Journal of Cybersecurity and Privacy

MDPI

Article

Deploying Secure Distributed Systems: Comparative Analysis of GNS3 and SEED Internet Emulator

Lewis Golightly [1], Paolo Modesti [1] and Victor Chang [2,*]

[1] Department of Computing and Games, Teesside University, Middlesbrough TS1 3BX, UK; l.golightly@tees.ac.uk (L.G.); p.modesti@tees.ac.uk (P.M.)
[2] Department of Operations and Information Management, Aston University, Birmingham B4 7ET, UK
* Correspondence: v.chang1@aston.ac.uk or victorchang.research@gmail.com

Abstract: Network emulation offers a flexible solution for network deployment and operations, leveraging software to consolidate all nodes in a topology and utilizing the resources of a single host system server. This research paper investigated the state of cybersecurity in virtualized systems, covering vulnerabilities, exploitation techniques, remediation methods, and deployment strategies, based on an extensive review of the related literature. We conducted a comprehensive performance evaluation and comparison of two network-emulation platforms: Graphical Network Simulator-3 (GNS3), an established open-source platform, and the SEED Internet Emulator, an emerging platform, alongside physical Cisco routers. Additionally, we present a Distributed System that seamlessly integrates network architecture and emulation capabilities. Empirical experiments assessed various performance criteria, including the bandwidth, throughput, latency, and jitter. Insights into the advantages, challenges, and limitations of each platform are provided based on the performance evaluation. Furthermore, we analyzed the deployment costs and energy consumption, focusing on the economic aspects of the proposed application.

Keywords: secure distributed systems; network emulation; GNS3; SEED Internet Emulator; Cisco routers; performance evaluation; cybersecurity

Citation: Golightly, L.; Modesti, P.; Chang, V. Deploying Secure Distributed Systems: Comparative Analysis of GNS3 and SEED Internet Emulator. *J. Cybersecur. Priv.* **2023**, *3*, 464–492. https://doi.org/10.3390/jcp3030024

Academic Editor: Rodrigo Román-Castro

Received: 6 June 2023
Revised: 24 July 2023
Accepted: 31 July 2023
Published: 3 August 2023

1. Introduction

Network emulation is an effective approach to network deployment for various use-cases by offering a dynamic, programmable, and efficient configuration, along with significant performance and monitoring advantages. This method has gained popularity as an alternative and addition to traditional network solutions, providing enhanced business agility through improved flexibility, scalability, and security in particular areas [1]. Organizations, including Small- and Medium-sized Enterprises (SMEs), can adopt network-emulation technology to enhance or expand their existing network systems. By dividing the network into physical and emulated devices using both vendor proprietary and open-source communication protocols, greater control over network device management and overall intelligence can be achieved, leading to efficient resource utilization and improved security, ultimately helping organizations achieve their business objectives [2]. Energy efficiency is a major benefit of implementing network emulation, with potential savings of up to sixty percent compared to physical network infrastructure [3]. This is due to the optimization of underutilized networking components through emulation.

Server consolidation is another approach to minimize the number of physical machines and makes use of virtual machines, contributing to energy efficiency. Additionally, network optimization and resource reusability from previous work can lead to further energy savings [4]. By considering these factors, organizations can significantly reduce energy consumption and costs while enhancing the overall sustainability of their network infrastructure through emulation implementation. However, scalability remains a signifi-

cant challenge in emulation alone, as it heavily relies on host system resources and can face bottlenecks, particularly when scaling the network size and node quantity [5].

Hybrid network emulation solutions, which combine a physical network infrastructure with a software-based emulation architecture, present significant opportunities. One of the main benefits is the cost reduction compared to a complete migration to an emulation architecture [6]. However, the hybrid model also introduces an expanded attack surface and increased security concerns. To address these concerns, integrating security solutions and adopting a security-by-design approach are essential for maximum mitigation [7].

Furthermore, the use of emulated environments, such as the SEED Internet Emulator [8], can improve deployment efficiency and provide value for large-scale deployments in a short period [6]. Recent studies have explored the communication architecture of network emulation and conducted performance testing to assess the technology's capabilities under diverse conditions. These network emulators can also be utilized to run security software such as access control for information security [9].

1.1. Research Motivations

Several motivations drove this research, including:

1. The investigation of how network emulation and hybrid systems can provide advanced control and flexibility when integrated with existing infrastructure, enabling organizations to adapt to changing business needs and network demands.
2. To explore the ability to use network emulation and hybrid systems for offering a cost-effective and efficient solution for businesses, particularly SMEs, by combining existing physical infrastructure with an emulation architecture.

1.2. Research Contributions

In this study, we present the experimental results of a proof-of-concept network and analyze the performance of various solutions, including physical (Cisco infrastructure), emulated (GNS3, SEED Internet Emulator), and a hybrid network model (hybrid emulated). We begin by conducting comparative testing of the performance between GNS3 the SEED Internet Emulator based on predefined performance criteria, such as the bandwidth, latency, throughput, as well as some other relevant metrics. The hybrid system is developed by integrating physical Cisco devices and emulated (GNS3) Cisco appliances, enabling us to evaluate the performance and quality of the hybrid emulation integration through similar experiments. Finally, we assess the economic benefits of a specific network implementation in energy consumption and cost.

The main contributions of our research are:

1. Providing a comprehensive analysis of cybersecurity in virtualization technology, networking, and hybrid systems, including cryptographic mechanisms, and a security evaluation through an exploration of evidence-based related literature (Section 2).
2. Conducting a performance comparison of network emulation systems and a hybrid system, considering specific benchmarks such as the bandwidth, latency, throughput, and other relevant metrics (Section 5), as well as evaluating the energy consumption and cost-effectiveness through data analysis and addressing the reliability and usability considerations around the platforms (Section 6).
3. Analyzing the deployment and application of hybrid emulation systems, combining physical Cisco infrastructure with emulated (GNS3) components, and conveying an evaluation, including identifying the strengths, challenges, and notable findings of hybrid emulation (Section 4), as well as providing specific recommendations for its implementation, optimization, and further enhancements (Section 6).

2. Background

This section explores the cybersecurity vulnerabilities of deploying virtualized network emulation platforms focused on exploiting low-level mechanisms. A Virtual Machine (VM) is a file that contains an emulated platform or operating system. A hypervisor

is a software solution that runs these files through resource allocation on the physical machine [10].

2.1. Emulated and Virtualized Systems

Network emulation refers to the use of software to replicate a network environment and nodes using specific files, such as Docker containers, that utilize the resources of a server's host system. On the other hand, network virtualization involves the utilization of virtual devices to replace physical devices in a production network [11]. Two variations of packaged computing environments are commonly used: ***containers*** and ***Virtual Machines (VMs)***.

Containers are isolated sets of one or more processes that have limited access to specified resources. They leverage copy-on-write layered file systems and version control, enabling continuous integration and delivery in software development. The SEED Internet Emulator utilizes Docker integration, making it an efficient network-emulation platform. Containers generally outperform VMs in terms of performance and scalability, making them well-suited for application deployment. However, in multi-tenant scenarios, where performance interference may occur, VMs offer enhanced security and root privilege, making them preferable for handling business-critical data [12].

2.1.1. Network Emulation vs. Virtualization

Network emulation involves emulating a virtual equivalent of a production device for a specific application domain. In contrast, virtualization refers to the software-based division of a physical system into various components [13].

2.1.2. Hypervisors and Data Privacy

A hypervisor, also known as a ***Virtual Machine Monitor (VMM)***, is a software, hardware, or firmware system that enables the execution of virtual machines. Two types of hypervisors exist: Type 1 (hardware) and Type 2 (software) [14].

VMware ESXi is an enterprise-grade virtualization system that provides a kernel and other operating system components. An open-source alternative, ***VMware Workstation Player***, similar to ***VirtualBox***, can be employed for private cloud deployments, offering a cost-effective data security solution [15]. ***KVM*** is a virtualization model integrated into the Linux kernel. It supports secure multi-processor hypervisor cores, thereby enhancing data integrity and confidentiality in cloud computing, making it a suitable choice for businesses and SMEs [16]. ***Hyper-V***, a Microsoft virtualization solution, has been found to be vulnerable to DoS attacks, allowing users running malicious code on a guest VM to cause a disruption. To ensure data confidentiality in multi-client environments, encryption can be integrated to prevent unauthorized data access [17]. While ***VirtualBox***, an open-source hosted hypervisor, is useful for educational and research purposes, it presents security risks, such as being a potential single point of failure and being susceptible to DoS attacks, access takeover by low-privileged users, and unauthorized access to the entire ***VirtualBox*** data [18]. ***QEMU***, another open-source hosted hypervisor used in this study for GNS3 network emulation, demonstrates robustness and unexploitability when subjected to a TCP amplification attack that targets the three-way handshake (SYN/SYN-ACK/ACK) [19].

Sgandurra et al. [20] investigated the threat modeling of virtualized systems, highlighting the vulnerabilities of hypervisors and the significant impact that cyber-attacks targeting hypervisors can have by allowing hackers to control all hosted VMs. They discussed different levels of attack, including application-level, kernel-level, virtualization layer, hypervisor, and lower-level attacks. Win et al. [21] proposed a security solution based on Mandatory Access Control (MAC) to defend guest VMs and hypervisors from cyber-attacks. Their solution assigns MAC policies reflecting the intended purpose of the VM and notifies the monitoring VM if access is denied, protecting the virtual environment from unauthorized access. They also suggested future opportunities for implementing MAC policies for guest VMs using the least privilege principle and Extensible Markup

Language (XML) to enforce access control across VM migration. Che et al. [22] introduced a Behavior-Based Access Control (BABAC) solution that combines Attribute-Based Access Control (ABAC) and Behavior-Based Access Control (BBAC) for network virtualization environments. This solution captures dynamic user behaviors and attributes to determine access rights, providing identity and authorization management within VMs.

2.2. Vulnerabilities, Exploitation, and Remediation Techniques in Emulated Networks and Virtualized Systems

In this subsection, we highlight the vulnerabilities, exploitation, and remediation techniques of emulated networks and virtualized systems which can be observed in Table 1.

Pearce et al. [23] discussed security issues in virtualization, highlighting the significant risk of implicit trust placed in Virtual Machines (VMs). They also emphasized the dangers associated with VM cloning, stressing the importance of accounting for all VM instances to ensure security. Additionally, the abstraction of VMs from the underlying hardware introduces vulnerabilities that can be exploited by cyber threats. Wu et al. [24] investigated vulnerabilities in virtualized computer networks, particularly the breach of isolation that allows a VM to monitor and access other VMs and the host machine. They also identified a Denial of Service (DoS) vulnerability arising from the sharing of resources, including the CPU, memory, disk, and network, between the VM and the host.

Hyde and Doug [25] highlighted the emergence of a new DoS attack vector in virtualization technology, particularly in cases where excessive resource allocation to a VM has been permitted. Such attacks can lead to the crash of the physical machine hosting the hypervisor and VM files.

Althobaiti et al. [26] focused on ***cross-VM side channel attacks*** that exploit vulnerabilities in VMs. They identified specific attack types, including ***VM hopping***, which results in DoS by making resources unavailable to the user; ***VM escape***, enabling a guest-level VM to attack itself; and ***VM mobility***, allowing a VM to move between physical hosts.

Brooks et al. [27] addressed security risks in Cloud Computing, including exploitation techniques such as footprinting, to analyze the OS platform remotely, botnets for DoS/DDoS attacks on virtualized systems, traversal attacks that compromise data integrity by modifying VM library contents, and malicious code injections through MITM attacks or redirection of communication within the virtualization management agent on the host platform.

Chelladhurai et al. [28] discussed cybersecurity and privacy considerations in Docker-based virtualization and containerization technology. They explored various existing techniques, such as Sockets and the API, Security Hardening, MAC policies, Type Enforcement, Multi-Category Security, and Docker Security Policies, which can mitigate vulnerabilities to cyber-attacks, including DoS.

Lombardi et al. [29] proposed a security architecture for kernel-based VMs that safeguards guest integrity against malware such as viruses and worms. They concluded that implementing their security model can serve as an effective cybersecurity measure for distributed systems based on virtualization architecture.

Wu et al. [30] demonstrated the Prevent Virtual Machine Escape (PVME) solution, which integrates an adaptation of the Bell–LaPadula Access Control (AC) model with virtualization technology to secure and prevent ***VM Escape Attacks***. This solution protects VMs on the same host through a core component in the hypervisor, containing an Access Matrix, and another component in the host OS. The VM Escape Attack consists of two phases: "Placement", where attackers place malicious VMs as files on the same physical machine as the hypervisor, and "Information Extraction", where the attacker extracts information from other VMs using the access key permissions of the hypervisor. The malicious applications aid the attacker in privilege escalation within VMs in IaaS, SaaS, or PaaS solutions.

Table 1. Cyber Attack Landscape in Virtualization Technologies.

Literature Reviewed	Vulnerability	Exploitation	Remediation Techniques
Pearce et al., 2013 [23]	Overloading network interface cards using a two-layer bridge, resulting in all traffic passing through it.	Compromised VMs used to launch attacks against other VMs or the hypervisor, spreading attacks across networks (Hyperjacking, Hyperjumping).	Host intrusion detection system: monitors specific activities and communications, audits file integrity, identifies insider threats, and detects modifications of file permissions.
Wu et al., 2010 [24]	Virtual network sniffing and virtual network spoofing.	"Bridge Mode" creates a virtual hub for network communication, allowing VMs to sniff the virtual network using tools such as Wireshark, leading to denial of service attacks.	Firewall and shared network layer: firewalls prevent spoofing attacks by identifying the network ID specified in the configuration file. A Shared Network Layer can block communication between shared VMs.
Hyde and Doug 2009 [25]	Unauthorized access to VM contents through file-level vulnerabilities and unrestricted resource allocation, resulting in VM crashes.	Unauthorized users with inappropriate file permissions can steal and observe VM contents. Improper resource allocation during VM creation can lead to theft and denial of service attacks.	Restricting file permissions and implementing appropriate resource allocation can secure against these attacks.
Althobaiti et al., 2017 [26]	Software-based vulnerabilities in VMs, including VM hopping, VM Escape, and VM mobility.	VM hopping can cause denial of service by blocking user access to resources. VM Escape allows a Guest VM to attack the host. VM mobility across physical machines can result in data breaches.	Access control solutions can restrict access to the VCCI by implementing Mandatory Access Control (MAC), Discretionary Access Control (DAC), Role-Based Access Control (RBAC), and virtual firewalls that provide packet filtering and monitoring services.
Brooks et al., 2012 [27]	Footprinting, botnets, hypervisor traversal attacks, and virtual code injection attacks.	Footprinting identifies virtualized operating systems. Botnets can be exploited in VMs for DDoS attacks. Hypervisor traversal attacks modify contents of VM image libraries. Malicious code can be injected into VMs through MITM attacks.	Security wrappers and application firewalls: reject packets incoming from the Internet containing internal IP addresses in the header and outgoing packets with external IP addresses in the header.
Chelladhurai et al., 2016 [28]	ARP Spoofing and MAC Flooding Attacks due to the bridge forwarding all incoming packets without filtering.	Containers' direct communication with the host kernel facilitates attacks on the host system.	Sockets and API, security hardening, Mandatory Access Control (MAC), Type Enforcement (TE), Multi-Category Security (MCS), Docker security policies, and best practices. Security mechanisms for enhanced security for Docker containers include process, file, system, device, IPC, and network isolation to defend the virtualized environment from denial of service.
Lombardi et al., 2010 [29]	Malware vulnerabilities creating backdoors in the system.	Malware provides remote control of the system through malicious code execution.	Kernel-based virtual machine intrusion detection system: secures VMs by checking data integrity and detecting modification of critical system files and data structures.

Table 1. *Cont.*

Literature Reviewed	Vulnerability	Exploitation	Remediation Techniques
Wu et al., 2017 [30]	VM Escape attacks conducted at the Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS) layers. Malicious applications aid attackers in gaining the highest privilege of VMs.	Attack steps include "Placement" (placing malicious VMs on the same physical machine as the hypervisor) and "Extracting Information" (accessing key permissions of the hypervisor or hosts and extracting information from other VMs, host, and hypervisor).	Bell–LaPadula adaptation for Prevent Virtual Machine Escape (PVME): applies basic security principles and adapts security axioms to prevent VM Escape attacks.
Dong et al., 2019 [31]	VM hopping attacks, where the attacker gains access to VMs on the same hypervisor.	The attacker uses a malicious VM to discreetly access or control other VMs on the hypervisor through communication between them. The attacker may also try to access the host OS, potentially destroying other VMs.	Bell–LaPadula adaptation for PVMH: secures and prevents VM hopping attacks.

Dong et al. [31] proposed the Prevent VM Hopping (PVMH) solution, which also adapts the Bell–LaPadula AC model with virtualization technology to secure and prevent ***VM hopping attacks*** between different VMs in operation. By significantly reducing the risk of attacks, this solution enhances the overall reliability of the computing platform. The Access Matrix is stored as a binary file in the hypervisor, with a backup file stored in the privileged VM.

2.3. Related Work

In this section, we review related works that explore the use of network emulation in different integration approaches, presenting studies similar to our own research.

2.4. Network Performance Comparison

Jimenez et al. [32] compared the performance of network emulation using Mininet with physical networks in terms of bandwidth and jitter. The study found no significant difference in bandwidth between physical and emulated networks, both reaching a maximum value of 10MB/s. However, the jitter results showed a significant improvement in the emulated Mininet model compared to the physical network, with maximum values of 0.028 ms for Mininet and 0.908 ms for the physical topology. Kh et al. [33] compared two network emulators, GNS3 and Mininet, in a data center topology. Their analysis showed that Mininet provided a more-stable and higher bandwidth compared to the GNS3 network. Mininet also demonstrated better resource utilization, resulting in higher speed and service accessibility. Gelberger et al. [34] compared the throughput, latency, and jitter capabilities of OpenFlow and Linux ProGFE. The results indicated that Linux ProGFE had enhanced throughput performance compared to OpenFlow, with similar latency, but lower jitter. The study also showed that a more-complex network with increased flexibility, functionality, and capability did not always result in performance degradation.

2.5. Hybrid System Deployment

Amin et al. [35] observed a growing interest in using network emulation as a replacement for a physical architecture, but some researchers are exploring the combination of both to create a distributed solution. In our study, we conducted experiments on both network emulation and physical networks to understand the potential of replacing physical network nodes and explore this area for businesses. Wang et al. [36] emphasized the technological considerations in deploying an emulated solution, such as coordinating centralized control

and distributed routing. Despite the configuration complexities, utilizing hybrid systems offers advantages, including maximizing network throughput in traffic engineering. In our research, we designed and configured a Distributed System that combines a physical and emulated network with Quality of Service (QoS), capable of running business systems and services.

Galan et al. [37] demonstrated how hybrid systems could provide energy consumption advantages through flexible centralized control. They achieved this by selectively shutting down idle links and switches in the deployed emulation architecture, which can be powered on later when needed. Our Distributed System also offers similar flexibility, allowing specific nodes and network connections to be easily shut down without causing interruptions or affecting the rest of the network. The user-friendly interface of the GNS3 platform facilitates the implementation of this feature. Incremental deployment has been identified as the most-effective approach in deploying a distributed solution. Xu et al. [38] discussed the use of a heuristic algorithm for implementing a Distributed System under budget constraints. Incremental deployment has shown a forty percent improvement in throughput compared to other deployment solutions with a small number of emulated nodes. Our work included a cost matrix that compares network emulation and the physical architecture, enabling interchangeability in the application of a Distributed System model.

Saadeh et al. [39] proposed combining hybrid systems with IoT technology, highlighting it as a promising solution. Their study focused on a privacy-aware IoT architecture that leverages the advantages of network emulation, enabling distributed control functionalities across different planes, including operational, data, and tactical planes. Integrating network emulation and the IoT provides benefits such as enhanced forwarding, caching, security, and integration with legacy IT infrastructure. Our research demonstrated the scalability and interchangeability of devices in a distributed solution through a simple configuration or adding previously configured devices via the GNS3 VM and NIC card on the host server. Additionally, Vissicchio et al. [40] addressed the challenges of incremental deployment, robustness, and scalability in emulated network solutions. These challenges make achieving a fully emulated network solution difficult, which can pose future problems. Hybrid network emulation offers a compromise by combining the benefits of both strategies to mitigate these challenges and provide a robust and scalable architectural solution.

3. Methodology

This section presents the six-step methodology (Figure 1) used in this research:

1. **Literature review**: Firstly, we conducted a comprehensive review of recent and relevant academic literature in the field of network emulation. We explored emerging network emulation platforms and identified opportunities for hybrid emulation configuration and deployment.
2. **Network design and development**: We built and configured a six-node network on different platforms: physical, GNS3, the SEED Internet Emulator, and hybrid, creating a data center topology. To maintain consistency, we utilized Cisco CLM appliances as routers in GNS3, programmatic nodes in the SEED Internet Emulator, and physical devices in the physical network using the OSPF protocol. The data center architecture was chosen for its widespread use and relevance in data center network construction, as highlighted by Luo et al. [41]. This model consisted of three layers: the core layer, distribution layer, and access layer.
   We employed a NAT node to provide outside Internet connectivity using one of the Network Interface Cards (NICs) on the host server. The host server connects to the Internet Service Provider (ISP) through a TP-Link Powerline. When connecting the hybrid system, we used the Cloud Node to connect to other devices using the OSPF protocol for dynamic routing. Additionally, we developed the same network topology in Python using the Internet Emulator platforms and Docker containers. This allowed us to build a six-node emulated network that uses the OSPF protocol, designed to the same specification as the GNS3 network. Once completed, we created a solution

using physical Cisco routers and the GNS3 network with the OSPF protocol, along with a bridged network adapter supporting Network Address Translation (NAT). The network design is shown in Figure 2, representing a data center model that can be further expanded into a Campus Area Network (CAN) model consisting of conceptually different locations across a campus.

3. **Data collection and analysis**: In this step, we employed automated scripts for data collection: using bash for the Internet Emulator under Linux and the proprietary Cisco TCLsh for the GNS3 network and the physical Cisco appliances. These scripts enable systematic and reliable data collection while minimizing measurement errors. We executed the scripts by repeating the measurements 1000 times. Once completed, we conducted a statistical analysis to compute latency metrics such as the minimum, maximum, average, median, and standard deviation, as well as jitter.
4. **Energy efficiency and cost analysis**: In this step, we considered two additional criteria for the networks regarding economic benefits. We compared the energy consumption of all network models and performed a cost analysis to understand the economic implications of adopting these networks in a business context.
5. **Network comparison**: In this step, we compared the performance of the network models and platforms to identify their advantages, constraints, and limitations. By analyzing and presenting the performance criteria, we can make comparisons to the SEED Internet Emulator, GNS3, physical, and hybrid solutions to begin identifying meaningful adoption based on the results.
6. **Evaluation and recommendations**: After completing the experiments and the network comparison, we performed an overall evaluation and provide recommendations based on the comparison of the various network models.

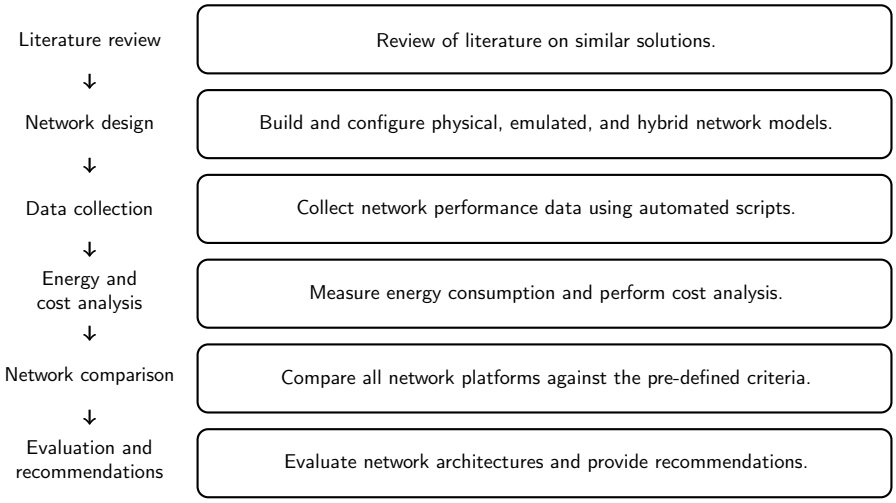

Figure 1. Research Methodology.

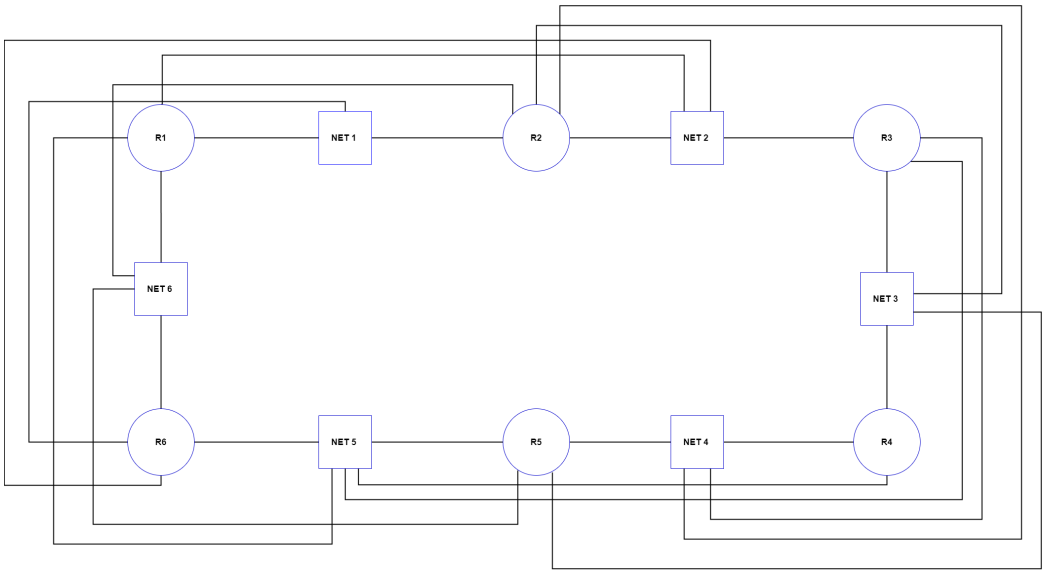

Figure 2. Logical Network Topology.

4. Network Models under Test

This section describes the network models used in the study, all configured in a data center topology and utilizing the OSPF protocol for dynamic routing.

The ***emulated (GNS3)*** model (Figure 2) consisted of six nodes, emulated Cisco routers running IOSv Version 15.5, built using Cisco Modeling Labs (CML). The configuration was performed using the CLI and node support shell processing. The routers employed the OSPF protocol for dynamic routing and established neighbor relationships by utilizing the same Autonomous System (AS) number.

The ***Physical (Cisco)*** network model followed the same structure as the ***emulated (GNS3)*** model, utilizing Cisco 1841 routers and adhering to the data center topology with the OSPF protocol for dynamic routing.

The ***SEED Internet Emulator*** model also consisted of six nodes. This platform provides essential Internet elements as Python classes, generating Docker containers and necessary files to set up the network. In Figure 2, circles represent routers and squares represent subnets. The SEED client is required to interact with the system. Once the network is bootstrapped, events are logged to a file for the verification of successful network instantiation. To modify the emulated network, users must adapt the Python code before re-emulating.

The hybrid network combines the emulated (GNS3) model (Figure 3) with three physical Cisco routers (1841) using a bridged network adapter on the host server to demonstrate the integration of different network models through GNS3 functionality. The system includes a cloud node representing physical Network Interface Cards (NICs). The cloud node, labeled as *Cloud1*, provides access to the physical network to build a distributed solution once the physical equipment is configured and operational.

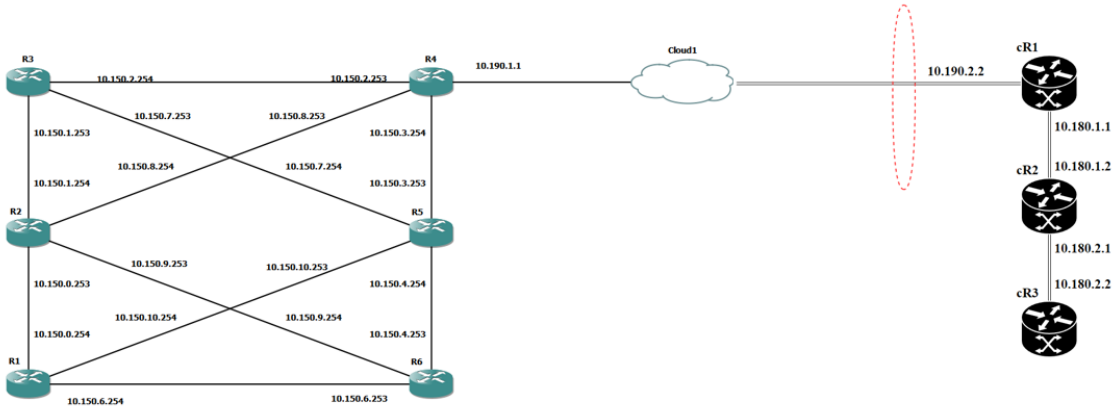

Figure 3. Hybrid System (Physical and GNS3) Network Model.

In Table 2, we present the IP addresses for all of our network models.

Table 2. IP Addresses and Network Interface of Routers.

Router	P. NICs	Physical	E. NICs	GNS3 | SEED Internet Emulator
R1	1	10.180.1.1	3	10.150.0.254 | 10.150.6.254 | 10.150.10.254
R2	2	10.180.1.2/10.180.2.1	4	10.150.0.253 | 10.150.1.254 | 10.150.8.254 | 10.150.9.253
R3	2	10.180.2.2/10.180.3.1	3	10.150.1.253 | 10.150.2.254 | 10.150.7.253
R4		N/A	3	10.150.2.253 | 10.150.3.254 | 10.150.8.253
R5		N/A	4	10.150.3.253 | 10.150.4.254 | 10.150.10.253 | 10.150.7.254
R6		N/A	3	10.150.4.253 | 10.150.6.253 | 10.150.9.254

5. Network Performance Comparison

In this section, we discuss the network performance parameters which can be observed in Table 3 of different emulated network models and the hybrid system model. We compared all network platforms based on performance benchmarks, including the maximum, minimum, average, median, and standard deviation in latency, as well as jitter. These measurements for network performance can be further understood through the work of Gelberger et al. [34], which provides insights into the maximum rate at which data can be transferred, the actual rate at which data are being transferred, any delays between the sender and receiver, and the variation in packet delays. This helped us understand the advantages and disadvantages of each solution and approach, allowing us to assess the efficiency, flexibility, and opportunities associated with each approach.

We list below some established parameters that are typically used to measure network performance.

Table 3. Performance Benchmark Comparison Criteria.

System Benchmarks	Explanation
Bandwidth	Measured in bits/second, it represents the maximum rate at which data can be transferred.
Throughput	The actual rate at which data are being transferred.
Latency	The delay between the sender and receiver in a transmission, which is a function of the signal's travel and processing time at any nodes the information traverses.
Jitter	Variation in packet delay at the receiver end of the communication.
Energy Consumption	The energy consumed when running different network models.
Routing Protocols	A set of defined rules that allow different devices on the network to communicate with each other.

Bandwidth

Bandwidth is the maximum amount of data that can be transmitted over a connection within a specific time frame, measured in bits per second (*bps*). It is often referred to as the "pipe width" for Internet traffic [42].

$$Bandwidth\ (bps) = \frac{Number\ Packets\ Transmissible\ \times\ Packet\ Size\ (bit)}{Transmission\ Time\ (s)} \tag{1}$$

Throughput

Throughput is the rate at which data are actually transmitted over a connection, measured by dividing the amount of information successfully delivered to the destination over a specific time by the duration of the time interval [43].

$$Throughput\ (bps) = \frac{Number\ Packets\ Received\ \times\ Packet\ Size\ (bit)}{Delivery\ Time\ (s)} \tag{2}$$

Latency

Latency is the time it takes for data to travel from the sender to the destination across a network, measured in seconds (s). The overall latency consists of three components: propagation, transmission, and queuing [44].

$$Latency\ (s) = Propagation + Transmit + Queuing \tag{3}$$

Propagation is the time it takes for a signal to travel from the sender to the receiver and is a function of the distance between the endpoints and the speed in the given communication medium.

$$Propagation\ (s) = \frac{Distance\ (m)}{Speed\ (m/s)} \tag{4}$$

Transmission delay considers the actual size of the data and the available bandwidth.

$$Transmit\ (s) = \frac{Packet\ Size\ (bit)}{Bandwidth\ (bps)} \tag{5}$$

Queuing delay considers the waiting times for packets before they can be transmitted.

Jitter

Jitter is the variation in packet delay at the receiver end of the communication. It is computed by measuring n samples of ping time between two endpoints and computing the difference D_i between the individual pings and the average value. As every packet can be routed to different paths to reach the destination, jitter measures the consistency in transmission delay. Therefore, lower jitter values are preferred [45].

$$Jitter\ (s) = \frac{\sum_1^n |D_i|}{n}\ (s) \tag{6}$$

Energy Consumption

Energy consumption is calculated as the product of power in Watts (W) and the time of use in hours (hrs), divided by 1000 to obtain kilowatts (kWh) [46].

$$Energy\ (kWh) = \frac{Power\ (W)\ \times\ Time\ (hrs)}{1000} \tag{7}$$

5.1. Routing Protocols

This subsection provides an overview and comparison of two established routing protocols for achieving dynamic routing: OSPF and EIGRP. It also explains the rationale behind choosing OSPF for adoption.

OSPF vs. EIGRP

The OSPF protocol analyzes the speed, cost, and path congestion, while the Enhanced Interior Gateway Routing Protocol (EIGRP) combines the features of link state routing. The OSPF protocol uses Dijkstra's shortest path algorithm and relies only on bandwidth to calculate the cost of a specific link, whereas the EIGRP uses the Diffusing Update Algorithm (DUAL), which incorporates bandwidth and delay in a composite metric calculation with a complex formula [47]. For our network models, we chose to use the OSPF protocol for the dynamic routing and neighbor relationship. This decision was based on the availability of open-source protocols that can be used across different platforms and facilitates result comparison.

5.2. Methods for Controlling Network Performance

One method to control latency and reduce traffic delay was to deploy Quality of Service (QoS) on networks. This approach has been demonstrated to improve network performance by prioritizing and allocating resources based on specific criteria [48].

There are various ways to reduce delay of a network:

- **Subnetting:** Grouping together endpoints that communicate most frequently can reduce latency across the network.
- **Traffic shaping:** By utilizing the QoS, traffic shaping, and bandwidth allocation, it is possible to improve latency for specific network segments.
- **Load balancing:** Load balancing distributes traffic to areas of the network with the capacity to handle the additional activity.

Applying the QoS enables control over network bandwidth by priority, which can significantly reduce latency on the network in terms of delay time (ms) [49]. To achieve this, the following actions can be taken to control network traffic [50]:

- **Class-map:** Categorize traffic into groups.
- **Policy-map:** Allocate the quantity of bandwidth and priority to the traffic from the class-map.

5.3. Energy Consumption

We conducted energy efficiency measurements on all of our network models to compare their economic benefits for adoption in SME environments. By providing cost analyses, we can identify variables that can be compared to make strategic decisions regarding the network architecture for the organization.

6. Results and Evaluation

In this section, we present the results of evaluating different network software platforms and hybrid models against the benchmark criteria. These evaluations were based on the state-of-the-art in the academic literature. When testing the network nodes, we followed a systematic sequence of distances between the nodes to demonstrate both close and distant proximity.

6.1. Bandwidth and Throughput

In this subsection, we present the results for the bandwidth and throughput of the network models. Figure 4 illustrates that the SEED Internet Emulator exhibited significantly higher bandwidth and throughput compared to the other models. This can be attributed to the model's design, which utilizes Docker containers and tightly integrates all resources. To measure bandwidth and throughput, we employed industry-leading tools in network

monitoring, namely Wireshark and Iperf. When conducting these measurements, we considered the farthest pair of endpoints and repeated the measurement five times.

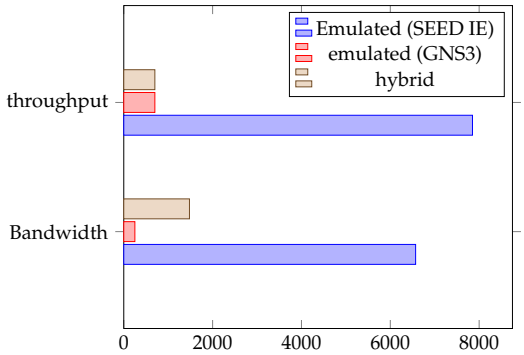

Figure 4. Bandwidth and Throughput Comparison.

6.2. Latency and Jitter

In this subsection, we present and discuss the results of the latency and jitter of the network models (Figure 5). While GNS3 and the hybrid model showed comparable results, it should be noted that the values for the SEED Internet Emulator were so small that can barely be noticed in the chart.

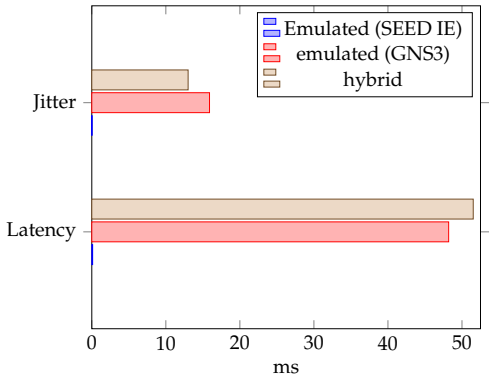

Figure 5. Latency and Jitter Comparison (6 Hops).

6.2.1. GNS3 Network

Table 4 presents the results based on the internal communication paths of the GNS3 platform. The paths chosen were to reflect and observe a variety of distances between the nodes in the network.

Table 4. GNS3 Latency and Jitter (ms).

Nodes	Min Hops	Max	Min	Avg	StdDev	Median	Jitter
R1-R2	1	119.00	13.00	41.45	21.39	35.00	16.99
R1-R3	2	183.00	26.00	70.39	32.79	66.00	24.97
R3-R2	1	134.00	15.00	52.75	23.62	50.00	29.73
R3-R6	2	132.00	21.00	59.55	24.65	54.50	19.41
R1-R6	1	127.00	16.00	48.22	21.24	44.00	15.86
R2-R5	2	138.00	18.00	55.97	23.54	48.50	17.94
R4-R5	1	150.00	15.00	49.62	28.73	43.00	20.57
R4-R6	2	147.00	9.00	50.17	27.62	42.00	20.65
R6-R1	1	155.00	10.00	49.88	27.01	45.50	20.10

The average latency results were around 20 ms for all variations of node communication, which can be observed in Table 5. In the study by Masruroh et al. [51], the OSPF

protocol produced the minimum value for packet loss in their emulated (GNS3) network compared to a variety of other routing protocols. The results in our emulated (GNS3) model were acceptable for network delay. In the study by Nugroho et al. [52], similar experiments were performed in our paper, measuring against the same criteria. They found that the issues in communication based on the link state can be mitigated by using the OpenDaylight controller instead of OSPF to remove reliance on link state principles as the result of a link failure. Moreover, Baggan et al. [53] conducted performance evaluation tests in GNS3 similar to the one conveyed in this paper. Their results concluded that OSPF, compared to other interior gateway protocols such as the Routing Information Protocol (RIP), EIGRP, and iBGP, had the minimum amount of latency while having the maximum amount of latency; in the study, they used eight nodes—a similar quantity to our study. Different network protocols can produce different results in performance [54]. Compared to the RIP and OSPF, the EIGRP demonstrated the minimum delay time for the most-effective communication.

Table 5. SEED Internet Emulator Latency and Jitter (ms).

Nodes	Min Hops	Max	Min	Avg	StdDev	Median	Jitter
R1-R2	1	0.304	0.054	0.110	0.025	0.107	0.016
R1-R3	2	0.579	0.052	0.108	0.030	0.104	0.018
R3-R2	1	0.393	0.053	0.109	0.029	0.104	0.019
R3-R6	2	0.420	0.054	0.109	0.031	0.105	0.020
R1-R6	1	1.040	0.054	0.112	0.042	0.105	0.020
R2-R5	2	0.332	0.052	0.109	0.027	0.105	0.018
R4-R5	1	1.250	0.054	0.118	0.053	0.105	0.026
R4-R6	2	1.980	0.074	0.148	0.071	0.136	0.031
R6-R1	1	0.255	0.055	0.111	0.027	0.107	0.019

6.2.2. SEED Internet Emulator Network

In Table 5, we can observe the results based on the internal communication of the Internet Emulator platform.

The latency results demonstrated excellent delay times, all being around 1 ms on average. Although there have been limited studies performed using the SEED Internet Emulator, a study by Kundel et al. [55] using Programming Protocol-independent Packet Processors (P4) as an emulated SDN model demonstrated achieving active queue management in the data plane, allowing for low latency at the edge and core levels. Additionally, Sedar et al. [56] highlighted achieving low latency and high availability using P4 through programmable data planes in SDN. Moreover, Kaur et al. [57] conveyed that low jitter results were achievable through programmable data planes in the P4 model for the data center application, following the same structure as the models in this paper. Furthermore, we can observe that Mininet [58] can also provide low latency and jitter results through programmability in the data plane, as shown in our results.

6.2.3. Hybrid System

Table 6 presents the results based on the external communication of the GNS3 platform and physical Cisco equipment in our distributed model, where "c" represents physical equipment in the topology.

The study [59] compared OSPF routers and OpenFlow switches for providing the least delay in communication for the Distributed System and found that the OSPF protocol had increased delay times due to the protocol having to calculate the shortest path route in the model. However, our results demonstrated that the average latency delay was around or below 20 ms for all our node communication combinations, which is acceptable. In a study by [60], a Distributed System was presented combining GNS3 and the Open Network Operating System (ONOS), where the latency was observed to be very good, demonstrating the scalability and flexibility to incorporate GNS3 into various other distributed models

while still achieving meaningful results. The future concept of a distributed architecture was conveyed, where the core layer will use physical devices and the edge layer will comprise SD-LANs [61]. Despite the possible opportunity of eventually migrating all networks into network emulation technology, one of the significant issues around this is the significant cost of the infrastructure needed to host the system. The distributed network technology aims to overcome this limitation, as demonstrated in Appendix A. Hybrid systems do provide a reasonable solution for cost-effective integration to solve particular use cases [62].

Table 6. Hybrid System Latency and Jitter (ms).

Nodes	Min Hops	Max	Min	Avg	StdDev	Median	Jitter
R1-cR1	3	192.00	11.00	50.69	29.43	42.00	20.91
R1-cR2	4	171.00	22.00	65.86	26.45	63.00	19.59
R1-cR3	5	107.00	23.00	51.52	17.07	47.50	13.01
R2-cR1	2	162.00	6.00	33.38	27.28	25.50	18.55
R2-cR2	3	147.00	9.00	37.26	19.64	32.00	13.59
R2-cR3	4	141.00	10.00	40.76	23.83	35.00	16.23
R3-cR1	2	201.00	16.00	59.96	30.26	56.50	20.73
R3-cR2	3	78.00	14.00	37.97	14.96	35.50	11.46
R3-cR3	4	149.00	22.00	47.51	23.72	40.50	17.09
R4-cR1	1	197.00	18.00	59.05	29.88	54.50	21.50
R4-cR2	2	156.00	4.00	29.56	26.26	19.00	16.78
R4-cR3	3	105.00	5.00	30.02	18.66	25.50	13.97
R5-cR1	2	165.00	12.00	62.15	29.66	56.00	23.16
R5-cR2	3	168.00	8.00	40.76	23.63	36.00	15.96
R5-cR3	4	152.00	9.00	40.89	20.27	38.00	13.71
R6-cR1	3	248.00	13.00	59.13	37.67	50.50	25.41
R6-cR2	4	159.00	19.00	58.52	25.75	54.00	17.27
R6-cR3	5	144.00	19.00	51.97	22.34	47.50	17.05

While configuring hybrid systems, we noted that emulated networks require a significant amount of time for the initial setup due to additional software configuration, including setting up the network emulation platforms. To expand on this work, we could build a larger-scale hybrid network consisting of both physical and emulated components to observe the results on a larger scale. Additionally, we can explore a more-comprehensive performance comparison of GNS3 and the SEED Internet Emulator, considering different performance criteria such as scalability, flexibility, and security. Future research can also focus on programmable automatic emulation technologies and incorporating security-by-design architectures to explore the opportunities provided by these technologies [63]. One approach to implementing the security architecture into network emulation during the development phase is to use vulnerability assessment tools that can be run simultaneously and automatically in the program configuration, providing enhanced operational security management before the architecture is released and deployed within an organization [64].

6.3. Energy Consumption Comparison

Energy consumption is a significant concern for organizations, and emulated networks have the potential to reduce consumption by efficiently utilizing existing hardware resources. Energy efficiency is a significant advantage, and consolidating all network devices onto a single server can aid in space management.

We collected energy consumption data for all of our network models and observed that the physical network architecture uses significantly less energy than the emulated networks. However, we also collected data on the host server's consumption for the emulation. The consumption was nearly the same as the emulated configuration, indicating that most of the consumption occurs on the host server. The emulation itself does not require a high level of energy. Similar studies have been conducted [65], where they evaluated the energy efficiency of a data center model using the NS-3 platform, yielding positive results.

We used energy monitoring equipment to measure energy consumption which can be observed in Figure 6. Table 7 presents the energy measurements and compares the physical, emulated, and hybrid approaches. We considered a base system ($x1$) and estimated the consumption of a network twice ($x2$) the size with the same components. The physical network had the lowest energy consumption for the base system, while the emulated network consumed approximately 2.4–2.6-times more energy than the physical architecture (1.2–1.5-times for the $x2$ system). We also tested the host server without the emulated network running, which produced almost identical results to the physical system, indicating that the host system absorbs most of the energy. Despite the host system consuming more energy than the physical network, if the server has available capacity, it can be utilized to run additional services for the organization. Therefore, the overall energy balance could still favor the emulated solution. Finally, we tested the hybrid emulated system, which uses both physical and emulated networks.

We found that both network models consume a similar amount of energy. The emulated SEED Internet Emulator model consumed slightly less energy (kWh) than the emulated GNS3 model.

Table 7. Energy Consumption.

Network	**Energy (kWh)**	
(Base System)	$x1$	$x2$
Physical (3 Nodes)	0.037	0.074
Physical (6 Nodes)	0.074	0.148
Emulated (GNS3)	0.151	0.302
Host Server	0.143	0.286
Emulated (Net)	0.008	0.016
Hybrid Emulated System (Emulated/Physical)	0.196	N/A
Emulated/Physical Ratio	*0.077*	*0.154*
Emulated (SEED Internet Emulator)	0.146	0.292
Host Server	0.143	0.286
Emulated (Net)	0.003	0.006
Internet Emulator/Physical Ratio	*0.071*	*0.145*

Energy Data Collection

To measure the energy data, we used the following methods:

1. **Defining energy consumption measurement:** We measured energy consumption in kilowatt-hours (kWh), as it is the universal standard for measuring electricity consumption.
2. **Defining the data collection scope:** We measured the energy consumption of all variations of the network technology used in the study.
3. **Defining the rationale of measuring device choice:** We used the Anglerfish Smart Meter Energy Monitor for data collection.
4. **Defining device implementation:** We simply installed the device in the electric output socket and used an extension cable (consisting of twelve outlets) to provide power to the devices.
5. **Data analysis** We recorded the collected data through the device in Table 7 to observe the variations in network consumption.

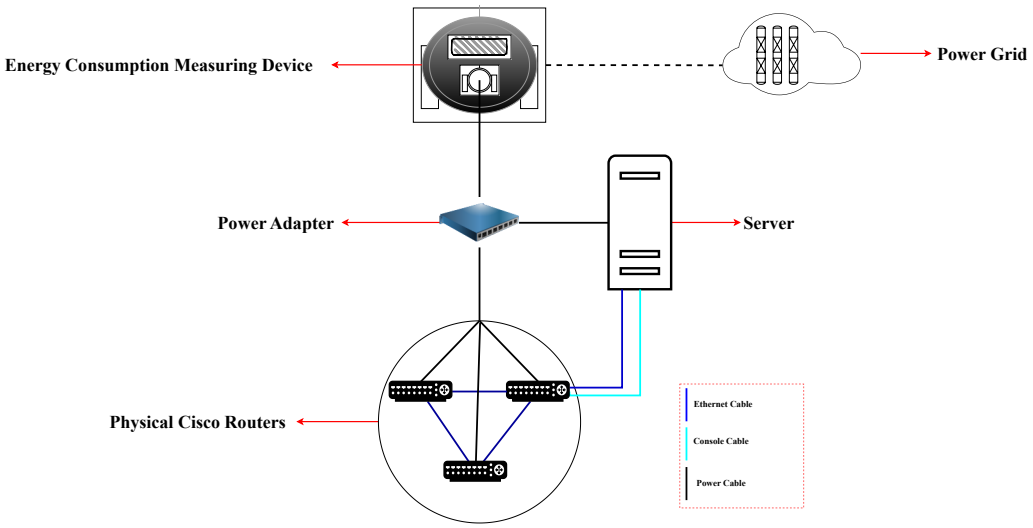

Figure 6. Energy Consumption Measurement Configuration.

6.4. Cost Evaluation

In this subsection, we provide quantitative results based on an estimated price range for Small- and Medium-sized Enterprises (SMEs) to plan their network architecture strategy, adopting an emulated and hybrid architecture.

The cost analysis (Table A2 in Appendix A) compares the network emulation and physical architectures using physical Cisco devices and the GNS3 platform. It is important to note that prices can change over time and are subject to various factors such as the vendor, model, and time of purchase. Our considerations were based on reasonable business application needs for a network similar to the one considered in this paper. Due to financial constraints, we implemented the network using second-hand devices. Comparing physical network architectures is challenging due to the many models, use cases, and time factors involved. For this comparison, we considered new state-of-the-art devices with prices taken from Amazon (U.K.) in November 2022.

In this comparison, the main component of network emulation was the host server (see Table A3 for the detailed configuration), which included two 22-core CPUs with 64 GB of RAM. Theoretically, the network emulation can support up to approximately 35 networking devices, allowing each device one logical core and 1–2 GB of RAM depending on the device's needs. This allows for significant scalability at a total maximum cost of GBP 2899. In contrast, the price for the physical Cisco equipment is GBP 4461. This cost comparison showed that purchasing and deploying the physical Cisco network architecture is significantly more expensive than an emulation solution such as GNS3 and Cisco CLM in this study. One of the most-impactful conclusions from the table is that physical devices are priced for a single item, unlike the Cisco CLM, which provides unlimited devices as long as the host server can provide the resources needed for emulation. This demonstrates a more-scalable and cost-effective solution for SMEs.

6.4.1. Infrastructure

We can observe the economic benefit of network emulation where, when a node reaches its end-of-life stage, it can be deleted from the program rather than physically discarded. Throughout the research, we provided recommendations for an organizational adoption strategy for their future networking architectural strategy. This work considered the challenges and opportunities of deploying a distributed architecture for businesses, either in the startup stages or moving forward with their networking strategy. Through testing the physical, emulated, and hybrid networks, we assessed the benefits and challenges of each option and showed that network emulation technology does not necessarily replace physical infrastructure, but can enhance existing physical networks. Based on our performance criteria, we observed that the physical infrastructure yielded signifi-

cantly better results; however, hybrid systems can allow for enhanced management and cost-effectiveness [66].

6.4.2. Depreciation

From a cost analysis perspective, when using emulated networks, the depreciation of value only needs to be considered for the host server. As time progresses, the server's specification and model will depreciate as newer state-of-the-art technology is developed. In the case of network emulation, this could include factors such as the number of CPU cores, RAM, and more-contemporary releases. Regarding physical networks, as all nodes were physically deployed, we will observe the physical depreciation of models over time as newer variations are released, which can occur as frequently as every three years. When emulating specific network nodes, such as specific Cisco routers in our GNS3 model, as the nodes depreciate, we can simply delete them from the topology, install new appliance images, and add them to the network [67].

6.4.3. Maintenance and Reliability

Computer network maintenance involves monitoring, updating, and running the network to prevent issues from arising. However, it is important to consider the cost implications associated with network maintenance. The cost considerations include the following:

- **Network security**: Ensuring a secure network environment by implementing contemporary and robust defense systems and mechanisms, such as access control, intrusion detection, and firewalls. The cost of acquiring and maintaining security solutions, as well as the associated personnel required to manage them, should be considered. In GNS3, these security measures can be implemented by integrating specific appliances such as pfSense, which can be attached to the network to provide firewall functionality and other security features. Similarly, the SEED Internet Emulator allows the use of programmable code integrated into the emulated network, which enables the deployment of various firewall software and other security solutions. Both platforms offer flexibility in choosing appropriate security solutions to protect the emulated networks effectively. By investing in robust security measures, potential threats can be mitigated, and the overall network resilience and data protection can be significantly enhanced.
- **Network performance**: Ensuring the optimal network speed and reliability of devices, which includes managing bandwidth usage and minimizing delay times.
- **Network scalability**: Ensuring that the network design and nodes can accommodate the operational demands, such as the number of users, locations, and business functions. To achieve this, it is essential to design the networks with scalability in mind. For instance, using open-source communication protocols such as OSPF can facilitate multi-vendor networking, and in GNS3 and the SEED Internet Emulator, designing the network to allow for the easy attachment of additional nodes contributes to scalability.
- **Hardware and software updates**: Regularly updating both the hardware and software elements of the network is crucial for maintaining security, performance, and compatibility. This involves ensuring all hypervisors and Virtual Machines (VMs) are kept up-to-date, along with the platforms running the emulated network.
- **Infrastructure compliance**: Ensuring that the network adheres to relevant policies and legislative requirements. This includes meeting security standards, data protection regulations, and any other compliance obligations applicable to the specific network environment.
- **Network repairs**: Proactively identifying and resolving problems before they escalate is essential for network stability. This can involve implementing measures such as regularly backing up GNS3 project files to prevent data corruption and creating copies of the code used for SEED Internet Emulator deployments, allowing for easy recovery in case of issues.

Computer reliability is a crucial characteristic of any computer-related component, encompassing software, hardware, and networks. A reliable system should demonstrate

dependability, characterized by high uptime, low downtime, and minimal system failure rates. To ensure this, the following metrics can be monitored:

- **System availability**: This metric represents the ratio of the system's actual operating time to the total time it is expected to be available. Ensuring availability in GNS3 requires attention to various factors, such as the GNS3 VM (GNS3 server)—which stores and runs all virtual devices, VMs providing services, and network nodes. Regular audits should be conducted to ensure all components of the emulated network are in optimal condition. On the other hand, the SEED Internet Emulator does not depend on decentralized factors such as external VMs. However, it is essential to take precautions to secure all the relevant files required for emulation. Any accidental removal, corruption, or unauthorized access to these files could potentially lead to system unavailability.
- **Mean time between failures (MTBF)**: The MTBF is the average time between system or component failures. To calculate this, we divide the system's total operating time by the number of downtime incidents that occur. Both GNS3 and the SEED Internet Emulator offer software-based appliances, eliminating the risk of specific device failures that were common in physical appliances. However, since both platforms rely on a single host system for emulation, there is a potential single point of failure. In the event of a server breakdown, whether due to natural causes or a cyber-attack, the entire emulation system could become unavailable. To mitigate this risk, it is crucial to have proper backup and redundancy measures in place to ensure the continuity of operations and minimize downtime.
- **Mean Time To Repair (MTTR)**: This is the measurement of how long it will take for a failed or disabled system component to return to operational. This is calculated by the time period of system downtime and dividing it by the number of downtime incidents. Both GNS3 and the SEED Internet Emulator can suffer from having a particular device failing despite it being in logical rather than physical format; in the case of a broken node, this can be simply fixed by restarting the node virtually or deleting the node and dragging in a new one. One of the significant benefits of doing this is that the logistical element is removed as there is no need to wait for a new device to be delivered; this can be performed in a matter of seconds.
- **Mean Time To Failure (MTTF)**: The MTTF represents the average lifetime period of a system or component that cannot be repaired. It is calculated by adding the total operating time before failure and dividing that by the quantity of these assets in use. Both GNS3 and the SEED Internet Emulator do not face this concern with their network nodes, as more nodes can be easily added to the network, ready to take over from a failing node. However, it is worth noting that the host server itself can be susceptible to this issue.

6.5. Network Security

We now examine the security of the platforms considered in this research by reviewing reported vulnerabilities and the product developers' recommendations for secure deployment. We also considered best practices when using such platforms.

6.5.1. Vulnerabilities

As of May 2023, the GNS3 documentation [68] reported a single vulnerability (CVE-2015-2667) in GNS3 Version 1.2.3. This vulnerability is a search path vulnerability that can allow local users to gain privileges through Trojan horse malware in an unspecified directory. The severity score for this vulnerability is 7.2 out of 10, with complete confidentiality, integrity, and availability impact and low access complexity. Exploiting this vulnerability can result in the disclosure of all files on the system, compromising system integrity, and the complete unavailability of the affected resource. The exploit does not allow an attacker to gain access to the system.

To improve security for GNS3, the developers recommend changing the default password for the GNS3 VM, adding authentication to the GNS3 server, avoiding running GNS3 as the root user or through the Windows administrator, and carefully risk-assessing remote server deployments to mitigate the risk of brute force attacks. They also suggest using a VPN tunnel for information security when running GNS3 remotely, along with SSL or SSH. Additional security measures include rate limiting authentications, disabling unused features, running the server in a container, providing quotas for users, enforcing strict user access privileges, limiting possible binaries for Qemu, using the API instead of Telnet, and binding virtual networks to a specific IP address.

Regarding the SEED Internet Emulator, which uses Docker as the underlying technology, the Docker documentation [69] reports 34 vulnerabilities as of May 2023. These vulnerabilities fall into several categories, including denial of service (4 vulnerabilities), directory traversal (8), code execution (7), bypassing security mechanisms (10), gaining information (2), and gaining privileges (5).

In terms of severity, two are in the range of 9–10 (CVE-2014-9357 and CVE-2019-5736 both exploiting code execution) and 7 in the range of 7–8, covering directory traversal (1), code execution (1), bypassing security mechanisms (1), gaining information (1), and gaining privileges (2).

In detail, the two most-serious vulnerabilities reported are: CVE-2014-9357 (score 10.0) and CVE-2019-5736 (score 9.3). These vulnerabilities are related to code execution, and their exploitation can have a complete impact on confidentiality, integrity, and availability, resulting in the disclosure of all file systems. It also leads to a complete compromise of the system's protection, resulting in the system being fully compromised. Furthermore, it causes a complete shutdown of the affected resource, rendering it completely unusable. To perform these exploits, certain preconditions must be satisfied, and authentication is not required to exploit the vulnerability.

To improve security for Docker, developers recommend focusing on securing the kernel, securing the Docker daemon, eliminating loopholes in the configuration file, and strengthening the security features of the kernel and its interactions with the containers. This can be achieved by using security features such as GRSEC or PAX, which provide additional safety checks at both compile and runtime, and using security model templates and custom policies for Docker containers.

6.5.2. Security Evaluation

We also discuss the best practices for evaluating the security when using the network emulation platforms in this study:

- **Software version**: Ensure that all platform software, including any Virtual Machines (VMs) and hypervisors used, are running the latest and most-stable versions. Regularly updating software helps to address security vulnerabilities and improve overall system performance.
- **Platform security and configuration analysis**: Conduct a thorough evaluation of the platform's configuration settings, network settings, node configurations, and security features. Identify any weak or misconfigured settings that could potentially create vulnerabilities in the system. Addressing these issues will enhance the platform's overall security condition.
- **Access and identity management**: Analyze and assess the effectiveness of the access control mechanisms and identity management processes used by the platform. This includes evaluating username and password combinations, as well as integration with external authentication systems to ensure only authorized users can access the network resources.
- **Analysis of emulated network nodes**: Perform security analysis on emulated components within the platform, such as routers, switches, and VMs. Pay specific attention to the security of different node models, such as Cisco equipment, to identify potential vulnerabilities and address them proactively.

- **Traffic analysis**: Monitor network traffic for anomalies during the testing and production phases. Identify unencrypted communication, unauthorized network traffic, and potentially malicious network activity. Timely detection and response to such incidents can prevent security breaches and data compromises.
- **Vulnerability testing**: Conduct vulnerability testing and analysis by scanning the emulated network for open ports, services, and potential weaknesses. Assess the severity of identified vulnerabilities and take appropriate measures to remediate them promptly.
- **Compliance**: Evaluate relevant industry standards and regulations that the platform should comply with, such as PCI-DSS [70], GDPR [71,72], ISO 27000 Series [73], NIST 800 Series [74], and Network Security Design (SS-018) [75]. Ensure the platform adheres to these standards and assess any vulnerabilities that could impact compliance. Maintaining compliance helps to protect sensitive data and maintain a high level of security within the network.

6.5.3. Cryptographic Mechanisms

There are also specific cryptographic protocols and mechanisms that can be used with both network emulation platforms, at the host and VM/container level, including:

- **Secure Shell (SSH)**: *Host Systems*: Secure Shell (SSH) should be used to secure remote access to the host systems where GNS3 and the SEED Internet Emulator are installed. Ensure that the host systems have been configured to allow SSH connections only from trusted sources and strong authentication methods are enforced. *VMs/containers*: Within GNS3 and the SEED Internet Emulator, SSH can be used to securely access and manage *VMs and containers*. By connecting to the VMs/containers through SSH commands and using IP address filtering, you can control access and protect sensitive configurations.
- **Internet Protocol Security (IPsec)**: *Built-in capabilities*: Both host systems, GNS3, and the SEED Internet Emulator, support IPsec as a suite of protocols for securing communication at the IP layer. Ensure that IPsec is correctly configured on the emulated network devices and *VMs/containers* to encrypt and authenticate network packets, maintaining confidentiality, and integrity.
- **Transport Layer Security (TLS)**: *Built-in capabilities*: GNS3 and the SEED Internet Emulator enable secure communication using TLS. In GNS3, TLS can be utilized for encrypted communication between compatible virtual appliances and machines within the environment. SEED Internet Emulator and TLS: Although the SEED Internet Emulator does not natively provide support for TLS, you can manually configure TLS on the operating system that the SEED Internet Emulator is installed on. This ensures secure communication within the emulator environment.
- **Virtual Private Network (VPN)**: Both platforms can be configured with either OpenVPN or IPsec VPN to establish secure network connections between physical and virtual locations. A VPN server can be set up on a virtual appliance to simulate secure communications between virtual clients and locations.

6.5.4. Summary

In summary, both GNS3 and Docker have reported vulnerabilities that require attention for a secure deployment. By following the recommendations provided by the developers and implementing best practices, these vulnerabilities can be mitigated, leading to an enhanced overall security of the network emulation platforms. Taking proactive steps to address potential security risks will ensure a safer and more-reliable environment for network emulation.

6.6. Usability

According to Nielsen [76], software usability can be assessed by considering the following attributes:

- **Learnability**: The system should be simple to understand, allowing users to start working immediately.
- **Efficiency**: The system should be efficient to use, enabling a high level of productivity once learned.
- **Memorability**: Users should be able to retain their knowledge of the system, allowing for easy re-use after a period of time.
- **Errors**: The system's error rate should be low, preventing users from encountering errors during usage.
- **Satisfaction**: The system should provide a pleasant experience for users and generate satisfaction during interactions.

In the context of network emulation platforms, the usability of GNS3 and the SEED Internet Emulator differs in terms of their interface and intuitiveness. Both platforms can be run on Windows, Linux, and Mac OS, making them versatile options that cater to a wide range of user needs and preferences. Considering Nielsen's attributes, the following observations can be made:

- **Learnability**: GNS3 is known for its ease of use and intuitive interface, making it suitable for professional training programs, such as Cisco certificates or university studies in network specialties. It features a graphical user interface that is simple and similar to Cisco Packet Tracer. On the other hand, the SEED Internet Emulator requires a deeper understanding of Linux system administration, including version control systems (e.g., Git), file permissions, Python programming, and Docker. However, once the initial setup is completed, the web client of the SEED Internet Emulator is straightforward and easy to navigate.
- **Efficiency**: The SEED Internet Emulator provides the opportunity for efficiency through a programmable environment, allowing for the automation of many tasks. This can be a requirement for adding more network nodes or changing communication protocols, which can be achieved through additional or different lines of code.
- **Memorability**: GNS3 presents a more-memorable platform for a casual user, in contrast to the SEED Internet Emulator, which is more relevant to an experienced IT professional [8].
- **Errors**: Challenges can arise when setting up and troubleshooting the GNS3 VM in a hypervisor environment. The software allows for a great amount of user accessibility, depending on how the user wants to configure it. The SEED Internet Emulator presents challenges around correct file permissions, programmable code, and issues with Docker. While most of the errors that can be encountered are trivial and can be mitigated easily, it does require a good knowledge of the Linux OS system administration [77].
- **Satisfaction**: In the short term, GNS3 can provide a more-pleasant experience for learners as it is presented in a GUI for all phases and is generally easy to use for all aspects, including design, development, and configuration [78]. The SEED Internet Emulator can initially be more difficult and complex as there is more system administration, programming, and understanding involved. However, this platform can provide enhanced satisfaction in the long term, as it has the ability to be used for a high variety of use-cases due to the nature of the platform using programming.

However, these general usability considerations for users when choosing between GNS3 and the SEED Internet Emulator need to take into account their specific requirements and expertise level.

7. Conclusions and Future Research Directions

In this study, we conducted a comparison between two emerging network emulation platforms: GNS3 and the SEED Internet Emulator. Our evaluation focused on performance criteria such as the bandwidth, throughput, latency, and jitter. While both platforms share a common concept, they differ significantly in terms of design, development, and execution.

Through this research, Sections 2 and 6 are linked to the first research contribution, which highlights related literature studies around virtualization technologies, networking, and cybersecurity applications. Sections 5 and 6 are linked to the second research contribution, demonstrating the network performance comparison of the systems. Section 4 demonstrates the third research contribution, showcasing the network models under test.

GNS3, being more heavyweight, offers greater integration flexibility and mimics a typical network environment. It facilitates easy connections of external nodes and services from multiple vendors. Conversely, the SEED Internet Emulator is more lightweight, flexible, and efficient in its development approach.

By exploring the integration of network emulation platforms into the data center architecture and creating a hybrid emulation system, we utilized the SEED Internet Emulator and GNS3 to emulate networks using the host system's resources. Our results indicated that, while GNS3 possesses certain advantages as a platform, there are notable bottlenecks to consider when incorporating it into a company's infrastructure strategy.

GNS3 encounters issues with low bandwidth and throughput speeds, even when the Internet Service Provider (ISP) allows for higher speeds. Additionally, due to the platform's nature and its method of achieving network emulation, one of its main drawbacks is the challenges associated with allocating a large number of CPU cores to the network appliances. Furthermore, our observations revealed that the high energy consumption of the host system poses a significant constraint to energy efficiency, which should be considered during the design phase. In fact, our study demonstrated that, for small networks, the energy consumption of the host system alone can outweigh the physical energy consumption.

On the other hand, the SEED Internet Emulator provides a lightweight emulation platform that exhibits flexibility in development and performs impressively across all performance criteria.

Author Contributions: Conceptualization, L.G. and P.M.; methodology, L.G. and P.M.; software, L.G. and P.M.; validation, L.G., P.M. and V.C.; formal analysis, L.G. and P.M.; investigation, L.G. and P.M.; resources, L.G., P.M. and V.C.; data curation, L.G. and P.M.; writing—original draft preparation, L.G. and P.M.; writing—review and editing, L.G., P.M. and V.C.; visualization, L.G. and P.M.; supervision, P.M. and V.C.; project administration, L.G. and P.M.; funding acquisition, L.G., P.M. and V.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by VC Research (VCR 0000203).

Data Availability Statement: No new data was created, apart from the one presented in this paper.

Acknowledgments: This paper contributes to part of Lewis Golightly's Ph.D. thesis.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CML	Cisco Modeling Labs
WAN	Wide Area Network
QoS	Quality of Service
GNS3	Graphical Network Simulator-3
SEED IE	SEED Internet Emulator
BGP	Border Gateway Protocol
iBGP	internal Border Gateway Protocol
eBGP	external Border Gateway Protocol
NAT	Network Address Translation
ICMP	Internet Control Message Protocol
SME	Small–Medium Enterprise
GUI	Graphical User Interface
OOP	Object-Oriented Programming

OSPF	Open Shortest Path First
IMUNES	Integrated Multiprotocol Network Emulator/Simulator
NIC	Network Interface Cards
RIP	Routing Information Protocol
P4	Programming Protocol-independent Packet Processors
OPOS	Open Network Operating System
DUAL	Diffusing Update Algorithm
EIGRP	Enhanced Interior Gateway Routing Protocol
AS	Autonomous System
IOS	(Cisco) Internetworking Operating System
kWh	kilowatt-hour
DoS	Denial of Service
VM	Virtual Machine
XML	Extensible Markup Language
XACML	Extensible Access Control Markup Language
AC	Access Control
MAC	Mandatory Access Control
MITM	Man-In-The-Middle
SSH	Secure Shell
IPsec	Internet Protocol Security
ISP	Internet Service Provider
TLS	Transport Layer Security
VPN	Virtual Private Network
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
MTTF	Mean Time To Failures

Appendix A. Requirements and Cost Analysis

Table A1. GNS3 Software and Hardware Requirements.

CPU	CPU	RAM	HDD	Network	Other
Windows 7 (64-bit)	2 or more logical cores	4 GB	1 GB	Cisco CLM Appliances	Virtualbox VMware

Table A2. Physical vs. Emulated Infrastructure Cost Analysis (November 2022).

Emulation Components	Cost (GBP)	Physical Components	Cost (GBP)
Physical Server (Dell R610 Spec)	1995.60	Cisco Firewall (ASA 5512-X)	2510.33
Windows Server 2019 OS	729.58	Cisco Router (C1111-4P)	939.75
CLM Appliances	173.71	Cisco Switch (WS-C2960S-48TS-S)	1000.00
GNS3 Software and Virtual Machine	FOSS 0.00	Console Cable	6.99
VMware Hypervisor	FOSS 0.00	Cat 6 Ethernet Cable	4.04
Putty Software	FOSS 0.00		
Total (£)	2898.89	**Total (£)**	4461.11

Table A3. Physical Server Specification and Cost (November 2022).

Component	Description	Cost (GBP)
Processor *(x2)*	Intel Xeon E5-2699v4 (2.2 GHz/22-core/55 MB/145 W)	1080.00
Memory (RAM)	64 GB (4 × 16 GB) PC4-17000R Dual Rank Memory	206.40
NIC 1	Dell Qlogic QLE2526 8 GB Fibre Channel Dual Port PCIe	61.20
NIC 2	Dell Intel Pro/1000 VT Quad Port 1 Gbit RJ45 Ethernet PCIe	90.00
Hard Drive (*x2*)	Dell 2 TB 7.2K 3G SATA 3.5″ Hotswap Hard Drive	134.40
Other	Power Supply + Case + Monitor + Mouse + Keyboard	369.90
Total		1995.60

Appendix B. Python Code—OSPF10.py

```
#!/usr/bin/env python3
# encoding: utf-8

from seedemu.layers import Base, Routing, Ebgp, PeerRelationship, Ibgp, Ospf
from seedemu.services import WebService
from seedemu.core import Emulator, Binding, Filter
from seedemu.compiler import Docker

emu = Emulator()

base = Base()
routing = Routing()
ebgp = Ebgp()
ibgp = Ibgp()
ospf = Ospf()
web = WebService()
################################################

# Create and set up the transit AS (AS-10)

as150 = base.createAutonomousSystem(150)

# Create 6 internal networks
as150.createNetwork('net1')
as150.createNetwork('net2')
as150.createNetwork('net3')
as150.createNetwork('net4')
as150.createNetwork('net5')
as150.createNetwork('net6')
################################################

# Create routers and link the routes in a Data Centre structure:

# r1 <--> r2  <--> r6  <-->  r5
# r2 <--> r3  <--> r1  <-->  r4  <--> r6
# r3 <--> r4  <--> r2  <-->  r5
# r4 <--> r5  <--> r3  <-->  r2
# r5 <--> r6  <--> r4  <-->  r3  <--> r1
# r6 <--> r1  <--> r5  <-->  r2

as150.createRouter('r1').joinNetwork('net1').joinNetwork('net2').joinNetwork(
    'net6').joinNetwork('net5')
as150.createRouter('r2').joinNetwork('net2').joinNetwork('net3').joinNetwork(
    'net1').joinNetwork('net4').joinNetwork('net6')
as150.createRouter('r3').joinNetwork('net3').joinNetwork('net4').joinNetwork(
    'net2').joinNetwork('net5')
as150.createRouter('r4').joinNetwork('net4').joinNetwork('net5').joinNetwork(
    'net3').joinNetwork('net2')
as150.createRouter('r5').joinNetwork('net5').joinNetwork('net6').joinNetwork(
    'net4').joinNetwork('net3').joinNetwork('net1')
as150.createRouter('r6').joinNetwork('net6').joinNetwork('net1').joinNetwork(
    'net5').joinNetwork('net2')
################################################

emu.addLayer(base)
emu.addLayer(routing)
emu.addLayer(ebgp)
emu.addLayer(ibgp)
emu.addLayer(ospf)
emu.addLayer(web)

################################################
# Save the emulation as a component (can be reused by other emulation)

emu.dump('base-component.bin')

################################################
```

```
# Generate the docker file

emu.render()
emu.compile(Docker(), './output')
```

References

1. Tancevski, L. SDN concept: From theory to network implementation. In *Optical Fiber Communication Conference*; Optica Publishing Group: Washington, NW, USA, 2014; p. W1E–3.
2. Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2014**, *103*, 14–76. [CrossRef]
3. Fernandez-Fernandez, A.; Cervello-Pastor, C.; Ochoa-Aday, L. Achieving Energy Efficiency: An Energy-Aware Approach in SDN. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–7. [CrossRef]
4. Assefa, B.G.; Özkasap, Ö. A survey of energy efficiency in SDN: Software-based methods and optimization models. *J. Netw. Comput. Appl.* **2019**, *137*, 127–143. [CrossRef]
5. Ahmad, S.; Mir, A.H. Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers. *J. Netw. Syst. Manag.* **2021**, *29*, 1–59. [CrossRef]
6. Khorsandroo, S.; Sánchez, A.G.; Tosun, A.S.; Arco, J.M.; Doriguzzi-Corin, R. Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Comput. Netw.* **2021**, *192*, 107981. [CrossRef]
7. Blake, S.; Zhang, Q.; Birkner, R.; Hahm, O.; Jarray, M. Security in Software-Defined Networking: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 623–646.
8. Du, W.; Zeng, H.; Won, K. SEED emulator: An Internet Emulator for research and education. In Proceedings of the 21st ACM Workshop on Hot Topics in Networks, Austin, TX, USA, 14–15 November 2022; pp. 101–107.
9. Zhang, K.; Zhao, X.; Peng, Y.; Yan, K.; Sun, P. Analysis of Mobile Communication Network Architecture Based on SDN. *J. Grid Comput.* **2022**, *20*, 28. [CrossRef]
10. Daniels, J. Server virtualization architecture and implementation. *XRDS Crossroads Acm Mag. Stud.* **2009**, *16*, 8–12. [CrossRef]
11. Lai, J.; Tian, J.; Zhang, K.; Yang, Z.; Jiang, D. Network emulation as a service (neaas): Towards a cloud-based network emulation platform. *Mob. Netw. Appl.* **2021**, *26*, 766–780. [CrossRef]
12. Sharma, P.; Chaufournier, L.; Shenoy, P.; Tay, Y. Containers and virtual machines at scale: A comparative study. In Proceedings of the 17th International Middleware Conference, Trento, Italy, 12–16 December 2016; pp. 1–13.
13. Blenk, A.; Basta, A.; Reisslein, M.; Kellerer, W. Survey on network virtualization hypervisors for software defined networking. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 655–685. [CrossRef]
14. Bauman, E.; Ayoade, G.; Lin, Z. A survey on hypervisor-based monitoring: Approaches, applications, and evolutions. *ACM Comput. Surv. (CSUR)* **2015**, *48*, 1–33. [CrossRef]
15. Sharma, K. An alleviated model for private cloud deployment using VMware. In Proceedings of the 2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC), Indore, India, 17–19 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–3.
16. Li, S.W.; Li, X.; Gu, R.; Nieh, J.; Hui, J.Z. A secure and formally verified Linux KVM hypervisor. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1782–1799.
17. Durrani, A. Analysis and prevention of vulnerabilities in cloud applications. In Proceedings of the 2014 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, Pakistan, 12–13 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 43–46.
18. Khan, R.; AlHarbi, N.; AlGhamdi, G.; Berriche, L. Virtualization Software Security: Oracle VM VirtualBox. In Proceedings of the 2022 Fifth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU), Riyadh, Saudi Arabia, 28–29 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 58–60.
19. Nguyen, S.D.; Mimura, M.; Tanaka, H. SVTester: Finding DoS vulnerabilities of virtual switches. *J. Inf. Process.* **2021**, *29*, 581–591. [CrossRef]
20. Sgandurra, D.; Lupu, E. Evolution of attacks, threat models, and solutions for virtualized systems. *ACM Comput. Surv. (CSUR)* **2016**, *48*, 1–38. [CrossRef]
21. Win, T.Y.; Tianfield, H.; Mair, Q. Virtualization security combining mandatory access control and virtual machine introspection. In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 8–11 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1004–1009.
22. Che, Y.; Yang, Q.; Wu, C.; Ma, L. BABAC: An access control framework for network virtualization using user behaviors and attributes. In Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, Hangzhou, China, 18–20 December 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 747–754.
23. Pearce, M.; Zeadally, S.; Hunt, R. Virtualization: Issues, security threats, and solutions. *ACM Comput. Surv. (CSUR)* **2013**, *45*, 1–39. [CrossRef]

24. Wu, H.; Ding, Y.; Winer, C.; Yao, L. Network security for virtual machine in cloud computing. In Proceedings of the 5th International Conference on Computer Sciences and Convergence Information Technology, Seoul, Republic of Korea, 30 November–2 December 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 18–21.
25. Hyde, D. A Survey on the Security of Virtual Machines. 2009. Available online: http://www.cse.wustl.edu/~jain/cse571-09/ftp/vmsec/index.html (accessed on 3 June 2023).
26. Althobaiti, A.F.S. Analyzing security threats to virtual machines monitor in cloud computing environment. *J. Inf. Secur.* **2017**, *8*, 1. [CrossRef]
27. Brooks, T.T.; Caicedo, C.; Park, J.S. Security vulnerability analysis in virtualized computing environments. *Int. J. Intell. Comput. Res.* **2012**, *3*, 277–291. [CrossRef]
28. Chelladhurai, J.; Chelliah, P.R.; Kumar, S.A. Securing docker containers from denial of service (dos) attacks. In Proceedings of the 2016 IEEE International Conference on Services Computing (SCC), San Francisco, CA, USA, 27 June–2 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 856–859.
29. Lombardi, F.; Di Pietro, R. A security management architecture for the protection of kernel virtual machines. In Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology, Bradford, UK, 29 June–1 July 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 948–953.
30. Wu, J.; Lei, Z.; Chen, S.; Shen, W. An access control model for preventing virtual machine escape attack. *Future Internet* **2017**, *9*, 20. [CrossRef]
31. Dong, Y.; Lei, Z. An access control model for preventing virtual machine hopping attack. *Future Internet* **2019**, *11*, 82. [CrossRef]
32. Jimenez, J.M.; Romero Martínez, J.O.; Rego Máñez, A.; Lloret, J. Analyzing the performance of software defined networks vs real networks. *Int. J. Adv. Netw. Serv.* **2016**, *9*, 107–116.
33. Kh, D.R.; Botirov, S.; Juraev, F. A simulation model of a cloud data center based on traditional networks and Software-defined network. In Proceedings of the 2021 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 3–5 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–4.
34. Gelberger, A.; Yemini, N.; Giladi, R. Performance analysis of software-defined networking (SDN). In Proceedings of the 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, USA, 14–16 August 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 389–393.
35. Amin, R.; Reisslein, M.; Shah, N. Hybrid SDN networks: A survey of existing approaches. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3259–3306. [CrossRef]
36. Wang, W.; He, W.; Su, J. Boosting the benefits of hybrid SDN. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2165–2170.
37. Galán-Jiménez, J.; Polverini, M.; Lavacca, F.G.; Herrera, J.L.; Berrocal, J. Joint energy efficiency and load balancing optimization in hybrid IP/SDN networks. *Ann. Telecommun.* **2022**, *78*, 13–31. [CrossRef]
38. Xu, H.; Li, X.Y.; Huang, L.; Deng, H.; Huang, H.; Wang, H. Incremental deployment and throughput maximization routing for a hybrid SDN. *IEEE/ACM Trans. Netw.* **2017**, *25*, 1861–1875. [CrossRef]
39. Saadeh, H.; Almobaideen, W.; Sabri, K.E.; Saadeh, M. Hybrid SDN-ICN architecture design for the Internet of things. In Proceedings of the 2019 Sixth International Conference on Software Defined Systems (SDS), Rome, Italy, 10–13 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 96–101.
40. Vissicchio, S.; Vanbever, L.; Bonaventure, O. Opportunities and research challenges of hybrid software defined networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 70–75. [CrossRef]
41. Luo, S.; Xing, H.; Li, K. Near-optimal multicast tree construction in leaf-spine data center networks. *IEEE Syst. J.* **2019**, *14*, 2581–2584. [CrossRef]
42. Jimson, E.R.; Nisar, K.; bin Ahmad Hijazi, M.H. Bandwidth management using software defined network and comparison of the throughput performance with traditional network. In Proceedings of the 2017 International Conference on Computer and Drone Applications (IConDA), Kuching, Malaysia, 9–11 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 71–76.
43. Basagni, S.; Petrioli, C.; Petroccia, R.; Stojanovic, M. Choosing the packet size in multi-hop underwater networks. In Proceedings of the OCEANS'10 IEEE SYDNEY, Sydney, NSW, Australia, 24–27 May 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 1–9.
44. Kuzlu, M.; Pipattanasomporn, M.; Gurses, L.; Rahman, S. Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability. In Proceedings of the 2019 IEEE international conference on blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 536–540.
45. Balestrieri, E.; Picariello, F.; Rapuano, S.; Tudosa, I. Review on jitter terminology and definitions. *Measurement* **2019**, *145*, 264–273. [CrossRef]
46. Matthews, H.S.; Hendrickson, C.T.; Chong, H.M.; Loh, W.S. Energy impacts of wired and wireless networks. In Proceedings of the Conference Record 2002 IEEE International Symposium on Electronics and the Environment (Cat. No. 02CH37273), San Francisco, CA, USA, 6–9 May 2002; IEEE: Piscataway, NJ, USA, 2002; pp. 44–48.
47. Vetriselvan, V.; Patil, P.R.; Mahendran, M. Survey on the RIP, OSPF, EIGRP routing protocols. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 1058–1065.
48. de Souza, F.R.; Miers, C.C.; Fiorese, A.; de Assunção, M.D.; Koslovski, G.P. Qvia-sdn: Towards qos-aware virtual infrastructure allocation on sdn-based clouds. *J. Grid Comput.* **2019**, *17*, 447–472. [CrossRef]
49. Held, G. *Quality of Service in a Cisco Networking Environment*; John Wiley & Sons: New York City, NY, USA, 2002.

50. Shukla, V.H.; Deshmukh, S.B. Implementing QOS Policy in MPLS Network. *Int. J. Comput. Appl.* **2015**, *975*, 8887.
51. Masruroh, S.U.; Fiade, A.; Iman, M.F.; Amelia. Performance evaluation of routing protocol RIPv2, OSPF, EIGRP with BGP. In Proceedings of the 2017 International Conference on Innovative and Creative Information Technology (ICITech), Salatiga, Indonesia, 2–4 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7.
52. Nugroho, A.S.; Safitri, Y.D.; Setyawan, T.A. Comparison analysis of software defined network and OSPF protocol using virtual media. In Proceedings of the 2017 IEEE International Conference on Communication, Networks and Satellite (Comnetsat), Semarang, Indonesia, 5–7 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 106–111.
53. Baggan, V.; Chaturvedi, S.P.; Snehi, J.; Snehi, M. An Efficient Model of IGP for Network-based Communication: A Comparison. In Proceedings of the 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, 10–11 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 223–228.
54. Biradar, A.G. A comparative study on routing protocols: RIP, OSPF and EIGRP and their analysis using GNS-3. In Proceedings of the 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, 1–3 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5.
55. Kundel, R.; Blendin, J.; Viernickel, T.; Koldehofe, B.; Steinmetz, R. P4-codel: Active queue management in programmable data planes. In Proceedings of the 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 27–29 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–4.
56. Sedar, R.; Borokhovich, M.; Chiesa, M.; Antichi, G.; Schmid, S. Supporting emerging applications with low-latency failover in P4. In Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies, Budapest, Hungary, 20 August 2018; pp. 52–57.
57. Kaur, S.; Kumar, K.; Aggarwal, N. A review on P4-Programmable data planes: Architecture, research efforts, and future directions. *Comput. Commun.* **2021**, *170*, 109–129. [CrossRef]
58. Rezaee, M.; Moghaddam, M.H.Y. SDN-based quality of service networking for wide area measurement system. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3018–3028. [CrossRef]
59. Khan, A.A.; Zafrullah, M.; Hussain, M.; Ahmad, A. Performance analysis of OSPF and hybrid networks. In Proceedings of the 2017 International Symposium on Wireless Systems and Networks (ISWSN), Lahore, Pakistan, 19–22 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–4.
60. Arifwidodo, B.; Oktavian, D.A.; Ginting, J.G.A. The Performance Analysis of Hybrid SDN–IP Reactive Routing on ONOS Controller in Tree Topologies. In Proceedings of the 2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), Solo, Indonesia, 3–5 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 118–122.
61. Salman, O.; Elhajj, I.H.; Chehab, A.; Kayssi, A. QoS guarantee over hybrid SDN/non-SDN networks. In Proceedings of the 2017 8th International Conference on the Network of the Future (NOF), London, UK, 22–24 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 141–143.
62. Al-Harbi, A.; Bahnasse, A.; Louhab, F.E.; Talea, M. Towards an efficient resource allocation based on software-defined networking approach. *Comput. Electr. Eng.* **2021**, *92*, 107066. [CrossRef]
63. Shirmarz, A.; Ghaffari, A. Automatic Software Defined Network (SDN) performance management using topsis decision-making algorithm. *J. Grid Comput.* **2021**, *19*, 16. [CrossRef]
64. Ur-Rehman, A.; Gondal, I.; Kamruzzaman, J.; Jolfaei, A. Vulnerability modelling for hybrid industrial control system networks. *J. Grid Comput.* **2020**, *18*, 863–878. [CrossRef]
65. Dhiab, I.; Barouni, Y.; Khalfallah, S.; Ben Hadj Slama, J. Performance evaluation of a hybrid IP/SDN network in data centre network architectures. *IET Commun.* **2019**, *13*, 1185–1191. [CrossRef]
66. De Oliveira, R.L.S.; Schweitzer, C.M.; Shinoda, A.A.; Prete, L.R. Using mininet for emulation and prototyping software-defined networks. In Proceedings of the 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, Colombia, 4–6 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–6.
67. Chen, Y.; Chen, Y.; Cao, Q.; Yang, X. PacketCloud: A cloudlet-based open platform for in-network services. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *27*, 1146–1159. [CrossRef]
68. GNS3. GNS3 Security. 2023. Available online: https://docs.gns3.com/docs/using-gns3/administration/gns3-security (accessed on 2 June 2023).
69. Docker. Docker Docs. 2023. Available online: https://docs.docker.com/engine/security (accessed on 2 June 2023).
70. PCI Security Standards Council. Payment Card Industry Data Security Standard. Available online: https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Standard/PCI-DSS-v4_0.pdf (accessed on 20 July 2023).
71. European Parliament; Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation). 2016. Available online: https://data.europa.eu/eli/reg/2016/679/oj (accessed on 20 July 2023).
72. UK Government. Data Protection Act 2018. 2018. Available online: https://www.legislation.gov.uk/ukpga/2018/12/pdfs/ukpga_20180012_en.pdf (accessed on 20 July 2023).
73. International Organization for Standardization. ISO/IEC 27001:2022(en) Information Security, Cybersecurity and Privacy Protection. 2022. Available online: https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-3:v1:en (accessed on 20 July 2023).

74. National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations. 2022. Available online: https://doi.org/10.6028/NIST.SP.800-53r5 (accessed on 20 July 2023).
75. Chief Security Office, Department of Work and Pensions (UK). Security Standard Network Security Design (SS-018). 2020. Available online: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/882774/dwp-ss018-security-standard-network-security-design-v1.4.pdf (accessed on 20 July 2023).
76. Nielsen, J. *Usability Engineering*; Morgan Kaufmann: Cambridge, MA, USA, 1994.
77. Zeng, H. SEEDEMU: The SEED Internet Emulator. Ph.D. Thesis, Syracuse University, Syracuse, NY, USA, 2021.
78. Wangchuk, T. Study on the usability of GNS3 for teaching and learning system and network administration. *Int. J. Sci. Technol. Eng.* **2018**, *4*, 34–37.