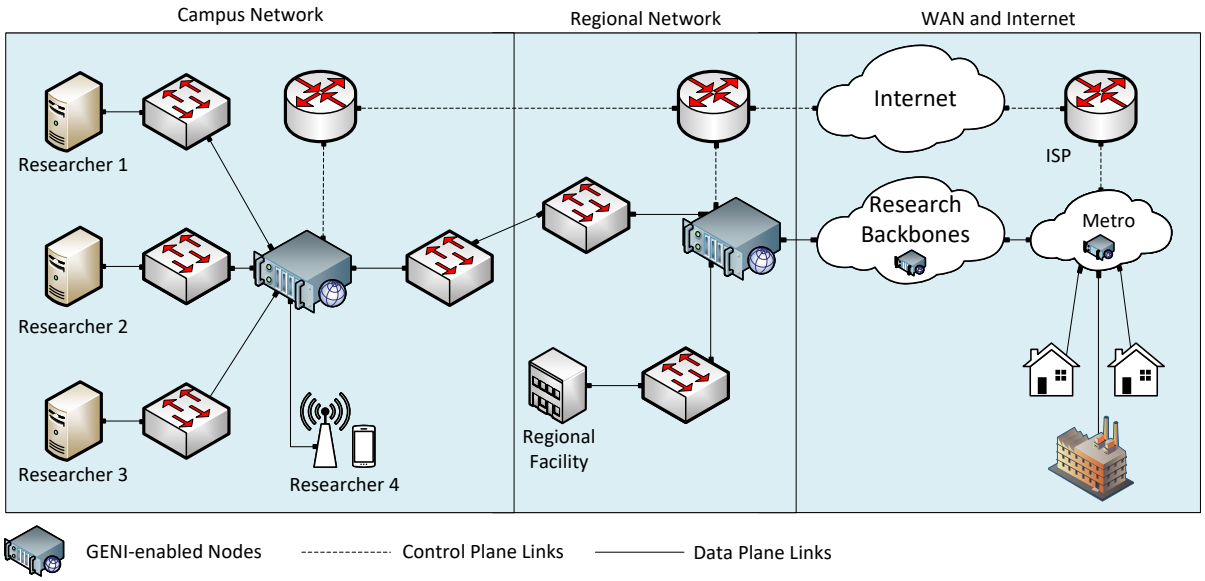

Figure 16: GENI architecture [178].

Table 6: GENI Rack resources specification.

Rack Type	Device	Model	Description
ExoGENI	VPN Appliance	Juniper SSG5 [180]	Backup management access
	Management Switch	IBM BNT G8052R [181]	1G client/10G uplink ports
	Worker Node	IBM x3650 M3 [182]	• CPU: Intel Xeon 5650 2.66GHz • Memory: 48 GB • Storage: 2x146 GB 10K SAS • CPU: Intel Xeon 5650 2.66GHz • Network: Dual 1G/10G adapter
	Management Node	IBM x3650 M3 [182]	• Memory: 12 GB • Storage: 2x146 GB 10K SAS
InstaGENI	Management Switch	HP Procurve 2620 [183]	24 10/100/100 Mbps ports, 2 1 Gbps ports
	Experiment Node	HP ProLiant DL360 G7 [184]	• CPU: Intel Xeon 5650 2.66GHz • Memory: 48 GB • Storage: 1 TB • Network: Dual 1G/10G adapter
	Control Node	HP ProLiant DL360 G7 [184]	• Memory: 12 GB • Storage: 4 TB RAID
OpenGENI	Management Switch	HP Procurve 2620 [183]	48 1 GbE ports, 4 10 GB ports
	Compute Node	PowerEdge R620 XL [185]	• CPU: Intel Xeon E5-2600 v2 3 GHz • Memory: 32 GB • Storage: 300 GB • Network: Dual 1G/10G adapter
	Control Node	Dell PowerEdge R620 XL [185]	• Memory: 32 GB • Storage: 300 GB

Initiatives such as EdGENI [192] facilitate using GENI resources by providing a user-friendly GUI and customized VM images. In this way, EdGENI offloads the configuration overhead by introducing a desktop environment where the user can save, duplicate, and share VMs with other users. The authors sustain that EdGENI built-in features improve the user experience and extend the user base of the GENI testbed, introducing new users to network education and experimentation. With EdGENI, an instructor can create a lab environment and share it with students. Then, the students work directly on the experiment in a shared environment skipping the initial configuration, which can be error-prone and time-consuming. Finally, the authors made available the repository [193] that contains the scripts, tools, and laboratories used to learn cybersecurity topics and blockchain principles. The laboratories in cybersecurity cover topics such as symmetric encryption and hash, public key encryption, denial of service (DoS), port scanning, and ARP poisoning. The blockchain laboratories cover wallet setup, smart contract deployment, and interaction, Solidity [194], Remix IDE [195], and others.

Figure 19 explains the usage scenario of EdGENI. With EdGENI, the instructor prepares and distributes lab exercises to students by defining the Rspec and lab instruction files. The Rspec file is a snapshot of the lab environment that offload student to perform manual configuration. The Rspec file defines the resources utilized in the GENI slice dedicated to the students. Additionally, the instructor provides the lab instructions with the activity requirements and steps to complete the lab exercises. The instructors can deploy the lab environment as a function of the availability of the resources.

7.1.3. StarBED

StarBED [61] is a large-scale general-purpose network testbed based on local nodes. With StarBED, experimenters can conduct research that includes layer two

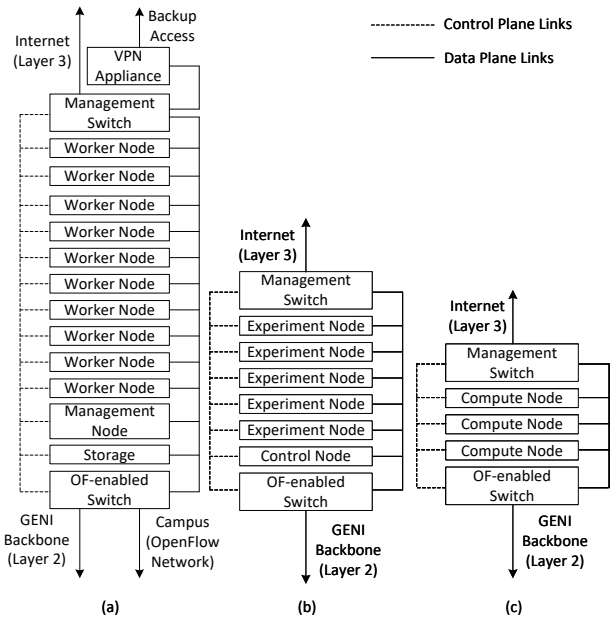

Figure 17: GENI rack hardware. (a) ExoGENI. (b) InstaGENI. (c) OpenGENI. [178].

switch benchmarks with multicast traffic, observation of TCP behavior, performance analysis, wireless network emulation, mobile network experimentation, overlay networks, and experiments with real traffic. StarBED comprises more than 512 nodes to build large-scale experiment topologies. The design of StarBED aims to allocate the experiment resources at one site to provide more flexibility when building complex topologies. Nodes in StarBED are classified into five groups according to their functionalities, such as the number of network interfaces.

StarBED offers the tools for developing a research idea from the early stages. The steps concerning the development process are summarized in Figure 20. The development begins with a new design proposal, then evaluated using a simulator. Once the simulation results meet the expected behavior, the system migrates to a laboratory-scale testbed already hosted in StarBED. If the nature of the topology requires testing in a large-scale testbed, the experimenter can allocate the necessary StarBED resources to conduct the experimentation. When the large-scale experiment meets the expected goal, it is ready to implement in a production environment. Figure 21 shows StarBED architecture, which consists of an experiment network, a management network, a service network, and a user network. The experiment network comprises high-performance switches connected to experiment nodes and

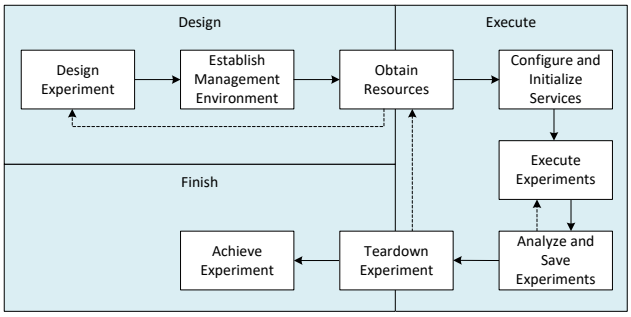

Figure 18: Experiment lifecycle in the GENI testbed. [178].

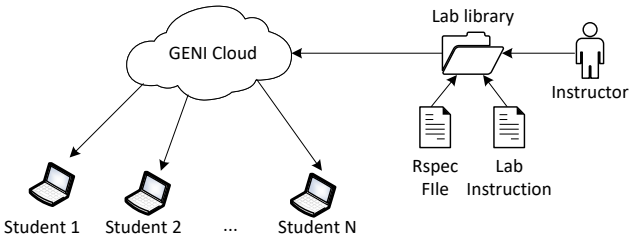

Figure 19: EdGENI lab environment. [192].

specific-purpose equipment. A management network acts as an interface between the service network and the experiment network, providing the resources experimenters will use to run experiments. The user network interacts with the service network by administering experimenters' accounts. Moreover, StarBED allows the creation of links between the experiment network and the external testbeds.

7.1.4. Grid'5000

Grid'5000 [62] is a distributed, highly reconfigurable testbed that aims at large-scale computing experimentation. Grid'5000 provides a scientific tool for computer scientists similar to the large-scale instruments used by physicists, astronomers, and biologists. The testbed design goal is to study the complex interaction of interconnected systems, which is not realistic with simulators and emulators. Grid'5000 comprises 15,000 CPU cores and 800 compute nodes grouped in homogenous clusters supporting various technologies such as Persistent Memory (PMEM), GPUs, SSD, NVMe, Infiniband [196], and others. The testbed is highly reconfigurable and programmable, allowing researchers to experiment with a fully customized software stack running on bare-metal deployments. With this feature, experimenters can obtain isolation at the networking layer. Moreover, the testbed supports advanced monitoring and measurement features collecting network traces and power consumption to provide a deep understanding of experiments. Finally, the authors designed Grid'5000 to be an environment that supports open science and reproducible research.

The testbed supports various experiments ranging from resource-intensive distributed systems to IoT orchestration. Grinsztajn *et al.* [197] used the testbed to experiment with a reinforcement learning-based strategy for heterogeneous dynamic scheduling. The proposed

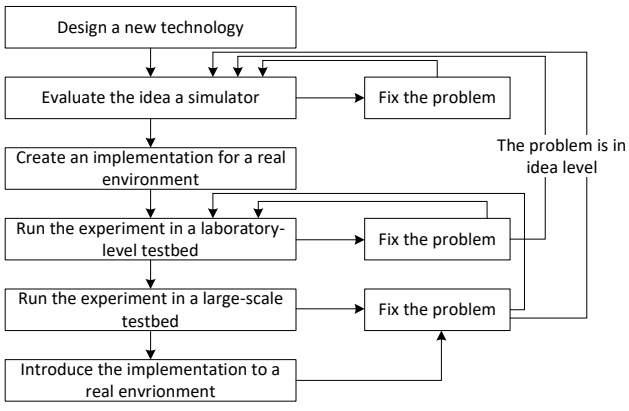

Figure 20: StarBED experiment lifecycle [61].

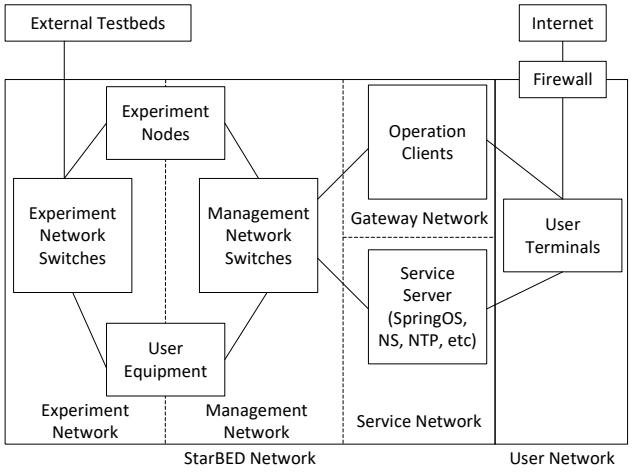

Figure 21: StarBED architecture [61].

system is a reinforcement learning algorithm for the dynamic scheduling of computations modeled as a Directed Acyclic Graph (DAG) where allocation and scheduling decisions are performed at runtime. Brandón *et al.* [198] deployed different service-oriented architectures in Grid' 5000 to evaluate an analysis framework that aims at identifying the root cause of anomalies in applications. Results show that the author's approach is 19.41% more effective than a machine learning method that does not consider the relationship between elements. Donassolo *et al.* [199] created an IoT fog environment in Grid' 5000 to implement an orchestrator. This orchestrator aims to automate the deployment, scale the management, and migrate microservice-based IoT applications. The authors reported that the testbed successfully reproduced most fog characteristics, such as geographical distribution and heterogeneity. Sarmiento *et al.* [200] used Grid' 5000 to implement the proof-of-concept of a mechanism that provides network virtualization operations to the users while dealing with scalability and intermittent network properties of geo-distributed infrastructures. The system can reduce management traffic overhead, automate multitenancy configuration, and enhance collaboration.

7.1.5. Comparison, Discussion, and Limitations

Emulab is a network testbed that allows researchers to create and test experimental network topologies. It provides a web-based interface for creating and managing experiments on a shared pool of resources. Emulab's strength lies in its ease of use and support for various network experiments, making it suitable for researchers of different expertise levels. However, Emulab's accessibility might be limited to certain research institutions, affecting its availability for researchers outside those institutions. The Emulab testbed supported many experiments and facilitated education in several computing areas [201–204]. Siaterlis *et al.* [174] analyzed the use of Emulab to run scientifically rigorous experiments by assessing the fidelity, repeatability, and accuracy of the testbed. Their finding pointed out that Emulab can realistically reproduce the performance of real networks even with the emulated delay. The authors recommended obtaining realistic results in scenarios with emulated delay, and the bandwidth utilization must not exceed 30%. The authors also pointed out that Emulab presented high repeatability in scenarios with moderate network load and recommended increasing hardware allocation to reproduce more accurate results. Finally, the authors suggested that experimenters must be aware of the interference from the tools used to conduct the experiments.

GENI is a large-scale network testbed that allows researchers to experiment with advanced network technologies. It offers a federated infrastructure of resources across multiple institutions. GENI's strength lies in its extensive infrastructure, providing researchers with a distributed platform for conducting experiments on a larger scale. Setting up experiments on GENI may be more complex due to the distributed nature of the infrastructure, and access to resources requires administrative approval. Edwards *et al.* [205] evaluated the GENI testbed to provide general guidelines on conducting networking experiments in shared, public testbeds. The authors describe a methodology to be followed by new experimenters to write and deploy repeatable and sharable experiments. They emphasize that defining the experiment's goal at early stages is essential to anticipate the number of resources to be allocated. This guideline suggests that an experimenter must design the experiment to decide how the resources will be employed to conduct repeatable tests. Then, the authors provide guidelines on automating experiments' execution following the best practices. Automating a task involves combining the scientific method with software engineering and system administration skills. The authors recommend starting with a small test to detect misconfigurations and keeping a version control to track the working configurations. Finally, the paper explains the steps to build scalable experiments: starting with a small deployment and ensuring that produced results meet the expectations.

StarBED is a network testbed designed for large-scale experiments with a high number of nodes. It aims to provide a realistic environment for testing and validating network protocols and technologies. StarBED can handle large-scale experiments, offering researchers the ability to test network protocols under realistic conditions. The hardware and infrastructure requirements for StarBED might be challenging for smaller research groups or institutions.

Grid'5000 is a large-scale experimental testbed that provides resources for researchers to conduct experiments in various fields, including networking. It allows researchers to deploy custom environments on a distributed infrastructure. Grid'5000's strength lies in its distributed infrastructure, enabling researchers to conduct experiments in realistic, geographically distributed settings. Setting up experiments on Grid'5000 may involve administrative steps and access to resources requires administrative approval. Balouek *et al.* [206] evaluated Grid'5000 software and services stack to support large-scale experiments using virtualization technologies as building blocks. The evaluation included customized software environments, the reservation of dedicated network domains, the isolation capabilities of the testbed, and the automation of experiments using APIs. The experiments consisted of orchestrated tests that used many VMs (i.e., $\tilde{4}000$) across several locations.

Results show that Grid'5000 experiences some challenges when managing applications that use large volumes of data. The authors acknowledge that a main limitation of the platform is the diversity of the software stack used to manage the testbed. Simplifying the management interface can add more visibility to the experiments and facilitate the orchestration of more complex experiments.

7.2. Specific-Purpose Testbeds

This category comprises testbeds dedicated to specific types of experimentation, such as cybersecurity, routing protocols, and cloud-specific applications. These testbeds aim to push innovation in a specific field that requires specialized hardware, custom configuration, and dedicated access to resources that are not available in regular testbeds. These requirements intend to enforce consistency in the results and facilitate scaling up the tested prototypes.

7.2.1. CloudLab

CloudLab [10] is a testbed for cloud computing research. It provides access and control over bare-metal servers comprising large computing, storage, and networking resources. With CloudLab, a researcher can run fully observable and repeatable experiments that cannot be conducted on traditional clouds. CloudLab gives control, visibility, and performance isolation, supporting experiments on cloud architectures and distributed systems. Therefore, experimenters can use, see, instrument, modify, and monitor all levels of the cloud stack and applications, including virtualization, networking, storage, and management abstractions. CloudLab comprises more than 25,000 cores distributed across three main sites the University of Wisconsin, Clemson University, and the University of Utah. The CloudLab deployment can also interoperate with existing testbeds such as GENI, Emulab, Chameleon Cloud, and, most recently, FABRIC.

With CloudLab, researchers request on-demand resources to run experiments. The Cloudlab staff handles these requests and takes charge of the testbed's construction, maintenance, and operation, allowing researchers to only focus on their experiments. Experiments in CloudLab are instantiated as profiles, which contain a description of the hardware resources (i.e., servers, switches, VMs) that the experiment will use. The profile also describes the software needed to run the experiment. This software comes in the form of disk images, GitHub repositories, and scripts. Instantiating a profile is how CloudLab allocates the available hardware that meets the profile's specifications and provisions the software and configuration options described in the profile. The most popular cloud software in CloudLab is OpenStack [142]. CloudLab's workflow to run an experiment starts by creating a new profile that involves disk images, installing custom software, and describing the hardware that meets the experiment's needs. Each profile expires after a few hours, which can be extended if needed for additional hours. If the experimenter wants to extend more (e.g., for weeks or months) he/she must submit a request to the administrator. The testbed comprises various hardware, including programmable Ethernet switches, GPUs, bare metal servers, Infiniband NICs, and high storage capacity. The CloudLab user interface leverages in-house software designed to support its research testbeds. CloudLab prioritizes the reproducibility of the experiments rather than on elasticity. Thus, in CloudLab, experimenters can easily describe their experiments and port them to other environments. CloudLab comprises three data centers in the following locations:

- CloudLab Utah: this data center hosts servers with modest specifications. There are 585 HPE Moonshot servers, which consist of 45 low-power Intel Xeon-D or ARM-based servers in each chassis. Each chassis supports two 10 Gbps switches which work as top-of-rack switches and are connected to a 160 Gbps switch. Another group of resources consists of 200 servers connected to a traditional 25 Gbps Ethernet network and to a layer 1 network for controlling the physical layer. This layer aims to wire nodes to Ethernet switches controlled by the user.

- CloudLab Wisconsin: The goal of this facility is to reproduce the hardware available in modern data centers. The servers are provisioned with dual sockets and have a mix of spinning and Hard Drives Disks (HDD) and Solid State Drives (SSDs). The servers have a large number of disks allowing the user to configure a Storage Area Network (SAN). These servers are also equipped with GPUs, which enable experimentation on machine learning and large-scale computing.

- CloudLab Clemson: This data center specializes in providing more CPU cores and a greater amount of RAM per core. These resources make it appropriate for hosting big data analytics applications, running high-performance computing workloads, and supporting a large number of VMs. The deployment also comprises an Ethernet experiment network topology that interconnects three core switches, each connected to a companion top-of-rack switch handling direct server connectivity. Additionally, experimenters can also use a 40 Gbps Infiniband [196] network dedicate to High-Performance Computing (HPC) and Remote Direct Memory Access (RDMA).

The hardware at each site is scheduled to grow up over time. No hardware has yet been retired.

Park *et al.* [208] integrated into CloudLab SDN security hands-on training labs. The authors developed an open laboratory leveraging the unique features (i.e., OpenFlow switches) provided by CloudLab where students can engage with SDN security concepts by following a step-by-step lab manual. The set of labs covers various security issues that include denial-of-service (DoS) attacks and application attacks.

Ngo and Denton [209] designed a hands-on environment to learn cluster computing topics using CloudLab. The authors employed virtualization to dedicate computing resources for each learner. In this lab environment, the learner uses network elements to like geographically distributed VMs running on top of bare metal hypervisors. The cluster computing course covers topics that include the Beowulf model of networked computers,

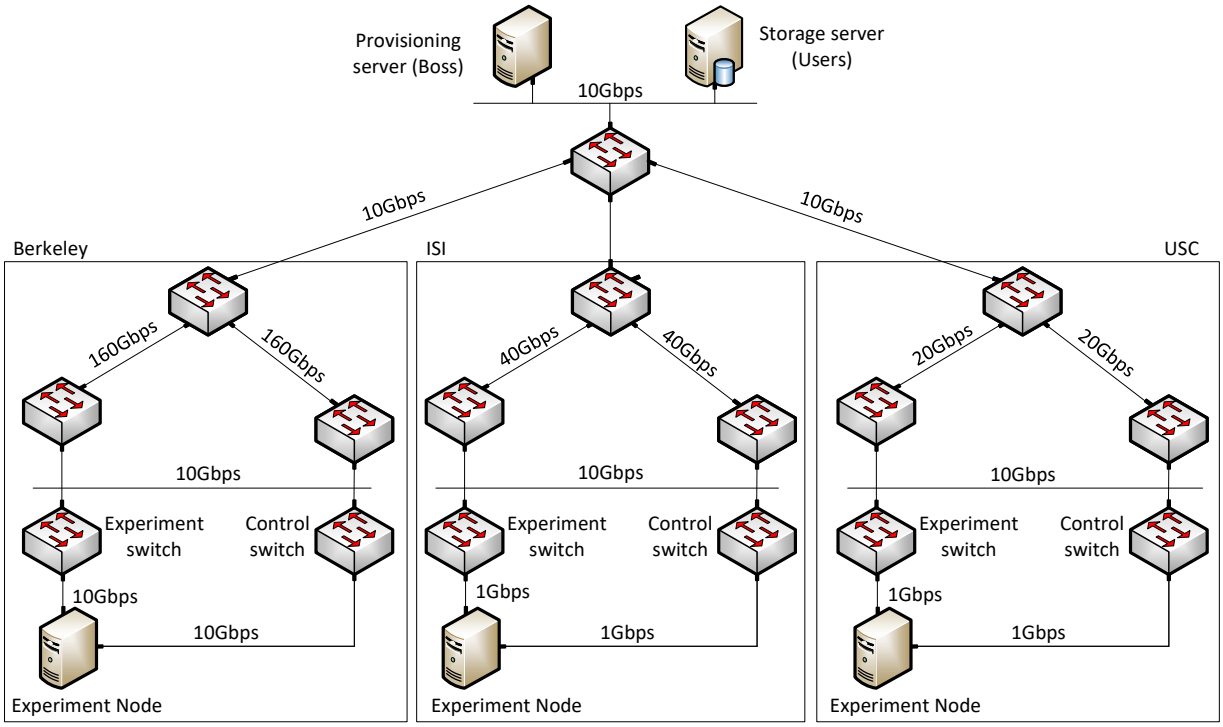

Figure 22: DETERLab architecture [207].

distributed file systems, the message-passing programming paradigm, scheduling on a cluster of computers, big data, and the map-reduce programming paradigm. The course aims at providing the fundamental concepts of cluster computing through hands-on training with real-world technologies and platforms.

Ngo *et al.* [210] used CloudLab to support teaching Parallel and Distributed Computing (PDC) in undergraduate courses. The authors solve the class preparation and lab configuration overheads by offloading the classroom resources by unifying on-site resources with remote computing resources hosted in CloudLab. The course covers PDC programming and architectural concepts, including speedup methods, real-time results, visual results, interactivity, active learning, reproducibility, and accessibility. Results show that CloudLab successfully supported training sessions with many learners. Moreover, a survey conducted at the beginning and the end of the course indicated that the students felt more affinity towards topics such as Message Passing Interface (MPI), MapReduce, Spark, and parallel programming.

7.2.2. DETERLab

The cyber DEfense Technology Experimental Research Laboratory (DETERLab) is a scientific computing testbed that enables experimenters to research, develop, discover, and test cybersecurity ideas [9]. DETERLab serves a broad community that involves academia, industry, and government. Projects in DETERLab cover behavior analysis and defensive technologies, including Distributed Denial of Service (DDoS) attacks, worm and botnet attacks, encryption, pattern detection, and intrusion-tolerant storage protocols. DETERLab facilitates a large-scale experimentation environment for rigorous and repeatable tests. With DETERLab, experimenters can securely observe and interact with malicious software operating in real-world network environments. The testbed aims to lead cyber-defense innovations focused on developing robust systems. DETERLab supports sharing resources among multiple concurrent experiments, providing an incrementally developed library of tools, interfaces, and datasets for security experiments. Moreover, DETERLab offers a wide range of components to enable hands-on training for colleges and universities via the DETER Project [211], which is committed to promoting research in cyber security through the use of DETERLab's innovative methods and advanced tools.

DeterLab comprises around 700 nodes distributed in three locations: the University of Southern California (USC), the Information Science Institute (ISI) in California and Virginia, and the University of California at Berkeley. Each location is connected via a public 10 Gbps link. Figure 22 shows a high-level architecture of the testbed. The architecture involves two service nodes, Boss and Users, to run provisioning and storage services. Each node has one control interface and multiple interfaces connecting to various switches. Experiments are isolated via VLANs at the experimental and control network. The boss node runs the ControlNet-Isolation (CI) software to implement isolation. The CI software ensures that each node can only send traffic to other nodes in the same experiment [207].

7.2.3. SCIONLab

Kwon *et al.* [64] proposed SCIONLab. This global testbed enables network experimentation in inter-domain routing areas, including path-aware routing, routing policies, and security policies against DDoS attacks. SCIONLab leverages the Scalability, Control, and Isolation On Next-Generation Networks (SCION) Internet architecture, which aims to offer high availability and efficient point-to-point packet delivery, including the mitigation of malicious network operators and devices.

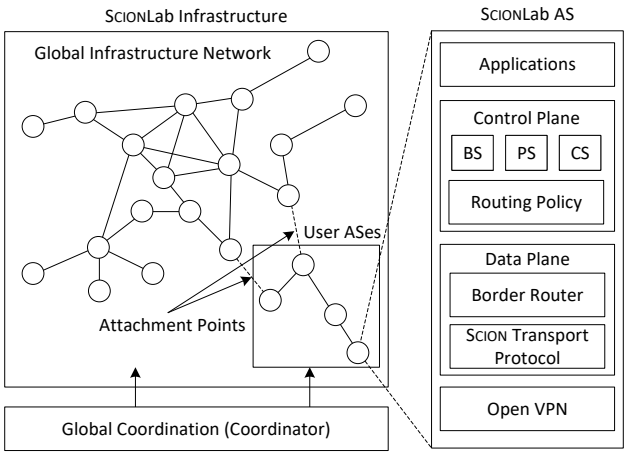

Figure 23: SCIONLab architecture. A global coordinator service connects the experimenters' ASes to the global coordination service [64].

In the SCION paradigm, Autonomous Systems (ASes) are organized into groups of independent routing planes known as isolation domains (ISDs). The goal of an ISD is to provide global connectivity. The administration of an ISD is performed by a group of ASes called ISD core, which define the certificate issuance policy in the ISD. When the ISDs are validated, they can establish a trusted route. SCION considers security aspects, routing infrastructure, and forwarding mechanisms. SCION is a path-based architecture that implements source routing [212] to establish end-to-end inter-domain paths. A cryptographic mechanism restricts the path construction to the routing policies defined by the ISPs and receivers. This way, SCION allows senders, receivers, and ISPs to enable path-aware Internet.

Figure 23 illustrates the testbed architecture. The SCIONLab administrators operate the network infrastructure and the global coordinator. The global topology is created to support a variety of paths between any two ASes, which empowers multipath operations. The infrastructure offers several attachment points, to which user ASes connect to participate in the SCIONLab network. Depending on who owns them, two AS types exist infrastructure AS, which SCIONLab administrators run, and user AS, which is run by the users. Both AS types are regular SCION ASes, with a beacon service (BS), certificate service (CS), path service (PS), and one or multiple border routers. Some infrastructure ASes provide an attachment point that provides connectivity to user ASes. SCIONLab operates with full cryptographic support. Each ISD defines its own roots of trust within a trust root configuration, where the core ASes control the root keys for the control plane. Each AS has a globally unique AS number (ASN) and a public/private key pair, which is validated through a certificate signed by a core AS. The certification infrastructure that includes the certificate authority and the validation mechanisms are part of the SCION architecture [213].

7.2.4. Colosseum

The Colosseum testbed [214] represents a state-of-the-art research platform designed to facilitate large-scale experimentation in the field of wireless communications. Its architecture embodies a highly reconfigurable and expansive RF environment, providing researchers with a controlled yet realistic setting for the evaluation and validation of wireless technologies and algorithms. Colosseum can emulate diverse wireless propagation scenarios, ranging from urban to suburban and rural environments, through the use of programmable antennas. The Colosseum architecture is highly reconfigurable and designed for large-scale experimentation in wireless communications. It consists of several key components that collectively enable the emulation and evaluation of wireless technologies and algorithms in a realistic and controlled environment. At the core of the Colosseum architecture lies its RF environment, which encompasses programmable antennas and extensive RF infrastructure. This allows researchers to create diverse wireless propagation scenarios, accurately replicating real-world settings like urban, suburban, and rural environments. Colosseum provides high capacity to support a large number of concurrent wireless devices. This capability facilitates experiments with high device density scenarios, enabling researchers to assess the performance of wireless networks under realistic traffic loads. The architecture supports the deployment of heterogeneous wireless networks, including various generations of cellular technologies like 4G LTE, 5G NR, and beyond. This ensures the emulation of cutting-edge wireless communication systems. A distinguishing feature of the Colosseum architecture is its high level of programmability and flexibility. Researchers have fine-grained control over network parameters, channel characteristics, interference levels, and mobility patterns, enabling precise and detailed emulations. The architecture includes comprehensive monitoring and data collection capabilities, providing researchers with a wealth of network performance metrics. This data-driven approach empowers in-depth analysis and comparison of different wireless technologies and algorithms. Colosseum incorporates an experimental management system that allows researchers to configure, schedule, and monitor experiments seamlessly. This enhances the efficiency and effectiveness of conducting large-scale wireless experiments

7.2.5. IEEE 5G/6G Innovation Testbed

The IEEE 5G/6G Innovation testbed [215] is a cloud-based, end-to-end 5G network emulator that enables testing and experimentation of 5G products and services. The testbed aims at enhancing current 5G technologies and defining the future 6G functions. The testbed is based on open-source components. This makes it flexible and adaptable to new network functions and features. The testbed is available to members of the IEEE Future Networks Initiative [216], and it is also available to other organizations on a fee-based basis.

The testbed allows experimenters to test their 5G and 6G systems in a realistic environment, helping them to identify and address any potential problems. Additionally, the testbed also helps companies to collaborate with other organizations on 5G and 6G research and development. The testbed offers a range of interconnected advantages for companies engaged in the development of 5G and 6G products and services. Firstly, it provides a realistic environment where these companies can thoroughly test their 5G products and services, ensuring they work as expected. Moreover, this testbed helps these compa-

nies in identifying and resolving any potential issues that may arise during testing, contributing to the overall quality of their offerings. Additionally, the testbed promotes collaboration between different organizations involved in 5G and 6G research and development. By offering a shared platform for experimentation and testing, it fosters teamwork and knowledge exchange, potentially leading to faster advancements in 5G and 6G technologies. Lastly, the testbed's cost-effectiveness is noteworthy, allowing companies to save on testing expenses while still reaping the benefits of a comprehensive and reliable testing environment. The IEEE 5G/6G Innovation testbed serves as a valuable tool that not only aids in testing and problem-solving but also acts as a catalyst for collaborative progress in the field of 5G and 6G technologies.

7.2.6. Comparison, Discussion, and Limitations

CloudLab is a network testbed that allows researchers to conduct experiments with cloud and edge computing technologies. It provides a user-friendly interface and supports a wide range of experimentation capabilities related to cloud computing. CloudLab's strength lies in its focus on cloud and edge computing, making it suitable for researchers interested in testing these technologies. However, CloudLab's availability might be limited to certain research institutions, affecting its accessibility for researchers outside those institutions. Duplayakin *et al.* [187] presented their experiences in designing and operating the CloudLab environment. The paper provides guidelines and good practices for designing and analyzing computing testbeds. They studied CloudLab's operations for four years, considering 4,000 users, 79,000 experiments, and 2,250 servers, switches, and other data center equipment. The authors focused on two sets of attributes to understand critical aspects of the testbed. The first set consisted of the analysis of CloudLab usage, which includes user interaction, the purpose of the experiment, and the implications for facility design and operation. The second set considered monitoring the resource allocation, observing how different algorithms utilized these resources, and how user experience is affected by unexpected behaviors. From analyzing the first data set, the authors concluded that providing low-level hardware access and administrative privileges to experimenters contributed to optimizing the resource utilization in the testbed. Considering the second set of aspects, the authors postulated that as the experimenters notice that the available resources for their experiments become scarce, they tend to submit more reservation requests to use the platform.

DETERLab is a network testbed designed for cybersecurity research. It provides resources and tools for researchers to conduct experiments related to network security and cyber threats. DETERLab focuses on cybersecurity research, offering a platform for testing and evaluating network security mechanisms. Ibrahim *et al.* [217] evaluated DETERLab from an educational perspective. The authors determined that DTERLab allows students and researchers to deploy a wide range of defense and attack mechanisms in cybersecurity scenarios. The authors documented the challenges found in the platform to help other educators to use the testbed effectively. Using a qualitative approach, the authors evaluated the testbed by launching experiments on the platform to evaluate its usability and adaptability. The authors found that getting started with an educational experiment is challenging due to the lack of appropriate documentation. Therefore, new users (i.e., students and instructors) should expect to spend several hours trying to find the resources and tools that work better for them. Finally, the authors recommended improving the user interface to make it less confusing, clarifying the online documentation [211], and increasing the availability of resources (e.g., VMs) in the platform.

SCIONLab is a network testbed specifically tailored for evaluating the SCION (Scalability, Control, and Isolation on Next-Generation Networks) network architecture. It allows researchers to experiment with SCION technology in a real-world environment. SCIONLab's strength lies in its specialization for SCION network architecture evaluation, providing researchers with a dedicated platform for this purpose. The scope of SCIONLab may be limited to evaluating SCION, and it may not offer the same level of versatility for other types of network experiments. The main characteristic of the SCION architecture is robustness and SCIONLab is the testbed that globally interconnects ASes to run experiments at a large scale. SCIONLab also provides the resources [218] that instruct users how to use and run experiments in the testbed. The tutorials include step-by-step instructions on running a SCION AS in SCIONLab and a list of interesting projects using the SCION infrastructure for communication. An *et al.* [219] evaluated the resilience of Multipath Routing mechanisms against attacks and failures using SCIONLab. They evaluated the performance in terms of latency and loss rate to demonstrate how multi-hop multi-path (MMP) routing can provide high availability in the presence of network attacks.

Colosseum is a large-scale wireless testbed that enables researchers to experiment with wireless communication technologies, including 5G and beyond. It offers a realistic environment for evaluating wireless systems and protocols. Colosseum focuses on large-scale wireless experiments, making it suitable for researchers interested in wireless communication technologies. The hardware and infrastructure requirements for Colosseum might be challenging for smaller research groups or institutions.

7.3. Production Networks as a Testbed

A vital aspect of the development process of network protocols is validation. Evaluating a new feature implies a trade-off between realism and cost. Testbeds can help to estimate the behavior of a new feature. However, real networking occurs on production networks (e.g., campus, enterprise, and government networks). Therefore, the research community leverages simulators, emulators, and testbeds to run experiments in a reproducible controlled environment. Although these tools are scalable and cost-efficient, they do not use real traffic. This limitation reduces the chances of new network features departing the research lab and being adopted in real-world networks [220].

7.3.1. P4Campus

Kim *et al.* [65] presented P4Campus, a P4-based system that allows network experimentation using campus

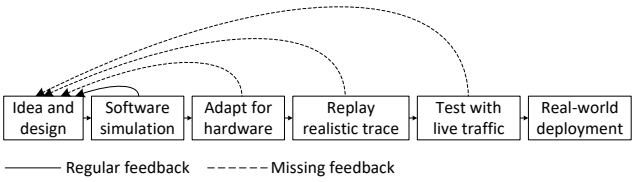

Figure 24: Research stages. Dashed lines represent the missing feedback loops that can improve the original research idea [65].

traffic. P4Campus facilitates testing network research ideas on real deployments so that researchers can run experiments in their network. The system uses network tapping devices, packet brokers, commodity, and programmable switches to evaluate research ideas on a production campus network. With these devices, P4Campus enables migrating from software-based simulation to implementation on hardware switches, capturing and replaying packet traces from their campus network, and running experiments against live production traffic. The system uses programmable switches for experimentation at line rate and a tool to anonymize personally identifiable information [221] and collect packet traces. Preliminary results show that the campus network is a rich source of research challenges that provides diverse traffic.

Figure 24 explains the conceptual research pipeline and how P4Campus fills the gap between a research laboratory and a production environment. The authors sustain that many of today's research ideas are prototypes evaluated using simulations and outdated packet traces. However, researchers find barriers when testing their ideas beyond an experimental prototype or laboratory. The authors argue that researchers should be able to evaluate their ideas in production environments (i.e., in the wild). Therefore, P4Campus is a promising approach to test research ideas using current traces or even live traffic without interfering with the regular operation of the production environment.

7.3.2. AmLight

Americas Lightpaths (AmLight) [66] is a project that aims at facilitating research and education between the United States and the nations of Latin America. AmLight operates international network links and connects research and education networks using a double-ring topology. The project's objectives are to improve operations efficiency and provide network programmability. The first objective refers to the coordination among institution that participates in AmLight. This coordination requires jointly managing the configurations that each institution has in its network and cooperating to troubleshoot network circuits. The second objective includes adding programmability to implement network functions that enable high tolerance, low delay, end-to-end visibility, and multipath routing. Applications such as in-network telemetry, big data, and video streaming can utilize network programmability to increase awareness about network conditions and optimally react to changes. Figure 25 shows the research and education institutions that are part of AmLight. The project includes international locations in places such as Miami (United States), Fortaleza (Brazil), São Paulo (Brazil), Porto Alegre (Brazil), Buenos Aires

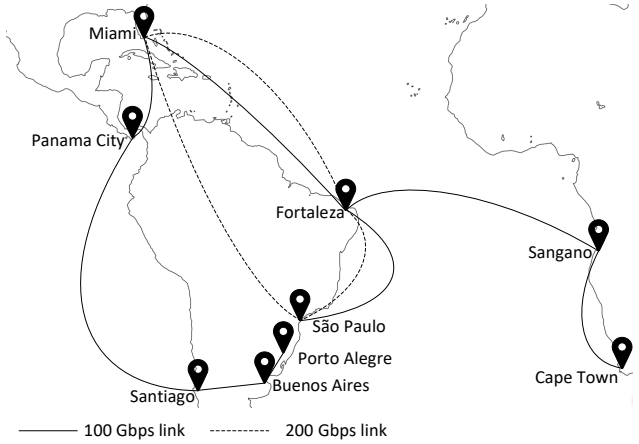

Figure 25: AmLight network topology [222].

(Argentina), Santiago (Chile), Panama City (Panama), Sagano (Angola), and Cape Town (South Africa). The goal of interconnecting these locations is to enable large data transfers (up to 600Gbps of upstream aggregate capacity). AmLight aims to isolate and detect faults and performance degradation of data transfers in long-haul networks with high latency.

AmLight deployed a geographical SDN/OpenFlow network that implements slicing, allowing isolated parallel testbeds in a production network [223]. The testbed provides sub-second network monitoring and performance evaluation by providing per-packet telemetry. This feature offers deeper visibility of network events to optimize resource utilization and to provide an additional security layer [224]. Measurements obtained from the production environment shows that AmLight can generate telemetry report and notify the network orchestrator in less than 200ms. Moreover, it can process 2,000,000 telemetry reports per second, enabling microburst detection, buffer utilization profiling, per-packet tracing, hash mismatches, packet jitter, and other network issues. More recently, AmLight announced its integration with FABRIC through a 100 Gbps dedicated link between the data center at Florida International University (FIU), and the Atlanta core node [225]. AmLight also integrates with ESnet and Internet2 research facilities. Cevik *et al.* [226] proposed a wide-are SDN controller prototype tested in AmLight. The system primarily focuses on enhancing software functions and configuration, creating a high-fidelity testing pipeline to add realism to the testbed, and establishing a continuous integration and deployment (CI/CD) environment. The authors reported that the proposed system increases software quality and development efficiency. This outcome can facilitate a more reliable migration to a production network deployment.

7.3.3. Entropy-based IDS

Crichigno *et al.* [67] presented a testbed to analyze anomalies in the traffic traversing a campus network. The testbed characterizes the Shannon entropy [227] on a per-flow basis to derive when an anomaly occurs. The testbed measured small/medium-sized flows traversing a campus network, focusing on the entropies of flow elements such as external IP address, external port, campus IP address, and campus port. The entropy indicates the random-

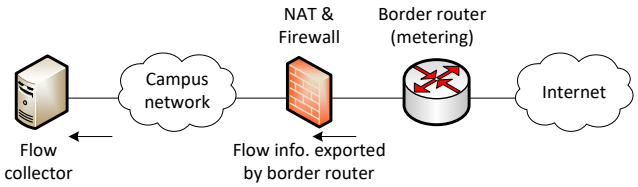

Figure 26: Topology used to characterize the flow entropy in a NATed network [67].

ness of a data set. The more random the data, the more entropy it contains. Their findings show that entropies widely vary in the course of a day. Therefore, the authors sustained that these variations can indicate anomalies in the network, which can be identified as attacks. Figure 26 shows the testbed architecture. The border router connects the campus network to the Internet Service Provider (ISP) / Internet. The NAT device translates private IP addresses to a single public IP address (campus IP). The campus network connects 15 buildings and different departments in a campus network. There are approximately 250 faculty/staff members, 1,500 students, 20 general-purpose computer laboratories, and faculty and staff offices. Many students access the Internet via WiFi using personal devices.

Results show that on a typical weekday, the external and campus ports' entropies may vary widely from below 0.2 to above 0.8 (considering a normalized entropy scale of 0-1). Similarly, the entropy of the campus IP address may vary from 0.1 to 0.4. Despite the wide range of values, findings indicate that building a granular (small time slots) entropy characterization of flow elements facilitates anomaly detection. Measurements show that specific attacks produce entropies that deviate from the expected patterns. They also show that the entropy of the 3-tuple {external IP, campus IP, campus port} is high and consistent over time, resembling the entropy of a uniform distribution variable. A deviation from this pattern is an encouraging anomaly indicator.

7.3.4. Comparison, Discussion, and Limitations

Testing research ideas in production networks reduces the prototype and the end product gap. With production networks, experimenters can know more realistically how their implementation can perform, detect limitations in the early stages, and apply the corrections. Examples of this methodology were demonstrated by Michel *et al.* [228], where they observed the performance of the Zoom videoconferencing application on a production network. The authors used the resources provided by P4Campus to analyze the packet traces of Zoom sessions. They found that although Zoom uses a proprietary protocol to encrypt packets, there are relevant unencrypted fields that can be used to group streams into meetings and identify peer-to-peer meetings. Additionally, they used the header fields to compute metrics such as media bit rates, frame rates, latency, and jitter. Similarly, Bai *et al.* [229] passively fingerprinted OSes using P4Campus. The motivation of this work is to inform network administrators about OS-specific vulnerabilities and administer security policies. Similarly, Kim *et al.* [230] proposed Meta4, a system a provides human-readable domain names to measure traffic, limit the rate, and identify IoT devices. The authors used P4Campus to collect the traces in an operational network.

7.4. On-demand Testbeds

This category describes the network emulators that combine container-based emulation, full virtualization, and network link emulation with real hardware to reproduce a hybrid environment. Using real hardware with VMs enhances the platforms' realism and facilitates creating and modifying network topologies efficiently. Network testbeds can be quickly and easily provisioned or set up to meet the needs of researchers or developers. These testbeds typically leverage virtualization and cloud computing technologies to enable researchers to access a wide range of networking resources, including routers, switches, servers, and storage, as well as various network topologies and configurations. These testbeds can also integrate VMs with real hardware to increase the realism of the experimentation testbed. On-demand testbeds are designed to provide researchers with greater flexibility and control over their experiments, allowing them to reserve predefined environments that meet their specific needs. This can be especially useful for researchers who need to consistently access the same pool of resources.

7.4.1. FABRIC

FABRIC (Adaptive Programmable Research Infrastructure for Computer Science and Science Applications) is a novel research infrastructure aimed to support large-scale research in networking, distributed computing, machine learning, and science applications [68]. Its main goal is to provide an experimentation testbed to explore methods and techniques to overcome the current architectural limitations of the Internet. One of these limitations is produced by the middleboxes in the network, which breaks the Internet architectural principles and complicates the process of troubleshooting connection problems and testing novel ideas. FABRIC will remove these limitations to allow researchers to study and prototype distributed architectures and applications. Moreover, FABRIC integrates existing testbeds such as GENI, CloudLab, and the Chameleon Cloud, with the possibility of expanding to more testbeds across borders.

The FABRIC architecture consists of distributed resources along national labs, campuses, and commercial collocation spaces. Each FABRIC site provides a large amount of computing (i.e., CPUs, GPUs, and FPGAs) and storage, interconnected by high-speed, dedicated optical links. Additionally, FABRIC integrates specialized testbeds in areas such as 5G, IoT, and cloud computing to create a rich environment for a wide range of experiments. Figure 27 shows the FABRIC core topology in the United States that comprises the following components:

- A Terabit network and several 100 Gbps links covering coast to coast. These sites incorporate FABRIC nodes interconnected by fiber links and collocation spaces from several organizations.
- Programmable edge nodes hosted on university campuses connected through regional network providers.

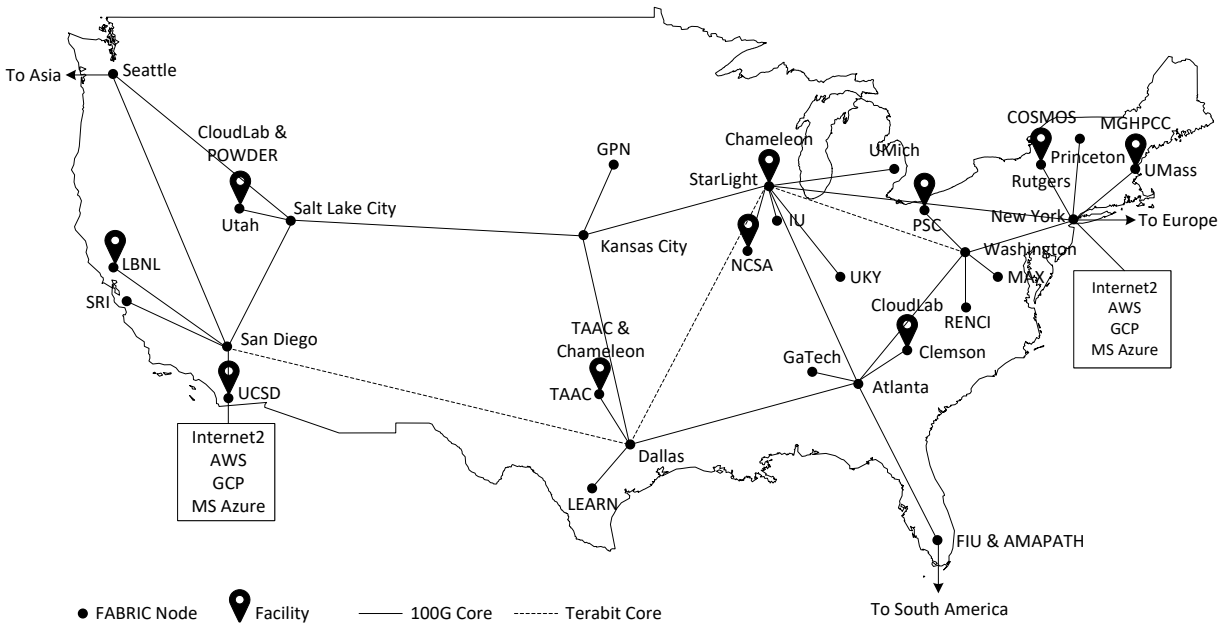

Figure 27: FABRIC core topology.

- A set of cyber resources provided by the community. These sites will connect to FABRIC nodes via an external link.

FABRIC nodes are interconnected via isolated optical links to guarantee predictable performance to experimenters. This characteristic will separate the production network from the FABRIC network. Therefore, an experimenter using FABRIC has access to a wide variety of public cloud platforms, computing centers, and scientific instruments. Moreover, an experimenter can connect custom hardware to a FABRIC node. Table 7 summarizes current organizations participating in FABRIC, their roles, and the type of experiments they support.

Experiments in FABRIC are conducted in a reservation-based portal. This portal provides the experimenter with the tools to reserve resources in the platform [231]. Experiments in FABRIC are described using JupyterHub notebooks [232] and, more recently a GUI that permits creating topologies by dragging and dropping components into a slice. In the context of the FABRIC testbed, a slice refers to the set of resources reserved for an experiment. The slice isolates the resources and performance from other slices to ensure reproducible experiments. A slice belongs to a single experiment that can be shared with multiple users. Experimenters can add resources to a slice depending on their availability and remove them in case they are not being used.

Figure 28 shows the components of a FABRIC node, which consists of programmable network elements (i.e., P4-programmable data planes, programmable NICs, and NetFPGAs), CPUs, GPUs, and different storage devices. A node can also support user-provisionable devices to integrate a wide variety of technology solutions and scientific instruments. A FABRIC node comprises a rack of high-performance servers (Dell 7525 [233]), programmable network interface cards, and kernel bypass technologies. The kernel bypass is performed by VMs and containers that support full-rate Data Plane Development Kits (DPDK) to access directly programmable NICs, FPGAs, and GPUs via PCI pass-through [234].

A FABRIC node has a GPS-disciplined clock at most of the sites, which allows experimenters to conduct accurate measurement experiments. This capability can be complemented with packet sampling and timestamping features available in the programmable NIC. Other capabilities of a FABRIC node include programmable port mirroring, smart Power Distribution Units (PDUs) to measure energy consumption, optical layer measurements, and monitoring tools to observe CPU, memory, and disk consumption as a function of time.

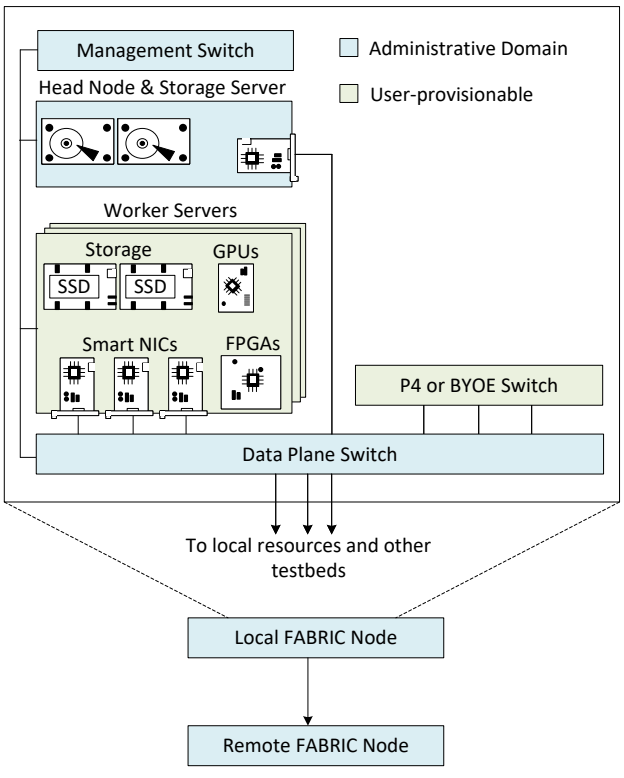

Figure 28: FABRIC Node ("Hank"). The node integrates compute (i.e., GPUs, CPUs, and FPGAs) and storage (i.e., SSDs and storage servers) into the path of fast packet flows.

Table 7: Facilities participating in FABRIC.

Organization	Role	Type of Experiments
The Texas Advanced Computing Center (TACC) [240]	• FABRIC node • Computing facilities • Private clouds	• Data analytics • Cybersecurity • Artificial intelligence • Software and applications
National Center for Supercomputing Applications (NCSA) [241]	• FABRIC node • Computing facilities	• HPC • Machine learning • Artificial intelligence • Cloud computing • Elastic Storage
Lawrence Berkeley National Laboratory (LBNL) [242]	• FABRIC node • Computing facilities • Science instruments	• High-speed networks • Cloud computing • SDN • Deep learning • Physics • Climate change
San Diego, Supercomputer Center (SDSC) [243]	• FABRIC node • Computing facilities	• Engineering • HPC • Big data • Cloud Computing • High-speed networks
CloudLab [10]	• FABRIC node • Computing facilities	• Cloud Computing • SDN • Software and applications
Chameleon [69]	• FABRIC node • Computing facilities	• Cloud Computing • SDN • Programmable data planes • Software and applications
Platform for Open Wireless Data-driven Experimental (POWDER) [244]	• Research facility	• Wireless Networks • SDN • 5G • RF monitoring
Aerial Experimentation and Research for Advanced Wireless (AERPAW) [245]	• Research facility	• Aerial experimentation • 5G • Wireless networks
PEERING [246]	• FABRIC node • Computing facilities	• Routing protocols

7.4.2. Chameleon

Chameleon [69] is a testbed system that supports high-performance hardware to run computer science experiments. The testbed is designed to be deeply reconfigurable by providing a wide range of capabilities for networking, distributed, and cybersecurity experiments. It also aims to reduce the entry barrier for experimenters by offering a user-friendly interface. Its user interface is based on OpenStack, which is an open-source cloud technology that proved to be practical for users and operators [38]. Chameleon gives experimenters access to hardware resources that support deep reconfigurability, orchestration, scalability, and isolation. Since its creation in 2015, Chameleon has supported numerous projects facilitating a broad set of experiment setups, data collection, and reproducibility. The experimentation testbed supports big data applications, Internet of Things (IoT), Software-defined Network (SDN) protocols, machine learning, and other distributed computing applications [235–239]. Chameleon also has the ability to connect ad hoc testbeds to larger pools of shared resources (e.g., data centers) so that experimenters can extend the capabilities of their local testbeds. The Chameleon testbed comprises two sites: one at the University of Chicago (UC) and the other at the Texas Advanced Computing Center (TACC). The goal of its hardware design is to balance scale and diversity so that the testbed can support HPC and Big Data experiments.

Experiments in Chameleon are specified using a GUI or a JupyterHub notebook, which makes it easier to share and reproduce experiments. An experimenter creates an account in Chameleon to have access to the environment. This process consists of signing up through an affiliated institution or through a Principal Investigator (PI) who is responsible for managing a project. The requirements to be eligible as a project PI are specified in [247]. After completing the signup process, an experimenter has access to his/her Chameleon profile, webinars, and shared community resources. Then, the user interacts with the testbed resources by creating a project. Once the experimenter participates in a project, he or she can reserve resources that are available in the Chameleon Infrastructure at TAAC (CHI@TACC) or Chameleon Infrastructure at the University of Chicago (CHI@UC). A dashboard displays an overview of the available resources. Based on that information, the experimenter can reserve a node that runs on a bare metal server. The user can specify the reservation period, which can last from one day to up to seven days. Longer reservations are subject to the organization's approval [248]. Once the node is reserved, the experimenter can launch an instance. An instance in the Chameleon testbed is referred to as an operating system running on a VM. This VM resides in a bare metal hypervisor and brings all the functionalities of the host Operating System (OS). The instance takes a few minutes to launch. Afterward, the experimenter is provided with the public and private Secure Shell (SSH) keys to access the instance and run experiments. The platform also provides access through a GUI that runs on OpenStack [142], an open standard cloud computing platform. The goal of this interface is to facilitate experimenters grouping the set of resources used in the project and taking snapshots of the VM states. A snapshot saves the current state of the virtual machine so that the experimenter can establish checkpoints to go back when testing features that might or might not produce an expected result[249].

The testbed facilities host twelve Haswell racks: two at UC and ten at TACC. These racks are equipped with computing and storage nodes interconnected using the InfiniBand (IB) standard [196] to facilitate high-speed communication between interconnected nodes. The facilities also have deployed three SkyLake (two at UC, one at TACC) racks and one CascadeLake rack at TACC. These racks contain SDN-enabled switches connected to 100Gbps networks to support experiments involving large flows. There are also smaller cluster nodes that support specialized experiments, including GPUs, FPGAs, low-power CPUs (i.e., ARM-based CPUs), memory hierarchy nodes, and arrays of solid and spinning storage. Chameleon facilitates users to allocate and configure based on the experiment's requirements, such as the model constraints, the node type, the resources available at a specific node, on-demand reservations, and advance reservations. The resources vary depending on the node type, and configurations are supported at the bare-metal level. The testbed also supports network stitching by implementing the Bring Your Own Controller (BYOC) abstraction, which enables attaching an external device to the testbed. The latter feature enables deeply programmable and SDN experimentation [250].

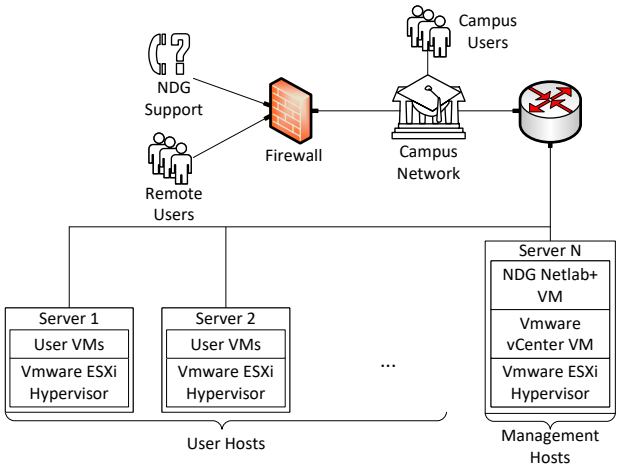

Figure 29: Netlab+ architecture deployed in a campus network [252].

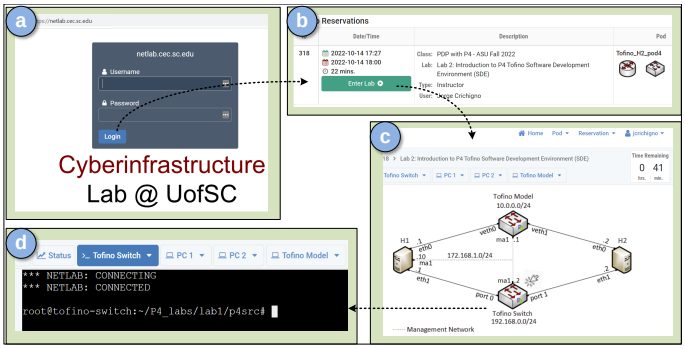

Figure 30: Accessing the cloud system. (a) A learner enters the system. (b) The learner reserves a virtual lab to conduct an experiment. (c) The learner enters the lab, and a scenario is recreated. The scenario consists of a pod of virtual devices. In this example, the learner clicked on the "Tofino Switch" device, which opened a window. (d) The learner operates the device via a CLI.

Chameleon provides fine-grained resource discovery for experimentation. This feature allows identifying the underlying hardware resources of a Chameleon node so that the experimenter can request access, reserve the node, and run experiments. Experimenters can consult the registry of resources via the resource discovery web interface or REST APIs [251]. This resource registry contains the hardware catalog with its corresponding availability status. Hardware resources in the Chameleon testbed include computing resources, InfiniBand support, GPUs, Storage, Field-Programmable Gate Arrays (FPGAs), low-power CPU servers, and ARM-based CPUs.

7.4.3. Netlab+

Netlab+ [70] is a network virtualization platform designed for educational, training, and experimentation purposes. It allows students and instructors to create and manage virtual networks, integrate hardware, and reproduce various networking scenarios using a web-based interface. Netlab+ integrates software and hardware resources to enable remote access to VMs, Docker containers, and real hardware. The platform allows users to interact with real equipment such as switches, routers, firewalls, and other appliances. The platform is supported by the Network Development Group (NDG) [163], a company that aims at training Information Technology (IT) students through hands-on experiences. The platform provides the elements to administer user accounts, create courses, manage pods and infrastructure, and develop content. Hardware and software resources in Netlab+ are organized as pods, a logical group of interconnected equipment that can be reserved as isolated instances using the platform's scheduler. Netlab+ also provides advanced capabilities such as network automation, virtual machine management, and remote access, enabling students to develop real-world skills in network administration, security, and automation. With Netlab+, instructors can easily create and deliver custom labs and assessments, monitor student progress, and provide feedback and support. Netlab+ is a powerful and flexible tool for network education, training, and experimentation.

The Netlab+ platform is a virtual appliance that relies on VMware vSphere [164] to host VMs and interact with real devices. The vSphere components used in Netlab+ are a bare-metal server, the VMware ESXi hypervisor [253], and the VMware VCenter [254], which is the appliance that centrally coordinates the system's components. Figure 29 illustrates the basic Netlab+ architecture that can be deployed in the data center of an academic institution. Netlab+ architecture consists of several key components that work together to provide a comprehensive network experimentation environment. The main components of the Netlab+ architecture are a central server, lab servers, client machines, a networking infrastructure, and a database. The central server is the heart of the Netlab+ system, responsible for managing user accounts, course content, and lab assignments. It also hosts the web interface and API that users access to manage and interact with the virtualized network environments. Lab servers are dedicated virtual machines that host virtualized network environments for students to work on. Each lab server can host multiple virtual machines and can be configured to simulate different types of network topologies, devices, and protocols. Client machines are the devices that students use to access the virtual lab environment hosted on the lab servers. These can be physical devices or virtual machines, and they connect to the lab servers via a web browser or remote desktop connection. The networking infrastructure is the backbone of the virtualized lab environment, consisting of virtual switches, routers, firewalls, and other devices that simulate the behavior of a real-world network. Netlab+ uses advanced network virtualization technologies such as VLANs and network address translation (NAT) to create complex and realistic network topologies. Finally, the Netlab+ system uses a database to store user account information, course content, lab assignments, and other relevant data. The database is hosted on the central server and can be backed up and restored as needed.

Netlab+ is a valuable platform for providing hands-on training in several IT disciplines, such as networking, cybersecurity, operating systems, network monitoring, cloud computing, and virtualization. The company also offers NDG Online, an on-demand solution where institutions can enroll learners into the courses available in the catalog [255] without deploying the whole infrastructure. Moreover, the platform allows the development of custom pods that can be used to create scenarios to

Table 8: Description of the P4 Tofino labs.

Labs	Description
Lab 1: Introduction to P4 and Tofino	The focus of this lab is to show the general lifecycle of programming, compiling, and running a P4 program on the Tofino switch.
Lab 2: : Introduction to P4 Tofino Software Development Environment	This lab describes the building blocks and the general structure of a P4 program corresponding to Tofino Native Architecture (TNA). The lab also shows how to compile, run, and track packets as they traverse the pipeline of the software switch (Tofino model).
Lab 3: Parser Implementation	This lab starts by describing how to define custom headers in a P4 program. It then explains how to implement a simple parser. The lab further shows how to track the parsing states of a packet inside the Tofino model.
Lab 4: Introduction to Match-Action Tables	This lab describes match-action tables and how to define them in a P4 program. It then explains the different types of matching that can be performed. The lab further shows how to track the misses/hits of a table key while a packet is processed in the Tofino model.
Lab 5: Populating and Managing Match-Action Tables at Runtime	This lab describes how to communicate with the switch's software interface (bfshell). Within bfshell, the lab shows a python-based tool (bfrt_python) used to program, manage, and populate the tables at runtime.
Lab 6: Checksum Recalculation and Packet Deparsing	This lab describes how to recompute the checksum of a header. The lab also describes how a P4 program performs deparsing to recompose the modified headers with the payload.
Lab 7: Inspecting the Resource Usage in the Tofino Switch	This lab describes the process of fetching the resource usage of a compiled P4 program. The lab also shows how the parser's states and the control blocks' tables are mapped to the hardware resources.

train students in specific topics. This feature also enables scaling-up experiments, such as the one conducted by Kfoury *et al.* [153]. In this paper, the authors created a pod consisting of a VM running Mininet. They defined the set of experiments to evaluate the performance of the Bottleneck Bandwidth and Round-trip Time version 2 (BBRv2) congestion control algorithm. The authors tested BBRv2 in several scenarios that measured the RTT Unfairness [256], Flow Completion Time (FCT), and coexistence with traditional congestion control algorithms such as CUBIC [257]. The experimenters created scripts to automate the tests. Then, they cloned the pod and ran the experiments in parallel using many VMs. This experimentation methodology aimed to aggregate the measurement results produced by each pod to improve the tests' accuracy. Organizing the pods in Netlab facilitated orchestrating large-scale tests, collecting the results, and troubleshooting the experiments. Gomez *et al.* [154] extended the experiments using a similar experimentation methodology in Netlab+ to evaluate the performance of BBRv2. This paper focused on the coexistence with competing flows as a function of time and the transient response of BBRv2 in the presence of new incoming flows. Similarly, Kfoury *et al.* [258] used Netlab+ as an interface to manage the resource to conduct experiments to test the impact of dynamically adjusting the buffer size in the flow completion time. The proposed system used passive taps to observe the traffic passing through a bottleneck link and derive the buffer size by measuring the RTT with a P4-programmable switch. Then, the system dynamically computes this time to set the buffer size in a legacy router. The prototype reported improvements in the FCT of short flows sharing the bottleneck link.

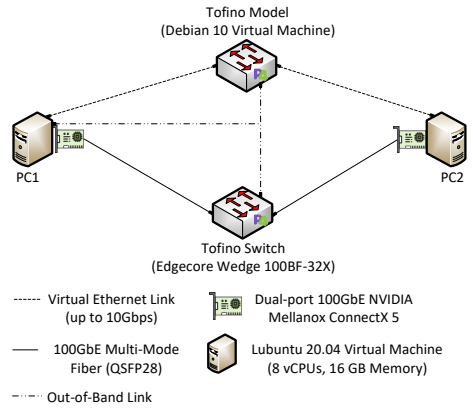

Figure 31: The Tofino pod components. PC1, PC2, and the Tofino model are VMs that communicate using virtual Ethernet links. The Tofino switch supports an Intel Tofino ASIC and is connected to PC1 and PC2 using a 100Gbps Ethernet link.

Netlab+ supports the integration of VMs with real hardware. This approach increases the flexibility of the type of topologies that can be deployed in the platform. The virtual labs on P4 developed at the University of South Carolina are an example of this integration. These labs integrate the Intel Tofino switch [259] to Netlab+ to instruct learners through hands-on lab activities. The lab activities explain P4 and data plane concepts following a step-by-step methodology. These labs introduce the fundamentals of P4 with a hardware switch and provide an environment to test new ideas. Netlab+ provides the tools to deploy pods, schedule lab reservations, and organize course content. Graduate students and researchers at USC interact with the cloud system using a web browser without installing additional software or acquiring the hardware resources to run the virtual labs.

Table 8 lists the labs included in the P4 Tofino library. The lab library contains seven hands-on activities introducing the fundamentals of P4. Labs 1 and 2 introduce the learner to the Software Development Environment (SDE) and the lifecycle of a P4 program. Labs 3 and 4 cover parser implementation and match-action tables. Lab 5 shows how to populate match-action tables from the control plane at runtime. Lab 6 explains how to perform the checksum calculation and define the deparser. Finally, in lab 7, the learner inspects the resource usage of a compiled P4 program to visualize how the parser states and control blocks are mapped to the hardware resources.

Figure 31 shows the components of the Tofino pod. These components include three VMs (i.e., PC1, PC2, and Tofino model), an Intel Tofino switch, Network Interface Controllers (NICs), physical links, and virtual links. PC1 and PC2 can communicate through the Tofino model or the Tofino switch. The Tofino model runs on a Linux Debian 10 VM, and the Tofino switch is an Edgecore Wedge 100BF-32X [260] that supports an Intel Tofino ASIC [261]. The Tofino model has logging enabled, which is unavailable in the hardware switch. This feature allows tracking the packet as it traverses across the switch's pipeline, facilitating debugging and troubleshooting P4 programs.

Figure 30 summarizes the steps to access the P4

Table 9: Testbeds comparison.

Ref.	Testbed	Description	Scale	Purpose	Education	Programmable data planes	Integrates w/ other testbeds	Cost
[11]	Emulab	Emulab is used to conduct scientifically rigorous experiments with high fidelity, repeatability, and measurement accuracy.	Medium	General	Supported	Not supported.	No	Free
[12]	GENI	GENI is a distributed virtual laboratory to run network science, services, and security experiments.	Large	General	Supported	Supported	Yes	Free
[61]	StarBED	StarBED allows the experimentation with L2 switch benchmarks, multicast protocols, performance analysis, wireless network emulation, mobile network experimentation, overlay networks, and experiments with real traffic.	Medium	General	Not supported.	Not supported.	Yes	Free
[62]	Grid'5000	Grid'5000 is a distributed, highly reconfigurable testbed that aims at large-scale computing experimentation. Grid'5000 provides a scientific tool for computer scientists similar to the large-scale instruments used by physicists, astronomers, and biologists.	Large	General	Supported	Not supported.	No	Free
[10]	CloudLab	CloudLab is a testbed for computing research that provides a large set of computing, storage, and networking resources.	Medium	Cloud Computing	Supported	Not supported.	No	Free
[9]	DETERLab	DETERLab is a public facility for medium-scale experimentation in cybersecurity. The platform is also used for instructing cybersecurity courses.	Medium	Cybersecurity	Supported	Not supported.	No	Free
[64]	SCIONLab	SCIONLab is a global testbed supporting experimentation in areas such as inter-domain routing that includes path-aware routing, routing policies, and security policies against DDoS attacks.	Medium	Routing	Not supported.	Not supported.	Yes	Free
[214]	Colosseum	Colloseum allows large-scale wireless experimentation. It provides state-of-the-art equipment to test novel wireless protocols.	Large	Wireless	Not supported.	Not supported.	Yes	Free
[65]	P4Campus	P4Campus is a P4-based system that allows network experimentation using campus traffic. P4Campus facilitates testing network research ideas on real deployments so that researchers can run experiments in their network.	Medium	General	Not supported.	Supported	No	Free
[66]	AmLight	AmLight's goal is to facilitate research and education between the United States and the nations of Latin America. AmLight operates international network links and connects research and education networks using a double-ring topology. The project's objectives are to improve operations efficiency and provide network programmability.	Large	General	Supported	Supported	Yes	Free
[67]	Entropy-based IDS	The testbed analyzes anomalies in the traffic traversing a campus network using the Shannon entropy.	Small	IDS	Not supported.	Not supported.	No	Free
[68]	FABRIC	FABRIC is a novel research infrastructure aimed to support large-scale research in networking, distributed computing, machine learning, and science. applications.	Large	Large-scale testing	Not supported.	Supported	Yes	Free
[69]	Chameleon	Chameleon is a testbed for running computing-intensive science experiments. It provides a broad array of state-of-the-art hardware to support deep reconfigurability, and enables experimentation at scale.	Medium	General	Supported	Supported	Yes	Free
[70]	Netlab+	Netlab+ is a system that integrates software and hardware resources to enable remote access to virtual machines, docker containers, and real hardware through a web browser.	< 100	Research & Education	Low	Supported	Yes	Paid

Tofino labs in the cloud system. A learner interacts with the cloud system using a web browser to access the system using a username and password. Then, the learner selects a lab activity and schedules a reservation. As a result, the lab scenario is recreated, and the learner can access the devices presented in Figure 31. Then, the learner operates the hardware switch using the Command-line Interface (CLI) and the VMs via a Graphical User Interface (GUI).

7.4.4. Comparison, Discussion, and Limitations

FABRIC is a network testbed designed to support cutting-edge research in networking and distributed computing. It aims to provide a programmable and highly configurable platform for experimenting with new network architectures and technologies. FABRIC focuses

on supporting advanced research in networking and distributed computing, offering researchers a flexible and customizable platform for testing innovative ideas. FABRIC's availability might be limited to certain research institutions or collaborations, affecting its accessibility for researchers outside those partnerships. FABRIC federates many affiliated testbeds as described in Table 7, enabling a wide range of experiments running nationwide scale [68]. However, FABRIC is not currently available in classrooms such as Emulab, DETERLab, GENI, CloudLab, and Chameleon. This limitation can be due to the early stages of the project and the purpose of the testbed, which is to facilitate the development of next-generation Internet applications and security countermeasures. FABRIC aims to provide researchers access to an everywhere-programmable infrastructure, including computing, storage, and networking connected with dedicated links. Moreover, fabric facilitates the creation of complex topologies by providing the FABLib library and JupyterHub environment to abstract network services and simplify the design of experiments. Ruth *et al.* [262] described the network service models available in FABRIC that help to build network experiments at a large scale. The network service models include layers two and three services, Virtual Private Networks (VPNs), port mirroring, and facility ports. These resources are located in the core of the wide-area network and can be connected using user-specified network services, which are unavailable in current testbeds. To extend FABRIC's capabilities, the testbed aims at integrating more testbeds outside the United States, which is part of the FABRIC Across Borders (FAB) [263] initiative. This extension includes testbeds in Europe, Asia, and South America, which aims to support collaboration in topics such as smart cities, weather and climate prediction, high energy physics, astronomy and cosmology, and computer science.

Chameleon is a cloud computing testbed that allows researchers to create custom virtualized environments for testing cloud-based applications and services. It offers a wide range of hardware configurations and software options. Chameleon's strength lies in its specialization for cloud computing experimentation, providing researchers with a platform to test and evaluate cloud-based applications in a controlled environment. Chameleon may have limitations in terms of network-specific experiments or support for advanced networking research, as it is primarily focused on cloud computing. Chameleon users can benefit from the contribution of a large development community and have the potential to contribute back to that community [38]. Chameleon also provides a suite of experimental digital resources compatible with the testbed. These resources include VM images, orchestration templates, and executable templates enabled by Jupyter Notebooks [232]. The latter capability enables sharing experiment templates to help experimenters easily replicate and modify testing scenarios. Cevik *et al.* [250] evaluated the limitations of the Chameleon testbed regarding its networking capabilities. The authors evaluated the testbed's behavior in an SDN context. Chameleon also provides up to 100Gbps connections using isolated layer two circuits transiting wide-area circuit providers, such as Internet2 AL2S [264] and ESnet On-Demand Secure Circuits and Advance Reservation System (OSCARS) [265]. This capability allows connecting to programmable switch hardware between Chameleon sites at the University of Chicago and Texas Advanced Computing Center (TACC). It also connects Chameleon hardware to other testbeds, facilities, or a user's home institution.

7.5. Summary and Lessons Learned

Network testbeds play a crucial role in advancing networking research and innovation by providing researchers with platforms to conduct real-world experiments and evaluate new technologies. Each testbed serves a specific purpose or research focus. For example, CloudLab specializes in cloud and edge computing experiments, FABRIC focuses on networking and distributed computing research, and P4Campus provides a platform for making improvements on production networks using the P4 language. These testbeds can also enable experimenters to use P4 programmable data planes to evaluate cybersecurity applications that operate at line rate [266]. AmLight facilitates high-performance networking research between the United States, Latin America, and Africa. Some testbeds, such as FABRIC and GENI, are collaborative efforts among research institutions. This highlights the importance of collaboration and sharing resources in network research to promote knowledge exchange and foster innovation. However, researchers should consider the accessibility of these resources and the approval process for gaining access to the testbeds. Emulab and GENI support a wide range of network experiments. On the other hand, specialized testbeds like SCIONLab and Chameleon are tailored to specific research areas. Researchers should balance their need for specialized features with the broader experimentation capabilities provided by versatile testbeds. Testbeds like StarBED and Colosseum are designed for large-scale experiments, providing realistic environments for evaluating network protocols under real-world conditions. However, these testbeds may require more significant hardware and infrastructure, which could pose challenges for smaller research groups or institutions.

Table 9 compares the surveyed testbed considering the scale, purpose, educational usage, data plane programmability, and integration. Most of the testbeds support medium to large-scale experimentation topologies. This feature allows experimenters to reproduce realistic results with a high chance of performing in real-world scenarios. It is also observed that not all the testbeds are compatible with existing network equipment, such as P4 programmable data planes and other SDN devices. Testing the coexistence between legacy and the new generation of network devices such as IoT, programmable data planes, firewalls, and other network appliances can enable the development of the next Internet architecture [68].

8. Challenges and Future Works

This section summarizes the challenges, describes the current initiatives, and proposes future works that can address the limitations of the surveyed works. Figure 32 depicts the challenges and their corresponding future works with references that focus on such limitations.

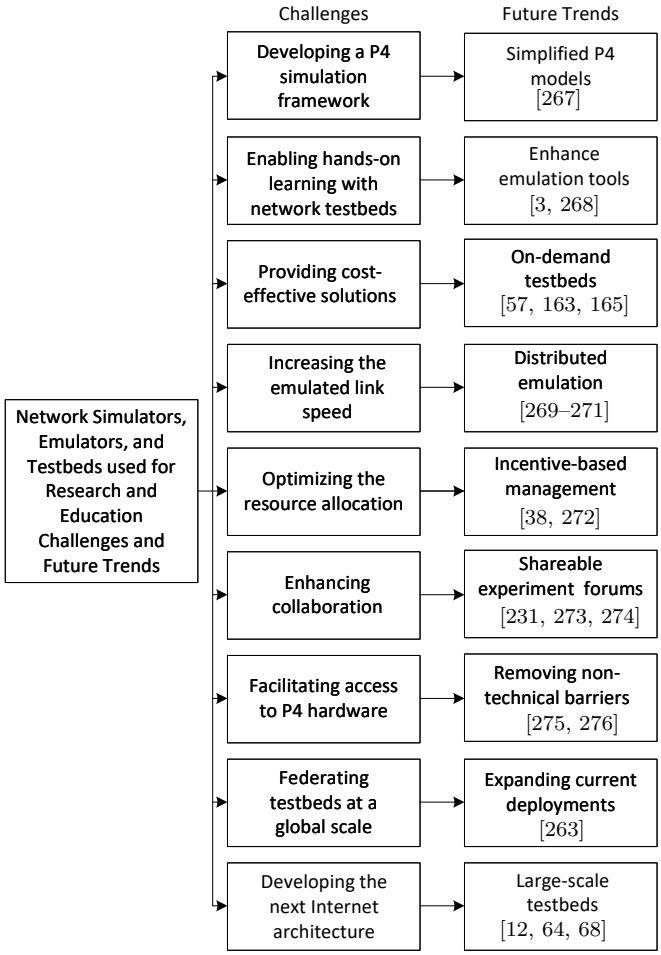

Figure 32: Challenges and future trends. The references represent examples of existing works and initiatives that tackle the corresponding future trends.

8.1. Developing a P4 simulation framework

Simulation models are scalable solutions that enable testing event-based and time-based experiments. P4 programmable data planes emerged as a promising technology to develop novel and improve existing systems. Developing a P4 simulation framework is a valuable endeavor for several reasons. Firstly, P4 is an increasingly popular language used for programming network devices, and having a simulation framework can help researchers and educators better understand how P4-based networks function and perform. By simulating network topologies and traffic patterns, learners can gain practical experience in designing, testing, and evaluating P4 programs in a controlled environment. Secondly, P4-based networks are becoming more complex, with a variety of devices and technologies involved. A simulation framework can help learners explore and experiment with complex P4-based networks without the need for expensive or specialized hardware. This can save time and resources while enabling learners to gain hands-on experience in designing and managing complex networks. Thirdly, a P4 simulation framework can facilitate collaboration and knowledge sharing among researchers and educators. By providing a common platform for simulating P4-based networks, learners can share code, data, and best practices, enabling faster and more efficient learning and experimentation. Finally, a P4 simulation framework can help bridge the gap between theory and practice in networking education. Learners can use the simulation framework to test and validate concepts they have learned in lectures and textbooks, gaining practical experience that reinforces their understanding of networking principles and technologies. Developing a P4 simulation framework can benefit learners, researchers, and educators alike by providing a valuable tool for exploring, experimenting, and collaborating on P4-based networking projects.

Moreover, test event-driven models with P4, something that is currently unavailable. With P4 switches, experimenters can efficiently use the resources available in the device (e.g., CPU and memory) by only programming the necessary features demanded by the application. This feature can enable interesting research topics that involve IoT devices, where analyzing the variables such as performance, energy consumption, memory, and CPU usage are metrics that the surveyed papers look to improve.

Current and Future Initiatives. Fan *et al.* [267] proposed ns-4, an ns-3 module for P4 programmable data planes. Ns-4 is a P4-driven network simulator, the first work in this field. The authors addressed issues such as eliminating the learning curve that current P4 emulators require, the portability issue that P4 experiences when porting the code to real hardware, and the lack of a reliable tool to validate systems developed using the P4 language. The authors demonstrated the effectiveness and convenience of NS4 by simulating a data center network and comparing performance metrics with the one obtained in Mininet. Future works should consider extending ns-4 and creating libraries that simulate different types of P4-programmable data plane pipelines such as the simple switch, V1Model [277], the Tofino Native Architecture (TNA) [278], and others. This P4 simulation model can reduce the time an experimenter spends setting the environment and focus more on the data plane behavior they want to describe with P4.

8.2. Enabling hands-on learning with network testbeds

Enabling hands-on learning network testbeds provides an opportunity for students and researchers to gain practical experience in network design, configuration, and operation. Hands-on learning is a powerful educational tool that allows learners to engage with complex concepts in a meaningful way and develop critical thinking and problem-solving skills. By using network testbeds, students can learn by doing, experimenting with real-world scenarios, and applying theoretical knowledge to practical situations. This approach to learning can also enhance students' ability to work collaboratively and communicate effectively as they work together to design and configure network topologies. In addition to benefiting students, network testbeds can also be used by researchers to develop and evaluate new networking technologies in a controlled and repeatable environment. Ultimately, enabling hands-on learning network testbeds can help bridge the gap between theory and practice and prepare students and researchers for the challenges of the rapidly-evolving field of networking.

Teaching networking concepts strongly relies on a suitable environment to help students apply the acquired knowledge. Simulation tools such as Packet Tracer

are effective in systematically training IT professionals. However, the proprietary nature, limited performance realism, lack of customization, and the inability to run actual code reduce the type of experiments the platform can support. On the other hand, using testbeds to teach networking courses is not flexible as they require having an agreement with the testbed management. Moreover, there is an overhead that the learner incurs when getting started with network testbeds.

Current and Future Initiatives. There are some efforts to solve most of these issues. Holterbach *et al.* [3] developed an open platform to teach various components of the Internet infrastructure works practically. The platform is called the mini-Internet project [268] and is used to teach networking. The platform enables students to operate and interconnect their Autonomous Systems (ASes), aiming at reproducing Internet-wide connectivity. Each Autonomous System (AS) exchanges routing information via Border Gateway Protocol (BGP) via Internet eXchange Points (IXPs) maintained by the instructors. The platform differs from traditional teaching courses by turning students into operators so that they can work on topics such as network design, network configuration, network monitoring, and network debugging. The mini-Internet project relies on open networking tools such as Open vSwitches [37] and Free Range Routing (FRR) [279] to provide L2 and L3 protocols (e.g., ARP, BGP, and, Open Shortest Path First (OSPF)). Additionally, the system supports L2 Virtual Private Network (VPN) servers and provides a connectivity matrix for the students to check if there is connectivity among ASes. The system scales up to 60 ASes per server supporting around 100 students considering that each group comprises 2-3 students. The authors believe that by using the mini-Internet project, students can have an early experience as operators and learn how to avoid mistakes.

Although the mini-Internet project is a scalable solution for networking courses, it relies on emulation, which depends on the available resources (i.e., CPU and memory) of the host. However, with the emergence of next-generation testbeds such as FABRIC, learners can reproduce experimentation scenarios using a script. This means that instead of manually configuring network topologies and experimenting with different configurations, learners can write scripts that automate these tasks. This not only saves time and effort but also enables learners to reproduce experiments more accurately and consistently. By automating repetitive tasks and enabling accurate and consistent reproduction of experiments, scripts can help learners focus on the design of experiments and the analysis of results, leading to a more efficient and effective learning experience.

8.3. Providing cost-effective solutions

Creating a suite of open-source resources to teach networking courses can be time-consuming and requires a dedicated staff to manage updates, maintain the platforms, and increase or reduce the number of resources to fulfill the student demands. Moreover, the cost of commercial-grade networking equipment leads to high student-to-equipment ratios, which limits hands-on time per student. In many parts of the world, these costs are prohibitive, yet the skills are essential for the growth of the local IT economy. Practical exposure to networking improves skills and enhances knowledge retention.

Current and Future Initiatives. The Network Development Group (NDG) [163] supports and maintains an academic cloud [57, 165] that aims at training IT professionals in various topics that include networking, P4-programmable data planes, routing protocols, SDN, Intrusion Detection System (IDS), monitoring tools, cloud concepts, certification training, and others. The academic cloud inherits all the benefits of cloud technology, such as saving deployment costs, offloading maintenance tasks, paying only for the utilized resources, accessing the resources from anywhere at any time, and other benefits. Therefore, the NDG academic cloud is an efficient solution for institutions that cannot afford an on-premises deployment and want to scale up the number of students they serve. Moreover, the NDG academic cloud also can integrate with real hardware to not only deliver training courses but giving experimenters access to equipment that is not widely available.

8.4. Increasing the emulated link speed

Mininet is one of the most popular network emulators that can reproduce SDN experiments and integrate with more specialized devices by supporting Docker containers (i.e., Containernet). Many experimenters and students use Mininet due to its simplicity. Mininet can work directly on the experimenter's laptop. Moreover, its Python APIs provide simplicity and flexibility to reproduce complex scenarios for basic tests quickly. However, Mininet was not designed for resource-intensive experiments; one computer is insufficient to support large-scale experiments and execute tasks properly. Heller [1] measured the network invariants to demonstrate the limits where experiments running on Mininet start differing from what is obtained with real hardware. The author created microbenchmarks to test the OS accuracy to emulate the behavior of a link for both TCP and UDP flows. Results show that the accuracy differs as the link rate approaches 1Gbps demonstrating that errors increase as the system runs near the CPU limits. The author highlights that the experiment fidelity decreases as CPU utilization increases.

Current and Future Initiatives. Di Lena *et al.* [269, 270] proposed Distrinet, a solution that distributes Mininet into several nodes as a function of the experiment's load. The system was implemented in Amazon Web Services (AWS) by orchestrating Elastic Cloud 2 (EC2) instances. Distrinet design supports intensive computing and I/O experiments on various experimental infrastructures. Distrinet determines the number of machines that can provide realism to an experiment and deploy them using an optimization technique. The platform can transparently provide the resources, so experimenters do not have to know the infrastructure details. Rizzo *et al.* [271] proposed an optimized link network emulator to deliver high performance with realism. Results show that the proposed system achieves high accuracy for links over 40 Gbps, which is higher than

the ones obtained with NetEm and Dummynet. Future initiatives should also consider adding link scheduling and classification features that operate at high-speed links. Future initiatives should consider integrating improved network link emulators with container-based emulators. Although it is resource intensive to emulate high-speed links with a single host, distributed solutions can balance the overheads produced by the requirements of container-based emulators.

8.5. Optimizing the resource allocation

The type of experiments supported by network testbeds relies on the type and amount of resources available in the testbed. A typical procedure in testbeds allows users to describe the hardware they need. For instance, commercial platforms such as Amazon Web Services (AWS) and Microsoft Azure offer a variety of instances that are sometimes vaguely described properties (e.g., high I/O bandwidth). Research-oriented testbeds allow experimenters to choose a specific hardware type to guarantee consistency in the kind of resources they provide and their capabilities. However, the number of resources to conduct experiments is not always available. If they are, these resources are usually underutilized [38]. Therefore, it is challenging for the experimenter and the testbed administrator to balance the resource allocation mechanism to utilize them better. The resource demand usually fluctuates as a function of the research topics in the research community's interest.

Current and Future Initiatives.. Keahey *et al.* [38, 272] evaluated usage and user experience in the Chameleon testbed after five years of operations. They considered how the testbed could support the broadest possible set of experiments for the most extensive possible collection of the experimenters to assess the capabilities of the testbed. The authors pointed out that when resources are limited, it is essential to balance scale and support for a broad range of resources to maximize the diversity in the experiments supported by the platform. To derive this, the authors observed the node availability against the percentage of time that it was available. Starting from the overall time since resource installation and during the busiest and least busy months as measured by highest and least utilization. Their observation noted that newly installed hardware is usually the one that is less utilized at the beginning. They also said that resources are busier close to conferences deadlines or teaching/workshop use. From these observations, the authors remarked on the importance of adapting the type and amount of resources to the community's demands. This demand changes based on the research interest, and it fluctuates periodically.

8.6. Enhancing collaboration

There is no standardized procedure to define how users will conduct testbed experiments. Each testbed has its way of explaining how resources are accessed and reserved and the way results are collected. Since deploying a testbed consists of an incremental process, the complexity grows as a function of the heterogeneity of the supported hardware. Therefore, documenting and maintaining appropriate documentation relies on limited human resources. Mirkovic and Pusey [280] collected and analyzed the user experiences on network testbeds by surveying a representative amount of users. Their findings show that testbeds have scarce human resources to develop documentation or support users one-on-one. Thus, there are limitations that new experimenters can encounter when getting started on using testbeds. This limitation can be worsened by user inexperience and user support deficiencies, resulting in research or learning obstacles.

Current and Future Initiatives. Current initiatives include the FABRIC and Chameleon documentation, and forum [231, 273]. The documentation provided by this testbed shows the necessary steps new experimenters should follow to get started with the platform and conduct experiments specific to the topic they want to explore. Moreover, these platforms aim at building a community by enabling users to post questions in the community forum. Experimenters can share their experiences and issues encountered using the platform in this forum. The testbed operators are also part of the discussions by addressing experimenters' needs in the form of a new release [274].

8.7. Facilitating access to P4 hardware

Current testbeds offer limited options regarding P4-programmable data planes. In the FABRIC testbed, the experimenter can use a virtualized version of a P4-programmable switch (i.e., a BMv2 switch [281]), which is mainly used for prototyping but lacks performance realism under heavy loads. Hardware-based P4 programmable data planes are essential resources for testing novel applications as they provide high performance, making adopting them in production networks attractive. However, there are barriers that testbed developers face when planning to adopt hardware-based P4-programmable data planes. A significant issue is the manufacturers' legal agreement to use their products. This includes the inability to make the Software Development Environment (SDE) available for experimenters to compile and load a P4 program into a hardware switch. Another limitation is the configuration overhead that hardware-based P4 programmable data planes carry. These devices come as white boxes that require testbed developers to install a Network Operating System (NOS), the SDE dependencies, and the SDE itself. Therefore, this could be a cumbersome limitation for a new entrant that incurs longer deployment times. Lastly, the costs and availability of the devices can also limit access to this technology.

Current and Future Initiatives. The FABRIC testbed is currently planning to offer hardware-based P4-programmable data planes in future releases [275]. Although the legal barrier is still a limitation, an intermediate solution consists of making available the P4 devices for all experimenters and dividing them into two groups. The first group will consist of experimenters that can run precompiled P4 programs. The second group will comprise the experimenters who fulfill the legal requirements and can use the SDE for creating, compiling, and running their P4 programs. Current initiatives

include works such as the one presented by Chung *et al.* [276], where they proposed P4MT, a control mechanism to enable multitenancy on a P4-programmable switch. The goal of P4MT is to enhance collaboration among institutions that possess P4-programmable switches. Evaluations show that P4MT consumes small resources and causes a negligible increment in data/control plane latency. Future initiatives must take the necessary action to remove the legal barriers so that the P4-based hardware and software are aligned with the open-source vision on which the language was founded.

8.8. Federating testbeds at a global scale

Conducting research with science workflows typically require the federation of geographically distributed science instruments and computing systems. These experiments produce a large amount of data and demand complex computation. Accessing these instruments also requires several months to set up and continued coordination among multiple organizations to operate and maintain. Therefore, enabling high-speed access to these resources is a challenge and an opportunity for testbed developers and the research community to enhance collaboration.

Current and Future Initiatives. FABRIC Across Borders (FAB) [263] is an extension of the FABRIC testbed connecting the core network in the North American infrastructure to four nodes in Asia, Europe, and South America. This initiative aims at creating the networks needed to move large amounts of data across oceans and time zones seamlessly and securely. The project will enable international collaboration to support federated research. FAB fulfills science needs in fields that will form the foundations of the next generation of Internet applications. It will offer the mechanisms to handle and share massive amounts of data generated by powerful new scientific instruments around the globe. FAB's use cases include smart cities, weather, physics, space observation, and computer science.

8.9. Developing the next Internet architecture

The current Internet architecture comprises intrusive middleboxes (e.g., firewalls, NAT, IDS, and others) that violate the end-to-end principle of the IP architecture and makes it difficult to diagnose problems for scientific research. The science DMZ [13] was proposed as a solution to overcome the barriers imposed by providers. The introduction of P4-programmable data planes opened a new horizon to experiment with more comprehensive network monitoring applications [282] that can facilitate finding the root cause of the problem. However, the science DMZ architecture represents only a small portion of the current Internet architecture which does not provide a realistic environment for researchers to test large-scale distributed applications. Therefore, there is a need for a large-scale, friction-free environment to test advanced protocols and novel distributed applications that involve handling massive data sets and having full visibility of the network performance.

Current and Future Initiatives. Testbeds such as ScionLab [64], GENI [12], and FABRIC [68] aim at enabling a large environment to test the next Internet architecture. These testbeds provide large storage, compute resources, and programmability in the network nodes, diverging from the current model where the end nodes are the ones that perform the heavy computation. They also provide the environment to test cybersecurity applications not considered in the existing Internet architecture. Adding programmability into the network enables an efficient way to respond to security events more rapidly and closer to their sources before the issue escalates and affects the end users. Paradigms such as path-aware networking and multipath communications are promising approaches to address new security challenges to drive the development of the next generation of applications.

9. Conclusion

This survey explored the network simulators, emulators, and testbeds used for research and education. The paper presents a taxonomy and summarizes works that discuss the experimentation platforms for conducting research and the tools employed in academia to deliver education on different computing areas. This survey aims at providing an overview of the main features of the surveyed experimentation platforms. With the information presented in this survey, experimenters can gain insights into selecting a platform aligned with their research needs. The findings in this survey highlight that the experimentation platform selection depends on the type of experiments, resource availability, and the realism the platform provides.

10. Acknowledgement

This work was supported by the U.S. National Science Foundation under grant number 2118311, funded by the Office of Advanced Cyberinfrastructure (OAC).

Table 10: Abbreviations used in this survey.

Abbreviation	Term
3GPP	3^{rd} Generation Partnership Project
5G	Fifth Generation
AODV	Ad-hoc On-Demand Distance Vector
API	Application Programming Interface
AQM	Active Queue Management
ARP	Address Resolution Protocol
AS	Autonomous System
ASIC	Application-specific Integrated Circuit
ATA	Advanced Technology Attachment
AWS	Amazon Web Services
BBR	Bottleneck Bandwidth and Round-trip Time
BGP	Border Gateway Protocol
BMv2	Behavioral Model Version 2
BYOD	Bring Your Own Device
CBWFQ	Class-Based Weighted Fair Queuing
CLI	Command-line Interface
CML	Cisco Modeling Labs
CN	Core Networks
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DASH	Dynamic Adaptive Streaming over HTTP
DMA	Direct Memory Access
DMZ	Demilitarized Zone

Abbreviation	Term
DNS	Domain Name Server
DOE	Department of Energy
DPDK	Data Plane Development Kit
DRAM	Dynamic Random-Access Memory
DYMO	Dynamic MANET on-demand Routing Protocol
EC2	Elastic Cloud 2
EPC	Evolved Packet Core
FIFO	First-In First-Out
FPGA	Field-Programmable Gate Array
FQ-CoDel	Fair Queueing Controlled Delay
FRR	Free Range Routing
GCP	Google Compute Platform
GPU	Graphics Processing Unit
HDS	HTTP Dynamic Streaming
HLS	HTTPS Live Streaming
ICMP	Internet Control Message Protocol
ICS	Industrial Control System
IDS	Intrusion Detection System
IERP	Interzone Routing Protocol
IETF	Internet Engineering Task Force
INT	In-band Network Telemetry
IoT	Internet of Things
IP	Internet Protocol
ISD	Isolation Domains
ISP	Internet Service Provider
JiST	Java in Simulation Time
MFA	Multi-Factor Authentication
MPEG	Moving Picture Experts Group
MQTT	Message Queueing Telemetry Transport
LLQ	Low Latency Queuing
LTE	Long Term Evolution
MANET	Ad Hoc Networks
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NACK	Negative Acknowledgement
NetEm	Network Emulator
NetFPGA	Network Field Programmable Gate Array
NF	Network Functions
NFC	Near Field Communication
NFV	Network Functions Virtualization
NGC	Next-Generation Core
NIC	Network Interface Controller
NOS	Network Operating System
NPU	Network Processing Unit
NSF	National Science Foundation
OAI	OpenAirInterface
OAC	Office of Advanced Cyberinfrastructure
OLSR	Optimized Link State Routing Protocol
OS	Operating System
OTCL	Object-oriented
OVA	Open Virtual Appliance
OVS	Open Virtual Switch
P4	Programming Protocol-independent Packet Processors
PDC	Parallel and Distributed Computing
PI	Principal Investigator
PIE	Proportional-Integral controller Enhanced
PMEM	Persistent Memory
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RDMA	Remote Direct Memory Access
R&E	Research and Education
RoCE	RDMA over Converged Ethernet
RF	Radio Frequency
RFC	Request For Comments
RTT	Round-Trip Time
SCSI	Small Computer System Interface
SDE	Software Development Environment
SDN	Software-Defined Networking
SFC	Service Function Chains
SLA	Service Level Agreement
SRAM	Static Random-access Memory
SR-IOV	Single-Root Input/Output Virtualization
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TBF	Token Bucket Filter
TCAM	Ternary Content Addressable Memory
TLS	Transport Layer Security
ToR	Top of Rack
UDP	User Datagram Protcol
VANET	Vehicular Ad-hoc Network
VPN	Virtual Private Network
VSA	Vector Symbolic Architecture
VIRL	Virtual Internet Routing Lab
WAN	Wide Area Network
WFQ	Weighted Fair Queuing
WSN	Wireless Sensor Network

References

[1] H. Brandon, *Reproducible network research with high-fidelity emulation*. Stanford University, 2013.

[2] M. Prvan and J. Ožegović, "Methods in teaching computer networks: a literature review," *ACM Transactions on Computing Education (TOCE)*, 2020.

[3] T. Holterbach, T. Bühler, T. Rellstab, and L. Vanbever, "An open platform to teach how the Internet practically works," *ACM SIGCOMM Computer Communication Review*.

[4] R. Birkner, *Improving Network Understanding*. PhD thesis, ETH Zurich, 2021.

[5] Bussiness Insider, "Amazon's one hour of downtime on Prime Day may have cost it up to $100 million in lost sales." [Online]. Available: `https://tinyurl.com/3vy789hk`, Accessed on 06-12-2022.

[6] H. S. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatama, and K. J. Eliazar, "Why does the cloud stop computing? lessons from hundreds of service outages," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*, 2016.

[7] T. Tsvetanov and S. Slaria, "The effect of the Colonial Pipeline shutdown on gasoline prices," *Economics Letters*, 2021.

[8] S. Rampfl, "Network simulation and its limitations," in *Proceeding zum seminar future internet (FI), Innovative Internet Technologien und Mobilkommunikation (IITM) und autonomous communication networks (ACN)*, 2013.

[9] T. D. Project, "DETERLAB." [Online]. Available: `https://deter-project.org/about_deterlab`, Accessed on 03-21-2022.

[10] R. Ricci, E. Eide, and C. Team, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," *The magazine of USENIX & SAGE*, 2014.

[11] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *ACM SIGOPS Operating Systems Review*, 2002.

[12] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, 2014.

[13] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science DMZ," *IEEE Communications Surveys & Tutorials*, 2018.

[14] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A survey on large-scale software defined networking (sdn) testbeds: Approaches and challenges," *IEEE Communications Surveys & Tutorials*, 2016.

[15] P.-W. Tsai, F. Piccialli, C.-W. Tsai, M.-Y. Luo, and C.-S. Yang, "Control frameworks in network emulation testbeds: A survey," *Journal of Computational Science*.

[16] N. Chouliaras, G. Kittes, I. Kantzavelou, L. Maglaras, G. Pantziou, and M. A. Ferrag, "Cyber ranges and testbeds for education, training, and research," *Applied Sciences*, 2021.

[17] L. Nussbaum, "Testbeds support for reproducible research," in *Proceedings of the reproducibility workshop*, 2017.

[18] M. Bakni, Y. Cardinale, and L. Moreno, "Experiences on evaluating network simulators: A methodological approach," *journal of communications*, 2019.

[19] A. R. Khan, S. M. Bilal, and M. Othman, "A performance comparison of open source network simulators for wireless networks," in *2012 IEEE international conference on control system, computing and engineering*, 2012.

[20] L. Nussbaum and O. Richard, "A comparative study of network link emulators," in *Communications and Networking Simulation Symposium (CNS'09)*, 2009.

[21] E. Lochin, T. Perennou, and L. Dairaine, "When should i use network emulation?," *annals of telecommunications-annales des télécommunications*, 2012.

[22] B. Martini, M. Gharbaoui, D. Adami, P. Castoldi, and S. Giordano, "Experimenting SDN and cloud orchestration in virtualized testing facilities: performance results and comparison," *IEEE Transactions on Network and Service Management*, 2019.

[23] T. Issariyakul and E. Hossain, "Introduction to network simulator 2 (NS2)," in *Introduction to network simulator NS2*, 2009.

[24] The Defense Advanced Research Projects Agency (DARPA), "The network simulator - ns-2." [Online]. Available: `https://www.isi.edu/nsnam/ns/`, Accessed on 01-17-2022.

[25] Cisco Network Academy, "Packet tracer." [Online]. Avail-

able: https://www.netacad.com/courses/packet-tracer, Accessed on 01-17-2022.
[26] Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, "ns3 - network simulator." [Online]. Available: https://www.nsnam.org/, Accessed on 01-17-2022.
[27] Galaxy Technologies, LLC, "Graphical network simulator (GNS3)." [Online]. Available: https://www.gns3.com/, Accessed on 01-17-2022.
[28] Pranav Viswanathan and Shashikant Suman, "Netsim: Network simulator." [Online]. Available: https://www.tetcos.com/index.html#, Accessed on 01-17-2022.
[29] Riverbed, "Opnet modeler: Optimized network engineering tools riverbed modeler." [Online]. Available: https://www.riverbed.com/en-gb/products/network-performance-management.html, Accessed on 01-17-2022.
[30] J. Crichigno, N. Ghani, J. Khoury, W. Shu, M. Wu, "Dynamic routing optimization in WDM networks," in *Proceedings of the 2010 IEEE 2010 IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.
[31] J. Crichigno, W. Shu, M. Wu, "Throughput optimization and traffic engineering in wdm networks considering multiple metrics," in *Proceedings of the 2010 IEEE International Conference on Communications (ICC)*, 2010.
[32] S. C. Ergen, "ZigBee/IEEE 802.15. 4 summary," *UC Berkeley, September*, 2004.
[33] Z-Wave Alliance, "Z-Wave." [Online]. Available: https://www.z-wave.com/, Accessed on 09-26-2022.
[34] Square Inc., "Near Field Communication (NFC)." [Online]. Available: http://nearfieldcommunication.org/, Accessed on 09-26-2022.
[35] Z. Lu and H. Yang, *Unlocking the power of OPNET modeler*. Cambridge University Press, 2012.
[36] VMware, "Understanding full virtualization, paravirtualization, and hardware assist." [Online]. Available: https://tinyurl.com/FSVirtualization, Accessed on 01-17-2022.
[37] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, *et al.*, "The design and implementation of open vswitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI15)*, 2015.
[38] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, *et al.*, "Lessons learned from the chameleon testbed," in *2020 USENIX Annual Technical Conference*, 2020.
[39] M. Stoller, R. Hibler, L. Ricci, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the emulab network testbed," in *USENIX annual technical conference, Boston, MA*, 2008.
[40] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008.
[41] OpenSim Limited, "Omnet++." [Online]. Available: https://omnetpp.org/, Accessed on 07-07-2022.
[42] X. Chang, "Network simulations with OPNET," in *WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'(Cat. No. 99CH37038)*, 1999.
[43] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation PADS'98 (Cat. No. 98TB100233)*, 1998.
[44] Scalable Network Technologies, "Qualnet network simulation software." [Online]. Available: https://tinyurl.com/musb5xyy, Accessed on 07-21-2022.
[45] F. Österlind, *A sensor network simulator for the Contiki OS*. Swedish Institute of Computer Science, 2006.
[46] Cisco Networking Academy, "Cisco packet tracer." [Online]. Available: https://tinyurl.com/468bdvxv, Accessed on 07-21-2022.
[47] 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*, pp. 1–6, 2010.
[48] M. Peuster, J. Kampmeyer, and H. Karl, "Containernet 2.0: A rapid prototyping platform for hybrid service function chains," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018.
[49] L. Veltri, L. Davoli, R. Pecori, A. Vannucci, and F. Zanichelli, "NEMO: A flexible and highly scalable network EMulatOr," *SoftwareX*, 2019.
[50] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real-time network emulator," in *MILCOM 2008-2008 IEEE Military Communications Conference*, 2008.
[51] Cisco, "Cisco modeling labs 1.0 user guide." [Online]. Available: https://tinyurl.com/538drxkm, Accessed on 10-06-2022.
[52] Z. Puljiz and M. Mikuc, "IMUNES based distributed network emulator," in *2006 International Conference on Software in Telecommunications and Computer Networks*, 2006.
[53] W. Du, "SEED: hands-on lab exercises for computer security education," *IEEE Security & Privacy*, 2011.
[54] S. Hemminger *et al.*, "Network emulation with NetEm," in *Linux conf au*, 2005.
[55] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, 1997.
[56] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for HTTP," in *Usenix annual technical conference*, 2015.
[57] J. Crichigno, E. Kfoury, K. Caudle, and P. Crump, "A distributed academic cloud and virtual laboratories for information technology education and research," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 2021.
[58] Amazon, "Amazon Web Services (AWS)." [Online]. Available: https://aws.amazon.com/, Accessed on 04-13-2023.
[59] Google, "Google Computing Engine (GCE)." [Online]. Available: https://tinyurl.com/3juuyb9z, Accessed on 04-13-2023.
[60] Microsoft, "Microsoft Azure." [Online]. Available: https://tinyurl.com/2p8j9yvr, Accessed on 04-13-2023.
[61] T. Miyachi, K.-i. Chinen, and Y. Shinoda, "StarBED and SpringOS: Large-scale general purpose network testbed and supporting software," in *Proceedings of the 1st international conference on Performance evaluation methodologies and tools*, 2006.
[62] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, *et al.*, "Grid'5000: A large scale and highly reconfigurable experimental grid testbed," *The International Journal of High Performance Computing Applications*, 2006.
[63] GÉANT, "Router for Academia Research and Education (RARE)." [Online]. Available: https://tinyurl.com/jkx764cb, Accessed on 10-17-2022.
[64] J. Kwon, J. A. García-Pardo, M. Legner, F. Wirz, M. Frei, D. Hausheer, and A. Perrig, "Scionlab: A next-generation internet testbed," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, 2020.
[65] H. Kim, X. Chen, J. Brassil, and J. Rexford, "Experience-driven research on programmable networks," *ACM SIGCOMM Computer Communication Review*, 2021.
[66] Florida International University's Center for Internet Augmented Research and Assessment (CIARA), "Americas Lightpaths Express and Protect." [Online]. Available: https://www.amlight.net/, Accessed on 10-17-2022.
[67] J. Crichigno, E. Kfoury, E. Bou-Harb, N. Ghani, Y. Prieto, C. Vega, J. Pezoa, C. Huang, and D. Torres, "A flow-based entropy characterization of a NATed network and its application on intrusion detection," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 2019.
[68] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, 2019.
[69] J. Mambretti, J. Chen, and F. Yeh, "Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn)," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, 2015.
[70] Network Development Group (NDG), "Netlab+." [Online]. Available: https://tinyurl.com/2p9az58x, Accessed on 08-27-2022.
[71] F. Kargl and E. Schoch, "Simulation of MANETs: a qualitative comparison between JiST/SWANS and ns-2," in *Proceedings of the 1st international workshop on System evaluation for mobile platforms*, 2007.

[72] K. Wehrle, M. Günes, and J. Gross, *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.

[73] J. Chamberlin, J. Hussey, B. Klimkowski, W. Moody, and C. Morrell, "The impact of virtualized technology on undergraduate computer networking education," in *Proceedings of the 18th Annual Conference on Information Technology Education*, 2017.

[74] A. M. Sllame and M. Jafaray, "Using simulation and modeling tools in teaching computer network courses," in *2013 International Conference on IT Convergence and Security (ICITCS)*, 2013.

[75] G. Carneiro, "ns-3: network simulator 3," in *UTM Lab Meeting April*, 2010.

[76] U. Lamping and E. Warnicke, "Wireshark user's guide," *Interface*, 2004.

[77] D. A. Joseph, V. Paxson, and S. Kim, "tcpdump tutorial," *University of California, EE122 Fall*, 2006.

[78] M. Mezzavilla, M. Miozzo, M. Rossi, N. Baldo, and M. Zorzi, "A lightweight and accurate link abstraction model for the simulation of LTE networks in ns-3," in *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, 2012.

[79] D. Ammar, T. Begin, and I. Guerin-Lassous, "A new tool for generating realistic Internet traffic in ns-3," in *Proceedings of the 4th international ICST conference on simulation tools and techniques*, 2011.

[80] M. Casoni, C. A. Grazia, M. Klapez, and N. Patriciello, "Implementation and validation of TCP options and congestion control algorithms for ns-3," in *Proceedings of the 2015 Workshop on Ns-3*, 2015.

[81] R. Ford, M. Zhang, S. Dutta, M. Mezzavilla, S. Rangan, and M. Zorzi, "A framework for end-to-end evaluation of 5G mmWave cellular networks in ns-3," in *Proceedings of the Workshop on ns-3*, 2016.

[82] A. Marinescu, I. Macaluso, and L. A. DaSilva, "System level evaluation and validation of the ns-3 LTE module in 3GPP reference scenarios," in *Proceedings of the 13th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2017.

[83] B. Bojovic, S. Lagen, and L. Giupponi, "Implementation and evaluation of frequency division multiplexing of numerologies for 5G new radio in ns-3," in *Proceedings of the 10th Workshop on ns-3*, 2018.

[84] L. Alberro, F. Velázquez, S. Azpiroz, E. Grampin, and M. Richart, "Experimenting with routing protocols in the data center: An ns-3 simulation approach," *Future Internet*, 2022.

[85] X. Li, M. Peng, J. Cai, C. Yi, and H. Zhang, "OPNET-based modeling and simulation of mobile ZigBee sensor networks," *Peer-to-Peer Networking and Applications*, 2016.

[86] P. Rukmani and R. Ganesan, "Scheduling algorithm for real time applications in mobile ad hoc network with opnet modeler," *Procedia Engineering*, 2013.

[87] X. Xian, W. Shi, and H. Huang, "Comparison of OMNET++ and other simulator for WSN simulation," in *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 2008.

[88] T. Zugno, M. Polese, M. Lecci, and M. Zorzi, "Simulation of next-generation cellular networks with ns-3: Open challenges and new directions," in *Proceedings of the 2019 Workshop on Next-Generation Wireless with ns-3*, 2019.

[89] I. Hammoodi, B. Stewart, A. Kocian, and S. McMeekin, "A comprehensive performance study of OPNET modeler for ZigBee wireless sensor networks," in *2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies*, 2009.

[90] R. Bagrodia, R. Meyer, M. Takai, Y.-a. Chen, X. Zeng, J. Martin, and H. Y. Song, "Parsec: A parallel simulation environment for complex systems," *Computer*, 1998.

[91] K. M. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *IEEE Transactions on software engineering*, 1979.

[92] Keysight Technologies, "Qualnet network simulator." [Online]. Available: `https://tinyurl.com/2ywntrzc`, Accessed on 01-10-2022.

[93] E. Ahvar and M. Fathy, "Performance evaluation of routing protocols for high density ad hoc networks based on QoS by glomosim simulator," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 2007.

[94] Keysight Technologies, "Network modeling." [Online]. Available: `https://tinyurl.com/22xtyn2x`, Accessed on 01-04-2022.

[95] K. A. Shuaib, "A performance evaluation study of WIMAX using QualNet," in *proceedings of the World Congress on Engineering*, 2009.

[96] RFC 5154, "Ip over ieee 802.16 problem statement and goals." [Online]. Available: `https://tinyurl.com/ycy29abt`, Accessed on 01-10-2022.

[97] A. Goyal, S. Vijay, and D. K. Jhariya, "Simulation and performance analysis of routing protocols in wireless sensor network using QualNet," *International Journal of computer applications*, 2012.

[98] P. Latkoski, V. Rakovic, O. Ognenoski, V. Atanasovski, and L. Gavrilovska, "SDL+QualNet: A novel simulation environment for wireless heterogeneous networks," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010.

[99] RFC 5677, "Ieee 802.21 mobility services framework design (msfd)." [Online]. Available: `https://www.rfc-editor.org/rfc/rfc5677`, Accessed on 01-10-2022.

[100] N. Finne, J. Eriksson, T. Voigt, G. Suciu, M.-A. Sachian, J. Ko, and H. Keipour, "Multi-trace: multi-level data trace generation with the Cooja simulator," in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2021.

[101] D. Jabba and P. Acevedo, "ViTool-BC: Visualization tool based on Cooja simulator for WSN," *Applied Sciences*, 2021.

[102] H. S. Zenalabdin, A. Buhari, and T. E. Nyamasvisva, "Performance analysis of IoT protocol stack over dense and sparse mote network using Cooja simulator," in *Journal of Physics: Conference Series*, 2020.

[103] Centre Tecnologic de Telecomunicacions de Catalunya, "ns-3 lena." [Online]. Available: `https://tinyurl.com/2ac2thrw`, Accessed on 07-24-2023.

[104] B. Bojović, S. Lagén, and L. Giupponi, "Realistic beamforming design using SRS-based channel estimate for ns-3 5G-LENA module," in *Proceedings of the 2021 Workshop on ns-3*, 2021.

[105] H. Assasa and N. Grosheva, "Wigig module." [Online]. Available: `https://tinyurl.com/2hcvuz7c`, Accessed on 07-24-2023.

[106] H. Assasa, N. Grosheva, T. Ropitault, S. Blandino, N. Golmie, and J. Widmer, "Implementation and evaluation of a WLAN IEEE 802.11 ay model in network simulator ns-3," in *Proceedings of the 2021 Workshop on ns-3*, 2021.

[107] M. Z. Khan, B. Askwith, F. Bouhafs, and M. Asim, "Limitations of simulation tools for large-scale wireless sensor networks," in *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, 2011.

[108] K. Tan, D. Wu, A. J. Chan, and P. Mohapatra, "Comparing simulation tools and experimental testbeds for wireless mesh networks," *Pervasive and Mobile Computing*, 2011.

[109] H. Sundani, H. Li, V. Devabhaktuni, M. Alam, and P. Bhattacharya, "Wireless sensor network simulators a survey and comparisons," *International Journal of Computer Networks*, 2011.

[110] N. M. M. Noor, N. Yayao, and S. Sulaiman, "Effectiveness of using Cisco Packet Tracer as a learning tool: A case study of routing protocol," *Computer software*, 2018.

[111] N. Gwangwava, T. B. Mubvirwi, *et al.*, "Design and simulation of IoT systems using the Cisco Packet Tracer," *Advances in Internet of Things*, 2021.

[112] J. Allison, "Simulation-based learning via Cisco Packet Tracer to enhance the teaching of computer networks," in *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, 2022.

[113] P. Li, "Selecting and using virtualization solutions: our experiences with VMware and VirtualBox," *Journal of Computing Sciences in Colleges*, 2010.

[114] T. R. Velieva, A. V. Korolkova, and D. S. Kulyabov, "Designing installations for verification of the model of active queue management discipline RED in the GNS3," in *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, IEEE, 2014.

[115] P. Gil, G. J. Garcia, A. Delgado, R. M. Medina, A. Calderon, and P. Marti, "Computer networks virtualization with

GNS3," in *proc IEEE Frontiers in Education Conference*, 2014.

[116] R. Emiliano and M. Antunes, "Automatic network configuration in virtualized environment using GNS3," in *2015 10th International Conference on Computer Science & Education (ICCSE)*, 2015.

[117] P. Mihăilă, T. Bălan, R. Curpen, and F. Sandu, "Network automation and abstraction using python programming methods," *MACRo 2015*, 2017.

[118] J.-I. Castillo-Velazquez, F. DeLaCruz-Alejandre, and M. Huerta, "An approach to management assessment for GEANT backbone using GNS3 for SNMPv3," in *2018 IEEE 38th Central America and Panama Convention (CONCAPAN XXXVIII)*, 2018.

[119] N. S. Tarkaa, P. I. Iannah, and I. T. Iber, "Design and simulation of local area network using Cisco Packet Tracer," *The International Journal of Engineering and Science*, 2017.

[120] U. Yaqub, A. Al-Nasser, and T. Sheltami, "Implementation of a hybrid wind-solar desalination plant from an Internet of Things (IoT) perspective on a network simulation tool," *Applied computing and informatics*, 2019.

[121] C. Dumitrache, G. Predusca, L. Circiumarescu, N. Angelescu, and D. Puchianu, "Comparative study of RIP, OSPF and EIGRP protocols using Cisco Packet Tracer," in *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*, 2017.

[122] N. A. Rashid, Z. bin Othman, R. bin Johan, and S. b. H. Sidek, "Cisco Packet Tracer simulation as effective pedagogy in computer networking course," 2019.

[123] A. Musheer, O. Sotnikov, and S. S. Heydari, "Multiuser simulation-based virtual environment for teaching computer networking concepts," *International Journal on Advances in Intelligent Systems*, 2012.

[124] Linux Containers, "Container and virtualization tools ." [Online]. Available: `https://linuxcontainers.org/`, Accessed on 06-23-2022.

[125] Docker Containers, "What is a Docker container? ." [Online]. Available: `https://tinyurl.com/bdz3ejdh`, Accessed on 06-23-2022.

[126] W. Almesberger *et al.*, "Linux network traffic control—implementation overview," 1999.

[127] Mininet-HiFi experiments, "Reproducing network research." [Online]. Available: `https://tinyurl.com/ycy3hbzb`, Accessed on 01-04-2022.

[128] A. Mohammad, G. Albert, M. David, P. Jitendra, P. Parveen, P. Balaji S. Sudipta, S. Murari, "Data center TCP (DCTCP)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010.

[129] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, M. Yasuda, "Hedera: Dynamic flow scheduling for data center networks.," in *Networked Systems Design and Implementation (NSDI)*, 2010.

[130] Ö. Sen, D. van der Velde, S. N. Peters, and M. Henze, "An approach of replicating multi-staged cyber-attacks and countermeasures in a smart grid co-simulation environment," 2021.

[131] M. Amoretti, R. Pecori, Y. Protskaya, L. Veltri, and F. Zanichelli, "A scalable and secure publish/subscribe-based framework for industrial IoT," *IEEE Transactions on Industrial Informatics*, 2020.

[132] The OpenAirInterface Software Alliance, "Open air interface." [Online]. Available: `https://openairinterface.org/`, Accessed on 07-24-2023.

[133] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5G research," *ACM SIGCOMM Computer Communication Review*, 2014.

[134] srsRAN Project, "Open source ran." [Online]. Available: `https://www.srsran.com/`, Accessed on 07-24-2023.

[135] iMdea Networks, "openleon." [Online]. Available: `https://tinyurl.com/4edcaxuv`, Accessed on 07-24-2023.

[136] C. Fiandrino, A. B. Pizarro, P. J. Mateo, C. A. Ramiro, N. Ludant, and J. Widmer, "openLEON: An end-to-end emulation platform from the edge data center to the mobile user," *Computer Communications*, 2019.

[137] M. Peuster, H. Karl, and S. Van Rossem, "Medicine: Rapid prototyping of production-ready network services in multi-pop environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2016.

[138] J. Joy, Y.-T. Yu, M. Gerla, S. Wood, J. Mathewson, and M.-O. Stehr, "Network coding for content-based intermittently connected emergency networks," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, 2013.

[139] R. C. Lunardi, R. A. Michelin, C. V. Neu, H. C. Nunes, A. F. Zorzo, and S. S. Kanhere, "Impact of consensus on appendable-block blockchain for IoT," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019.

[140] R. Tomsett, G. Bent, C. Simpkin, I. Taylor, D. Harbourne, A. Preece, and R. Ganti, "Demonstration of dynamic distributed orchestration of node-RED IoT workflows using a vector symbolic architecture," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2019.

[141] J. Obstfeld, S. Knight, E. Kern, Q. S. Wang, T. Bryan, and D. Bourque, "VIRL: the virtual internet routing lab," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014.

[142] O. Sefraoui, M. Aissaoui, M. Eleuldj, *et al.*, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, 2012.

[143] S. Knight, A. Jaboldinov, O. Maennel, I. Phillips, and M. Roughan, "Autonetkit: simplifying large scale, open-source network experimentation," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012.

[144] L. M. M. Zorello, S. Troia, S. Giannotti, R. Alvizu, S. Bregni, and G. Maier, "On the network slicing for enterprise services with hybrid SDN," in *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, 2020.

[145] B. Al-Musawi, R. Al-Saadi, P. Branch, and G. Armitage, "BGP replay tool (BRT) v0.1," *Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. A*, 2016.

[146] T. T. Dinh-Trong and J. M. Bieman, "The FreeBSD project: A replication case study of open source development," *IEEE Transactions on Software Engineering*, 2005.

[147] S. Kuman, S. Groš, and M. Mikuc, "An experiment in using IMUNES and Conpot to emulate honeypot control networks," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017.

[148] D. Salopek, V. Vasić, M. Zec, M. Mikuc, M. Vašarević, and V. Končar, "A network testbed for commercial telecommunications product testing," in *2014 22nd international conference on software, telecommunications and computer networks (SoftCOM)*, 2014.

[149] J. N. Herder, H. Bos, B. Gras, P. Homburg, and A. S. Tanenbaum, "MINIX 3: A highly reliable, self-repairing operating system," *ACM SIGOPS Operating Systems Review*, 2006.

[150] J. Ahrenholz, "Comparison of CORE network emulation platforms," in *2010-Milcom 2010 Military Communications Conference*, 2010.

[151] D. Tyler and T. Viana, "Trust no one? a framework for assisting healthcare organisations in transitioning to a zero-trust network architecture," *Applied Sciences*, 2021.

[152] M. Zec and M. Mikuc, "Operating system support for integrated network emulation in IMUNES," in *1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (OASIS)*, 2004.

[153] E. F. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Computer Communications*, 2020.

[154] J. Gomez, E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "A performance evaluation of TCP BBRv2 alpha," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020.

[155] R. Lübke, P. Büschel, D. Schuster, and A. Schill, "Measuring accuracy and performance of network emulators," in *2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, IEEE, 2014.

[156] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, 2010.

[157] S. Szilágyi and I. Bordán, "Throughput performance measurement of the MPT-GRE multipath technology in emulated wan environment," in *Proceedings of the 1st Conference on Information Technology and Data Science: CITDS*, 2020.

[158] K. Noda and Y. Ito, "Study of multi-path TCP scheduler to

suppress QoS fluctuation for improving WebQoE," in *2019 International Conference on Electronics, Information, and Communication (ICEIC)*, 2019.

[159] R. Al-Saadi and G. Armitage, "Dummynet AQM v0. 2–CoDel, FQ-CoDel, PIE and FQ-PIE for FreeBSD's ipfw/dummynet framework," *Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. A*, 2016.

[160] R. A. Netravali, *Understanding and improving web page load times on modern networks*. PhD thesis, Massachusetts Institute of Technology, 2015.

[161] B. Zhang, T. Teixeira, and Y. Reznik, "Performance of low-latency HTTP-based streaming players," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021.

[162] A. G. Moe, "Implementing rate control in NetEm: Untying the NetEm/tc tangle," Master's thesis, 2013.

[163] Network Development Group (NDG), "Mission and vision." [Online]. Available: `https://tinyurl.com/2vha2h7m`, Accessed on 08-27-2022.

[164] VMware, "vSphere." [Online]. Available: `https://www.vmware.com/products/vsphere.html`, Accessed on 03-21-2022.

[165] J. Gomez, E. F. Kfoury, and J. Crichigno, "Enabling P4 hands-on training in an academic cloud," in *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2022.

[166] P. Gomez-Sanchez, S. Mendez, A. De Giusti, M. Naiouf, E. Luque, A. Bezerra, D. Encinas, J. Panadero, and D. Rexachs, "Using AWS EC2 as test-bed infrastructure in the i/o system configuration for HPC applications," *Journal of Computer Science and Technology*, 2016.

[167] P. Ruth and M. Cevik, "Experimenting with AWS direct connect using chameleon, ExoGENI, and Internet2 cloud connect," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, 2019.

[168] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high-performance computing applications on the amazon web services cloud," in *2010 IEEE second international conference on cloud computing technology and science*, 2010.

[169] B. Ruan, H. Huang, S. Wu, and H. Jin, "A performance study of containers in cloud environment," in *Advances in Services Computing: 10th Asia-Pacific Services Computing Conference, APSCC 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings 10*, 2016.

[170] N. Kratzke and P.-C. Quint, "Investigation of impacts on network performance in the advance of a microservice design," in *Cloud Computing and Services Science: 6th International Conference, CLOSER 2016, Rome, Italy, April 23-25, 2016, Revised Selected Papers 6*, 2017.

[171] V. Persico, P. Marchetta, A. Botta, and A. Pescapé, "On network throughput variability in Microsoft Azure cloud," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015.

[172] H. A. Hassan, S. A. Mohamed, and W. M. Sheta, "Scalability and communication performance of HPC on Azure cloud," *Egyptian Informatics Journal*, 2016.

[173] A. AlSabeh, H. Safa, E. Bou-Harb, and J. Crichigno, "Exploiting ransomware paranoia for execution prevention," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020.

[174] C. Siaterlis, A. P. Garcia, and B. Genge, "On the use of Emulab testbeds for scientifically rigorous experiments," *IEEE Communications Surveys & Tutorials*, 2012.

[175] Flux Research Group, "Emulab user guide." [Online]. Available: `https://tinyurl.com/4e88yw5a`, Accessed on 07-07-2022.

[176] NS-3 Consortium, "Network simulator 3 user guide." [Online]. Available: `https://tinyurl.com/2p9brfe2`, Accessed on 07-07-2022.

[177] Flux Research Group, "Emulab installation documentation." [Online]. Available: `https://tinyurl.com/2p836txk`, Accessed on 07-07-2022.

[178] GENI Documentation, "Geni architecture." [Online]. Available: `https://www.geni.net/documentation/geni-architecture/`, Accessed on 05-22-2022.

[179] C. Barnes and T. E. Jackson, "Internet2: The backbone of the future," tech. rep., Air Force Research Lab, Meza, AZ, 2002.

[180] Juniper, "Ibm bnt g8264r." [Online]. Available: `https://tinyurl.com/yndx38fp`, Accessed on 05-24-2022.

[181] IBM, "IBM BNT G8052R rack switch." [Online]. Available: `https://tinyurl.com/2t8bm73p`, Accessed on 05-25-2022.

[182] IBM, "Ibm x3650 m3." [Online]. Available: `https://tinyurl.com/3ptrr25u`, Accessed on 05-24-2022.

[183] Hewlett Packard, "HP Procurve 2620 switch." [Online]. Available: `https://tinyurl.com/ym3hvsvx`, Accessed on 05-25-2022.

[184] Hewlett Packard, "Hp proliant dl360 g7." [Online]. Available: `https://tinyurl.com/437ftrj7`, Accessed on 05-25-2022.

[185] Dell, "Dell PowerEdge R620 series." [Online]. Available: `https://tinyurl.com/mrwp288n`, Accessed on 05-31-2022.

[186] DELL Technologies, "Force10 s2410-01-10ge-24p." [Online]. Available: `https://tinyurl.com/3j6u38yr`, Accessed on 05-24-2022.

[187] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb, *et al.*, "The design and operation of CloudLab," in *2019 USENIX annual technical conference (USENIX ATC 19)*, 2019.

[188] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in *IEEE Wireless Communications and Networking Conference, 2005*, 2005.

[189] J. Zamora, F. Fund, A. Koutsaftis, and S. S. Panwar, "An open-access research testbed for visible light communication," in *Proceedings of the 4th ACM Workshop on Visible Light Communication Systems*, 2017.

[190] A. Abdelhadi, F. Rechia, A. Narayanan, T. Teixeira, R. Lent, D. Benhaddou, H. Lee, and T. C. Clancy, "Position estimation of robotic mobile nodes in wireless testbed using GENI," in *2016 Annual IEEE Systems Conference (SysCon)*, 2016.

[191] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, 2003.

[192] Y. Wang, W.-J. Hsin, and M. Lamsal, "EdGENI: Making GENI User-Friendly for General Computer Education," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, 2022.

[193] GENI testbed, "The edgeni project." [Online]. Available: `https://tinyurl.com/2p8rvtmc`, Accessed on 05-25-2022.

[194] The Ethereum Foundation, "Solidity documentation." [Online]. Available: `https://tinyurl.com/2p9fjajc`, Accessed on 05-25-2022.

[195] The REMIX Project, "Remix ide." [Online]. Available: `https://remix-project.org/`, Accessed on 05-25-2022.

[196] InfiniBand Trade Association, "InfiniBand Architecture Specification." [Online]. Available: `https://tinyurl.com/2p8jc3pm`, Accessed on 06-10-2022.

[197] N. Grinsztajn, O. Beaumont, E. Jeannot, and P. Preux, "READYS: A reinforcement learning based strategy for heterogeneous dynamic scheduling," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, 2021.

[198] Á. Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, and V. Muntés-Mulero, "Graph-based root cause analysis for service-oriented and microservice architectures," *Journal of Systems and Software*, 2020.

[199] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos, "FogIoT orchestrator: an orchestration system for IoT applications in fog environment," in *1st Grid'5000-FIT school*, 2018.

[200] D. E. Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Multi-site connectivity for edge infrastructures: DIMINET: DIstributed Module for Inter-site NETworking," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pp. 121–130, IEEE, 2020.

[201] G. Song, S. Park, and M. Lee, "Using Emulab for deep learning performance comparisons among network topologies," in *Proceedings of the 2019 4th International Conference on Intelligent Information Technology*, 2019.

[202] C. C. Kuo, K. Chain, and C. S. Yang, "Cyber attack and defense training: Using Emulab as a platform," *Int. J. Innov. Comput. Inf. Control*, 2018.

[203] G.-B. Song and M.-H. Lee, "Emulearner: Deep learning library for utilizing Emulab," *Journal of information and com-*

munication convergence engineering, 2018.

[204] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the Emulab network testbed," in *2008 USENIX Annual Technical Conference (USENIX ATC 08)*, 2008.

[205] S. Edwards, X. Liu, and N. Riga, "Creating repeatable computer science and networking experiments on shared, public testbeds," *ACM SIGOPS Operating Systems Review*, 2015.

[206] D. Balouek, A. C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, *et al.*, "Adding virtualization capabilities to the Grid'5000 testbed," in *International Conference on Cloud Computing and Services Science*, 2012.

[207] J. Mikovic, P. G. Kannan, C. Mun Choon, and K. Sklower, "Enabling SDN experimentation in network testbeds," in *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2017.

[208] Y. Park, H. Hu, X. Yuan, and H. Li, "Enhancing security education through designing SDN security labs in CloudLab," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018.

[209] L. B. Ngo and J. Denton, "Using CloudLab as a scalable platform for teaching cluster computing ambassador program," *The Journal of Computational Science Education*, 2019.

[210] L. B. Ngo, A. T. Srinath, J. Denton, and M. Ziolkowski, "Unifying computing resources and access interface to support parallel and distributed computing education," *Journal of Parallel and Distributed Computing*, 2018.

[211] T. D. Project, "DETERLAB wiki." [Online]. Available: `https://trac.deterlab.net/wiki`, Accessed on 09-13-2022.

[212] C. A. Sunshine, "Source routing in computer networks," *ACM SIGCOMM Computer Communication Review*, 1977.

[213] D. Barrera, L. Chuat, A. Perrig, R. M. Reischuk, and P. Szalachowski, "The SCION internet architecture," *Communications of the ACM*, 2017.

[214] Institute for the Wireless Internet of Things at Northeastern, "Colosseum." [Online]. Available: `https://tinyurl.com/3av3v3da`, Accessed on 07-24-2023.

[215] IEEE Future Networks Initiative, "Ieee 5g/6g innovation testbed." [Online]. Available: `https://testbed.ieee.org/`, Accessed on 08-04-2023.

[216] IEEE, "Ieee future networks." [Online]. Available: `https://futurenetworks.ieee.org/`, Accessed on 08-04-2023.

[217] A. Ibrahim and V. Ford, "Observations, evaluations, and recommendations for deterlab from an educational perspective," *Journal of Cybersecurity Education, Research and Practice*, 2021.

[218] SCIONLab tutorials, "SCION Education." [Online]. Available: `https://tinyurl.com/3uj7mewj`, Accessed on 09-19-2022.

[219] H. An, Y. Na, H. Lee, and A. Perrig, "Resilience evaluation of multi-path routing against network attacks and failures," *Electronics*, 2021.

[220] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, N. Mckeown, and G. Parulkar, "Can the production network be the testbed?," in *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[221] H. Kim and A. Gupta, "Ontas: Flexible and scalable online network traffic anonymization system," in *Proceedings of the 2019 Workshop on Network Meets AI & ML*, 2019.

[222] J. Ibarra, J. Bezerra, H. Morgan, L. F. Lopez, D. A. Cox, M. Stanton, I. Machado, and E. Grizendi, "Benefits brought by the use of OpenFlow/SDN on the AmLight intercontinental research and education network," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[223] H. Galiza, M. Schwarz, J. Bezerra, and J. Ibarra, "Moving an IP network to SDN: a global use case deployment experience at AmLight," in *Anais do WPEIF 2016 Workshop de Pesquisa Experimental da Internet do Futuro*, 2016.

[224] J. Bezerra, I. Brito, A. Quintana, J. Ibarra, V. Chergarova, R. Frez, H. Morgan, M. LeClerc, and A. Paneri, "Deploying per-packet telemetry in a long-haul network: the amlight use case," in *2021 IEEE Workshop on Innovating the Network for Data-Intensive Science (INDIS)*, 2021.

[225] J. Ibarra, "Amlight express and protect (amlight-exp)," *[Presentation] NSF Distinguished Lecture Series*, 2022.

[226] M. Cevik, M. Stealey, C. Wang, J. Bezerra, J. Ibarra, V. Chergarova, H. Morgan, and Y. Xin, "Towards production deployment of a SDX control framework," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, 2022.

[227] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, 1948.

[228] O. Michel, S. Sengupta, H. Kim, R. Netravali, and J. Rexford, "Enabling passive measurement of Zoom performance in production networks," in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022.

[229] S. Bai, H. Kim, and J. Rexford, "Passive OS fingerprinting on commodity switches," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, 2022.

[230] J. Kim, H. Kim, and J. Rexford, "Analyzing traffic by domain name in the data plane," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pp. 1–12, 2021.

[231] FABRIC, "Knowledge Base." [Online]. Available: `https://tinyurl.com/2b9j7mdy`, Accessed on 06-09-2022.

[232] Project Jupyter, "JupyterHub." [Online]. Available: `https://jupyter.org/hub`, Accessed on 06-09-2022.

[233] Dell Technolgies, "PowerEdge R7525 Rack Server." [Online]. Available: `https://tinyurl.com/2hfc77u4`, Accessed on 06-10-2022.

[234] Kuang-Ching Wang, Paul Ruth, "FABRIC: An Everywhere Programmable Research Infrastructure for Network Experimentation ." [Online]. Available: `https://tinyurl.com/2szb2kbw`, Accessed on 06-10-2022.

[235] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data," 2020.

[236] R. Kumar, M. Baughman, R. Chard, Z. Li, Y. Babuji, I. Foster, and K. Chard, "Coding the computing continuum: Fluid function execution in heterogeneous computing environments," in *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2021.

[237] M. Baughman, R. Kumar, I. Foster, and K. Chard, "Expanding cost-aware function execution with multidimensional notions of cost," in *Proceedings of the 1st Workshop on High Performance Serverless Computing*, 2020.

[238] V. Turina, Z. Zhang, F. Esposito, and I. Matta, "Combining split and federated architectures for efficiency and privacy in deep learning," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020.

[239] P. Yu, J. Liu, and M. Chowdhury, "Fluid: Resource-aware hyperparameter tuning engine," *Proceedings of Machine Learning and Systems*, 2021.

[240] Univerisy of Texas, "Texas Advanced Computing Center (TACC)." [Online]. Available: `https://www.tacc.utexas.edu/`, Accessed on 06-09-2022.

[241] University of Illinois Urbana-Champaign, "National Center for Supercomputing Applications (NCSA)." [Online]. Available: `https://www.ncsa.illinois.edu/`, Accessed on 06-09-2022.

[242] U.S. Department of Energy, "Lawrence Berkeley Lab (LBL)." [Online]. Available: `https://www.lbl.gov/`, Accessed on 06-09-2022.

[243] University of Califorina, San Diego, "San Diego Supercomputing Center (SDSC)." [Online]. Available: `https://www.sdsc.edu/`, Accessed on 06-09-2022.

[244] The University of Utah, "The Platform for Open Wireless Data-driven Experimental Research (POWDER)." [Online]. Available: `https://powderwireless.net/`, Accessed on 08-24-2022.

[245] The AERPAW development group, "The Aerial Experimentation and Research for Advanced Wireless (AERPAW)." [Online]. Available: `https://aerpaw.org/`, Accessed on 08-24-2022.

[246] P. development team, "The PEERING testbed." [Online]. Available: `https://peering.ee.columbia.edu/`, Accessed on 08-24-2022.

[247] Chameleon cloud, "PI eligibility." [Online]. Available: `https://tinyurl.com/29s2k7ar`, Accessed on 06-23-2022.

[248] Chameleon cloud, "Best practices manual." [Online]. Available: `https://tinyurl.com/zsvp4mdd`, Accessed on 06-23-2022.

[249] Chameleon cloud, "Graphical User Interface." [Online]. Available: `https://tinyurl.com/yc8acn8w`, Accessed on 06-

23-2022.
[250] M. Cevik, P. Ruth, K. Keahey, and P. Riteau, "Wide-area software defined networking experiments using Chameleon," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019.
[251] Chameleon cloud, "Resource discovery." [Online]. Available: `https://tinyurl.com/2hzvy8fw`, Accessed on 06-23-2022.
[252] Network Development Group (NDG), "Designated Operating Environment - 2022." [Online]. Available: `https://tinyurl.com/bp4w7wma`, Accessed on 08-27-2022.
[253] VMware, "VMware ESXi: The Purpose-Built Bare Metal Hypervisor." [Online]. Available: `https://tinyurl.com/yx4c8w3m`, Accessed on 03-22-2022.
[254] VMware, "VCenter server." [Online]. Available: `https://www.vmware.com/products/vcenter-server.html`, Accessed on 03-22-2022.
[255] Network Development Group (NDG), "NDG online courses and labs." [Online]. Available: `https://tinyurl.com/3fa2c92d`, Accessed on 10-17-2022.
[256] E. Gavaletz and J. Kaur, "Decomposing RTT-unfairness in transport protocols," in *2010 17th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, 2010.
[257] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, 2008.
[258] E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "Dynamic router's buffer sizing using passive measurements and P4 programmable switches," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021.
[259] Intel, "Intel Tofino Ethernet Switch ASIC." [Online]. Available: `https://tinyurl.com/mry8a8c4`, Accessed on 02-21-2023.
[260] Edgecore Networks, "Wedge 100BF-32X." [Online]. Available: `https://tinyurl.com/2xay8kky`, Accessed on 02-23-2023.
[261] Intel, "Intel Tofino Ethernet switch ASIC." [Online]. Available: `https://tinyurl.com/mry8a8c4`, Accessed on 02-24-2023.
[262] P. Ruth, I. Baldin, K. Thareja, T. Lehman, X. Yang, and E. Kissel, "FABRIC network service model," in *2022 IFIP Networking Conference (IFIP Networking)*, 2022.
[263] FABRIC, "FABRIC Across Borders (FAB)." [Online]. Available: `https://fabric-testbed.net/about/fab/`, Accessed on 09-19-2022.
[264] Internet2, "Advance layer 2 service." [Online]. Available: `https://tinyurl.com/4pje2s7w`, Accessed on 09-14-2022.
[265] Energy Sciences Network, "On-demand secure circuits and advance reservation system OSCARS." [Online]. Available: `https://tinyurl.com/yeyk97h4`, Accessed on 09-14-2022.
[266] A. AlSabeh, J. Khoury, E. Kfoury, J. Crichigno, and E. Bou-Harb, "A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment," *Computer Networks*, 2022.
[267] C. Fan, J. Bi, Y. Zhou, C. Zhang, and H. Yu, "NS4: A P4-driven network simulator," in *Proceedings of the SIGCOMM Posters and Demos*, 2017.
[268] ETH Zurich, "An open platform to teach how the internet practically works." [Online]. Available: `https://github.com/nsg-ethz/mini_internet_project`, Accessed on 01-04-2022.
[269] G. Di Lena, A. Tomassilli, D. Saucez, F. Giroire, T. Turletti, and C. Lac, "Distrinet: A Mininet implementation for the cloud," *ACM SIGCOMM Computer Communication Review*, 2021.
[270] G. Di Lena, A. Tomassilli, D. Saucez, F. Giroire, T. Turletti, and C. Lac, "Mininet on steroids: exploiting the cloud for Mininet performance," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*.
[271] L. Rizzo, G. Lettieri, and V. Maffione, "Very high speed link emulation with TLEM," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2016.
[272] K. Keahey, J. Anderson, M. Sherman, C. Hammock, Z. Zhen, J. Tillotson, T. Bargo, L. Long, T. Ul Islam, S. Babu, *et al.*, "CHI-in-a-Box: Reducing operational costs of research testbeds," in *Practice and Experience in Advanced Research Computing*, 2022.
[273] FABRIC, "FABRIC forum." [Online]. Available: `https://tinyurl.com/pb9zwauw`, Accessed on 10-17-2022.
[274] FABRIC, "FABRIC testbed release 1.3." [Online]. Available: `https://tinyurl.com/2jaevsca`, Accessed on 10-17-2022.
[275] FABRIC, "FABRIC testbed release 1.3." [Online]. Available: `https://tinyurl.com/56mtbew4`, Accessed on 10-17-2022.
[276] B. Chung, C. Chen, C.-C. Tseng, J. H. Chen, and J. Mambretti, "P4MT: Designing and evaluating multi-tenant services for P4 switches," in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2021.
[277] P4 consortium, "P4 lang github repository." [Online]. Available: `https://tinyurl.com/2as9pnwn`, Accessed on 07-01-2022.
[278] Barefoot Networks, "P4 Intel Tofino native architecture - public version." [Online]. Available: `https://tinyurl.com/5d7nznwd`, Accessed on 03-12-2022.
[279] FRRouting Project, "Frrouting." [Online]. Available: `https://frrouting.org/`, Accessed on 02-18-2022.
[280] J. Mirkovic and P. Pusey, "User experiences on network testbeds," in *Cyber Security Experimentation and Test Workshop*, 2021.
[281] The P4 Consortium, "The BMv2 simple switch target." [Online]. Available: `https://tinyurl.com/yc35ttuc`, Accessed on 10-17-2022.
[282] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, 2021.