Teaching Computer Networks by Building Networks

An Experiential Learning Approach Using Docker

Ram P Rustagi
University of Maryland Baltimore County
Baltimore, USA
rprustagi@umbc.edu

Abstract

Most Computer Networking curricula at the undergraduate level aim to familiarize students with key networking technologies and standards traditionally taught through a combination of lectures, analytical exercises, simulations, and packet trace analysis. These approaches help students deelop a basic understanding of inherently complex yet fundamental concepts, such as, complexities of web communication, reliable delivery and packet forwarding in IP networks etc. Many students struggle to connect theoretical models with actual protocol behavior observed in real systems and often unable to translate their limited understanding into diagnosis and action when troubleshooting networks. This paper presents a container-based experiential learning approach based on a four-stage pedagogical cycle: Prediction, Implementation, Observation, and Explanation (PIOE). The framework uses Docker-based networking laboratory, in which routers, switches and end hosts are instantiated as docker containers. It enables students to experimentally validate protocol behavior across multiple layers of the networking stack. With the increasing adoption of generative AI in education, the framework emphasizes experiential learning by requiring students to relate conceptual understanding to the observed behavior of actual network implementations.

We illustrate the framework using representative assignments covering Network Delay, TCP Flow Control, TCP Fairness, Longest Prefix Match based forwarding, IPv6 deployment and Layer-2 switching. Preliminary classroom observations and feedback collected from our students confirm that these real time experiential activities improve student engagement and facilitate deeper understanding of networking concepts.

CCS Concepts

• **Applied computing → Interactive learning environments.**.

Keywords

Computer Networks, Experiential Learning, Docker containers.

1 Introduction

Computer networking courses often present students with a significant challenge of how to relate network protocol specifications to the behavior of actual networked systems. Classroom teaching helps students solve textbook [11, 13] problems involving networking concepts: delay calculations, IP addressing and subnetting, forwarding and routing decisions, TCP protocol [6] behavior, TCP congestion and flow control [1] etc., yet students struggle to explain how these concepts manifest in a running network. Typical academia laboratory environments partially address this issue through packet captures, protocol analyzers, network simulators and emulators, providing some avenues for students to observe real network behavior.

Experiential learning theory emphasizes that knowledge is constructed through the transformation of experience into understanding. According to Kolb [10], *"experiential learning is a particular form of learning from life experience"* and keeps the learner *"directly in touch with the realities being studied"*. In networking education, students learn protocol concepts through lectures and analytical exercises and improve their understanding by implementing network configurations, observing protocol behavior and reflecting on the relationship between theoretical predictions and observed outcomes. There exist many open source courses with practical exercises on the internet. The course by UCLovian University [2] provides hands on exercises with network protocols and implementation of mini-internet that mimics the real internet [9]. The accompanying website for Network textbook by Kurose and Ross [11], provides interactive exercises for each protocol layer in TCP/IP stack. These interactive exercises are very helpful in assimilating network concepts. The approach adopted by this author is to help students' learning by implementing a series of assignments involving real network entities and analyze real protocol behavior at every layer as the course progresses.

Open availability of containerization technologies have made it possible to create lightweight networking environments that can execute on commodity laptops. These environments provide realistic implementations of networking protocols and remain accessible to students at all times without confining them to dedicated laboratory facilities. This

paper presents a container-based experiential learning framework designed for an undergraduate computer networking course. The framework is based on a four-stage learning cycle: a) Prediction, b) Implementation, c) Observation, and d) Explanation and is aligned with these experiential learning principles by integrating conceptual reasoning with hands-on experimentation and reflection.

Students first predict protocol behavior using theoretical knowledge and understanding. They then implement the corresponding network scenario, observe actual protocol behavior using standard networking tools, and finally explain any differences between predicted and observed results. The primary contribution of this work is not a new laboratory platform, but rather a pedagogical methodology that systematically integrates theoretical reasoning with experimental validation covering a large number of networking topics discussed in textbooks and class lectures etc.

Currently, Generative AI with its widespread availability and use is reshaping computing education. While AI tools can provide explanations, generate code, and assist with troubleshooting, they also create the possibility that students complete assignments without fully grasping the underlying concepts. In networking education, meaningful learning requires students to reason about protocol behavior, construct networks, observe actual outcomes, and reconcile differences between expected and observed results. The proposed framework is designed to support this process by placing experimentation and observation at the center of learning. Within this framework, AI can serve as a learning companion that assists in diagnosis and explanation, while the core learning experience remains grounded in hands-on interaction with real network systems.

The contributions of this paper are:

- A Prediction, Implementation, Observation and Explanation (PIOE) framework for experiential learning of networking education.
- A versatile and reusable Docker-based infrastructure that supports networking experiments across all layers of the protocol stack.
- A classroom deployment demonstrating the application of the framework across a number of networking topics, along with preliminary student feedback.

2 Related Work

There exist numerous platforms and methodologies that are being used to teach computer networking concepts through interactive exercises and hands-on experimentation. Over the years, the author has experimented with some of these in teaching computer network courses, such as, Cisco Packet Tracer [3], GNS3 [7], Mininet [12], NS2 simulator, VM-based environments and Wireshark [17] (regularly being used even today) etc. Network simulators are software implementations that attempt to replicate the behavior of real networking protocols. However, being a simulator, students are constrained to the functionality and protocol behavior implemented by the simulator rather than interacting with the standard networking utilities and protocol implementations available in real operating systems. Like any software, simulators can occasionally contain implementation errors. The author has encountered such discrepancies from as early as 2012 through 2026 and has observed that they can confuse students and hinder students' understanding of networking concepts..

2.1 Cisco Packet Tracer

Cisco Packet Tracer [3], the first network visualization tool used by this author, is widely used to teach network design, routing, switching, and network configuration. Its graphical interface allows students to construct network topologies and visualize packet movement through simulated devices. However, it being a simulator, limits exposure to actual networking stacks in Operating Systems and real protocol implementations.

2.2 GNS3

Graphical Network Simulator-3 (GNS3)[7] enables emulation of network using virtualized router and switch images. It has been successfully used to teach network routing and switching concepts. However, it typically focuses on network device configuration and often requires a more involved configuration setup as experienced by the author.

2.3 Mininet

Mininet [12] is a popular platform for networking education and extensively used in the study of Software Defined Networking (SDN) [12]. It uses programming artifacts to create network topologies consisting of hosts, switches, controllers, and links. Mininet has been widely adopted in networking courses (the author adapted it a decade ago) as it provides realistic protocol behavior while having low resource requirements. While Mininet excels in SDN and network experimentation in programmable networks, our framework differs in its educational focus. We utilize containerized environments to support experiential learning activities spanning all layers of the networking stack.

2.4 Wireshark-Based Laboratories

Packet capturing and analysis using Wireshark [17] has become a de-facto component of many networking curricula. The author introduces it as the first network tool when teaching a computer network course. It is very helpful for students in exploring the full protocol stack and dissecting packets at the bits and bytes level. The packet analysis provides valuable

visibility into network traffic and is very helpful in network diagnosis. In our networking exercises, students are required to use it to analyze protocol behavior when generating real traffic.

2.5 Virtual Machine Based Laboratories

Virtual machine (VM) based networking laboratories provide students with real networking environments and enable experimentation using actual operating systems and networking tools. The author has extensively used Linux VMs for experiential learning based teaching[1]. However, for students with regular use laptops, a VM-based approach does not scale well when exercises involve 5+ VMs as it involves substantial computational resources, large disk images, complex setup and configuration procedures. Container-based environments offer a lightweight alternative that supports rapid deployment, reproducibility and efficient resource utilization while maintaining access to real protocol implementations. On a typical laptop, docker based containers easily scales up to 10+ containers without any noticeable performance degradation and thus provide a technological convenience as well as substantial improvement in practice over using VMs.

2.6 Positioning of Our Work

Our work builds upon these previous approaches by combining Linux networking tools with a structured experiential learning methodology. Students interact directly with protocol implementations at all layers of the TCP/IP stack, i.e., application layer - HTTP and DNS; network layer - routing tables construction and lookup; transport layer - end to end delivery, traffic control mechanisms and Layer-2 forwarding behavior etc. Furthermore, the proposed framework explicitly links theoretical analysis with practical experimentation. This approach encourages students to reconcile expected and observed protocol behavior. This focus on bridging analytical reasoning and implementation distinguishes our approach from existing networking laboratory environments.

3 Experiential Learning Framework

Most networking exercises use a typical network setup (or its variant) as shown in Figure 1 which consists of two hosts (A and B) and two edge routers R_1 and R_2 connected by a network representing internet (using router(s) R_N or direct connection). Each host and router is instantiated as a separate docker container having required functionality under Docker Desktop [5]. The links represent ethernet connectivity. The

Figure 1: A typical network setup used in experiential learning framework

full network environment is created using docker compose YAML file. For Layer 2 Network setup, the router instance is replaced by an L2 switch instance. The GitHub repository [16] provides all the required docker compose YAML files to create network instances and associated Dockerfile(s) to build docker images for each of the experimental exercise. This proposed framework follows the PIOE cycle as shown in Table 1.

3.1 Prediction

Students are given a networking scenario based on theoretical concepts introduced during class lectures. For example, on the topic of computing network delays, students first calculate transmission and propagation delays, compute (predict) queue build-up and hence queuing delay etc. For the topic of network routing, students design routing table at each router. When discussing end to end delivery at transport layer, students workout messages to be transmitted and received, both by TCP stack and associated application, estimate packet loss and retransmission (for TCP), and similarly reason about Layer-2 forwarding behavior based on entries in the MAC table of switches

3.2 Implementation

Students start by constructing the corresponding network environment using Docker-based containers. Each student is assigned a different network number range which enables them to create their own customized network[2]. Unlike simulation environments, the framework leverages the native Linux network stack and protocol implementations. Students then perform the assigned exercises on the resulting network.

3.3 Observation

Students capture protocol behavior using standard networking tools and software. The docker images are pre-installed with the required Linux networking utilities and software tools including the Apache Web server, netcat (nc), ping/-traceroute, Python socket library, traffic shaping (tc), iproute2, bridge, iptables, tcpdump, netstat, iftop, bridge and python. The observation phase encourages students to identify the

[1]The author, in the previous decade, first started with a setup consisting of physical Linux systems (representing hosts and routers) and commodity Layer-2 switches. This setup, however, faced challenges w.r.t. scaling and resource availability.

[2]This somewhat hinders a simple copying of docker compose files from other students

Table 1: Kolb EL theory and PIOE framework

Theory (Kolb)	PIOE
Abstract Conceptualization	Prediction
Active Experimentation	Implementation
Concrete Experience	Observation
Reflective Observation	Explanation

relationship between protocol state and observed packet exchanges.

3.4 Explanation

After conducting the experiment, students analyze the observed behavior and summarize their learning by explaining any differences between the observed results and their theoretical predictions. When observed network behavior is different from expected lines, students debug and diagnose the network setup and configuration and repeat Implementation and Observation steps. This cycle plays a crucial role in developing a deeper understanding of the underlying network topic while honing students' diagnostic skills.

4 Representative Learning Activities

Below we present some of the representative experiments commonly used in our teaching of computer network course.

4.1 End-to-End Delay and Queueing Analysis

Students use a multi-router network having links with heterogeneous bandwidths (transmission delays), propagation delays, and (student specific) customized queue capacities at each of the router for a series of packets transmitted back to back (or at some interval) from host A to host B (Figure 1). They compute transmission and arrival time at each router accounting for transmission, propagation, and queueing delays for multiple packets as well as identify specific packet(s) that would be dropped due to finite buffering (different queuing capacity for each student) at routers. The processing or computing delay is ignored (treated as zero) as it plays an insignificant role. Subsequently, the given scenario is implemented using Linux traffic control (*tc*) mechanisms which enables them to configure queue capacity, transmission and propagation delays. Lastly, students validate their calculations through packet captures with the predicted timestamp/delay values and analyze discrepancies observed in the implementation. This activity helps students understand the packet scheduling and delay interactions.

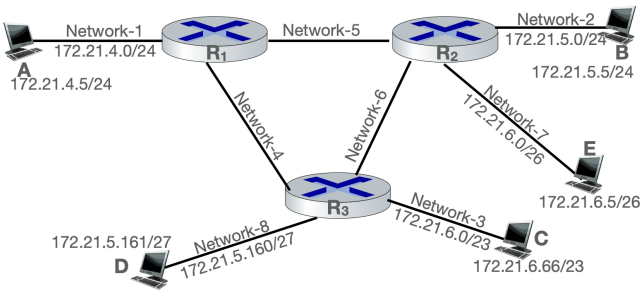

Figure 2: Longest Prefix Match based Routing

4.2 TCP Flow Control and Fairness

TCP flow control often comes up as a difficult topic for students to visualize because in a normal application usage, the role of receiver window advertisements and receive buffer occupancy and its interaction with receiver application remain hidden. In this experiment, using the network setup of Figure 1, students configure a small receive buffer size (e.g., 4096 bytes using *setsockopt()* socket library API [15]) for the receiver application and a corresponding send buffer size at the sender application. The sender application generates traffic that exceeds receiver processing capability. The receiver program is configured to read data at a slow rate leading to gradually filling up the receive buffer completely. This also results in sender application getting blocked as its send buffer becomes full. Packet captures (*tcpdump*) at receiver shows decreasing receive window size followed by zero-window advertisements and window reopening events. Similarly, Linux *netstat* command shows the current status of receive and send buffers occupancy at receiver and sender applications.

A related activity introduces multiple TCP senders competing for bandwidth on a shared bottleneck link connecting R_1 and R_2. Using the topology of Figure 1, multiple hosts are added to routers R_1 and R_2. Students observe convergence toward equal bandwidth allocation by TCP congestion control algorithm AIMD (Additive Increase and Multiplicative Decrease) [1] and TCP fairness principle.

4.3 IP Routing and Longest Prefix Match

Longest Prefix Match (LPM) based packet forwarding plays a key role in Internet routing. Students often find it difficult to grasp. The network setup shown in Figure 2 represents a simple network that involves LPM based forwarding. In this exercise, students manually construct forwarding tables in a multi-router network with overlapping IP subnets. For example, network-7 is fully contained in network-3 and similarly network-8 is fully contained in network-2. The exercise requires students to reason about routing decisions using

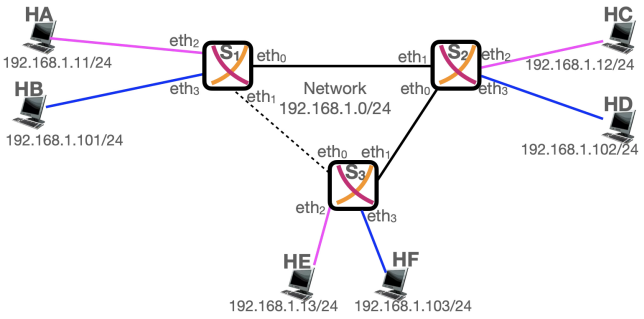

Figure 3: Layer 2 network with provision for loop

longest-prefix match rather than using simplistic destination matching. For example, when $R1$ receives a packet with destination address of host E, its routing table will match two entries corresponding to Network-3 and Network-7, but it should forward the packet to router R_2 instead of R_3. As an additional exercise, students create routing entries for unreachable network destinations as well as routing loop among routers $R1$, $R2$ and $R3$. This helps them study ICMP [14] protocol behavior and observe error messages corresponding to Time to Live (TTL) expiration and Destination unreachable cases. These activities provide direct exposure to router forwarding behavior that is typically abstracted away in textbook examples.

4.4 IPv6 and Transition Mechanisms

IPv6 [4] network is increasingly being deployed and its adoption by google users has currently exceeded 47% [8]. Students study native IPv6 networks, dual-stack networks, and IPv6-over-IPv4 tunnels by making appropriate network setup and configuration changes in Figure 1. For the IPv6-over-IPv4 tunnel part, both hosts are configured as pure IPv6 network whereas internet part ($R_1 - R_N - R_2$) is configured purely as an IPv4 network with an IPv6-over-IPv4 tunnel configured between $R1$ and $R2$. Through packet captures and routing analysis, students observe and become familiar with encapsulation mechanisms used during IPv6 transition and develop practical familiarity with IPv6 addressing and routing. They access the webserver running on host B from host A using both IPv4 and IPv6 addresses and experience working with dual IP stacks.

4.5 Layer-2 Switching and Broadcast Storms

In network setup shown in Figure 3, students construct Layer-2 topologies replacing routers by L2 switches using Linux bridges and observe MAC address learning and unknown-unicast forwarding. By introducing Layer-2 loops (additionally connecting S1 with S3 directly, shown as dashed line), students learn persistent flooding. By creating additional links between these switches, students observe broadcast storm bringing the network to halt in a very short time due to exponential traffic generation. These activities help students understand L2 switch operation beyond the simplified forwarding models typically presented in class lectures.

4.6 Artifacts Repository and Reusable Infrastructure

The proposed framework is built around a set of reusable Docker-based networking templates. The complete collection of templates and instructional materials is publicly available through the GitHub repository [16], enabling instructors to adopt, extend, and customize the exercises as per their requirements.

5 Classroom Deployment, Preliminary Findings and use of Generative AI

The proposed framework has been deployed in an undergraduate Computer Networks course with an enrollment of approximately 40 to 80 students per semester. Students complete an average of six major networking assignments and one optional extra-credit assignment during the semester. These experiential assignments collectively account for more than 50% of the course grade, emphasizing the importance of hands-on learning.

Based on students' submission reports, the average time spent on each assignment ranged from four to six hours. This duration suggests that students were required to engage substantially with the networking concepts, perform troubleshooting, and analyze protocol behavior rather than merely execute predefined instructions. As part of semester end course feedback, students were asked to identify the most valuable aspect of the course. A recurring theme across student responses was appreciation for the opportunity to directly implement and observe networking concepts in realistic environments. Representative comments included:

> *"Could see how the topics we learned in class could be utilized in a practical, day-to-day sense."*

> *"Got to implement what we were learning with my hands."*

Students also highlighted the connection between course content and professional practice, and several students explicitly identified the assignments as critical to their understanding of networking concepts. These responses suggest that students valued the ability to connect theoretical concepts with practical implementation and experimentation. Representative comments included:

"The homeworks were fun but required additional knowledge that was not taught in lecture"

"Explained networking concepts, especially those connected to real jobs and industry skills"

"Assignments were very well described and very useful for understanding concepts taught in class"

5.1 Generative AI as a Learning Aid

Students frequently use generative AI tools to obtain explanations of networking concepts, troubleshoot configuration issues, and resolve implementation-related runtime errors. AI-assisted learning has emerged as a recurring theme during classroom deployment and is discussed further in the next Section.

6 Discussion, Lessons Learned and Future Work

Although a formal educational study has not yet been conducted, several observations emerged consistently during classroom deployment as given below.

First, students appeared to develop a direct connection between theoretical analysis, protocol implementation and observable protocol behavior. Rather than treating protocol fields and packet exchanges as theoretical constructs, students directly observed queue build-up, packet loss, intricacies of HTTP, end to end delivery with TCP/UDP protocol, IP routing table construction and lookup including IPv6, self-learning by Layer2 switches and Virtual LANs (VLANs).

Second, the framework uses standard networking tools commonly employed by network administrators. The use of containerized environments reduced dependency on specialized laboratory infrastructure while allowing students to experiment with realistic network configurations on their own systems. This portability enabled students to conduct experiments without the need of scheduled laboratory sessions and encouraged self-directed and self-paced exploration.

Third, student feedback suggests that experiential activities improved engagement by witnessing visibility of protocol operation. Many comments specifically referenced the Docker-based infrastructure and the opportunity to implement networking concepts.

An interesting observation was that some students reported needing additional support beyond lecture material to complete assignments. While this increased the challenge of the coursework, it also encouraged independent exploration, troubleshooting, and engagement with networking documentation and Linux networking tools. Such activities closely resemble the problem-solving processes used by networking professionals in practice.

Finally, the exercises spans the entire TCP/IP stack, enabling students to develop a holistic understanding of protocol interactions. These exercises have been used into an undergraduate computer networking course across multiple networking classes spanning last 5 semesters. Informal observations indicate that students demonstrate stronger engagement when protocol behavior is directly observable. Students frequently report that the experiential learning approach helps them understand concepts that previously appeared abstract. Furthermore, explaining the differences between predicted and observed behavior encourages critical thinking and promotes deeper conceptual reasoning.

However, several challenges still remain to be addressed. Students initially require guidance when using Linux networking tools, diagnosing network configuration issues and interpreting packet captures. Additionally, they need some mentoring when measurements obtained from real systems exhibit some variability that differs from idealized textbook models. However, these discrepancies themselves often become valuable learning opportunities. Further, it is possible that students may ask AI to write the "prediction" part (which may be incorrect at times) and bypass this cycle. However, when implementation mismatches with the prediction and student need to debug and fix the issue, the real learning process begins.

6.1 Generative AI and Experiential Learning

An important observation during deployment was the increasing use of generative AI tools by students. While AI easily accelerates troubleshooting and provide explanations of networking concepts, it also creates the possibility that students rely on generated solutions without fully engaging with the underlying concepts. The PIOE framework mitigates this risk by requiring students to formulate predictions, execute experiments, observe actual network behavior, and account for discrepancies between expected and observed outcomes. The approach is expected to prevent students from using AI to fake their way through the labs. However, AI appeared to be most beneficial during the Implementation and Explanation phases, where students used it to resolve setup challenges, interpret protocol interactions, and better understand experimental results. These observations suggest that experiential learning frameworks can provide a structure within which AI can serve as a learning aid rather than a replacement for hands-on engagement.

6.2 Future Work

Future work will focus on conducting a more rigorous educational assessment using structured surveys to quantify learning gains associated with the PIOE framework. Additional work is needed to evaluate the effectiveness of the framework across other topics related to networking (e.g., Network Security), class sizes, and instructional settings. Given the increasing use of generative AI in computing education, an important direction for future research is a systematic study of how AI influences student learning, problem-solving strategies, and conceptual understanding in networking courses. In particular, we plan to explore how AI can be integrated into the PIOE cycle to support prediction, troubleshooting, reflection, and explanation while preserving the hands-on and evidence-based nature of experiential learning.

7 Conclusion

This paper presented a container-based experiential learning framework for computer networking education based on a PIOE cycle. By combining analytical reasoning with reproducible network experimentation, the framework enables students to bridge the gap between networking theory and protocol implementation. Representative activities spanning delay analysis, TCP behavior, routing, IPv6 transition, and Layer-2 switching demonstrate the applicability of the approach across multiple networking topics. Future work will focus on systematic evaluation of learning outcomes, integration of generative AI within experiential learning environments, and extension of the framework to advanced networking and security topics.

Acknowledgments

The author thanks the Department of Computer Science and Electrical Engineering (CSEE), College of Engineering and Information Technology (CoEIT) at the University of Maryland, Baltimore County (UMBC) for providing the teaching environment and institutional support that made this work possible. The author also appreciates the valuable feedback provided by students who participated in the course offerings.

References

[1] M. Allman, V. Paxson, and E. Blanton. 2009. RFC 5681: TCP Congestion Control.

[2] Olivier Bonaventure and Quentin De Coninck. 2024. Computer Networking : Principles, Protocols and Practice. available at https://www.computer-networking.info, Accessed: July 2026.

[3] Cisco Networking Academy. 2026. Cisco Packet Tracer. https://www.netacad.com/cisco-packet-tracer. Accessed: June 2026.

[4] S. Deering and R. Hinden. 2017. RFC 8200: Internet Protocol, Version 6 (IPv6) Specification.

[5] Docker Inc. 2026. Docker Desktop. https://www.docker.com/products/docker-desktop/. Accessed: June 2026.

[6] Mohammad J. Eddy. 2022. RFC 9293: Transmission Control Protocol (TCP).

[7] GNS3 Project. 2026. GNS3: Your Virtual Network in a Suitcase. https://www.gns3.com/. Accessed: June 2026.

[8] Google. 2026. IPv6 Adoption Statistics. https://www.google.com/intl/en/ipv6/statistics.html. Accessed: June 2026.

[9] Thomas Holterbach, Tobias Bühler, Tino Rellstab, and Laurent Vanbever. 2020. An Open Platform to Teach How the Internet Practically Works. *SIGCOMM Comput. Commun. Rev.* 50, 2 (2020), 45–52. https://doi.org/10.1145/3402413.3402420

[10] David Kolb. 2014. *Experiential Learning: Experience as the Source of Learning and Development* (4 ed.). Pearson Education, Upper Saddle River, NJ.

[11] James F. Kurose and Keith W. Ross. 2025. Computer Networking: A Top-Down Approach, 9th ed. https://gaia.cs.umass.edu/kurose_ross/index.php. Accessed: July 2026.

[12] Mininet Project. 2026. Mininet: An Instant Virtual Network on Your Laptop. https://mininet.org/. Accessed: June 2026.

[13] Larry L. Peterson and Bruce S. Davie. 2020. *Computer Networks: A Systems Approach* (6 ed.). Systems Approach om Web, Burlington, MA. Available at https://book.systemsapproach.org/, Accessed: July 2026.

[14] J. Postel. 1981. RFC 792: Internet Control Message Protocol.

[15] Python Software Foundation. 2026. socket — Low-Level Networking Interface. https://docs.python.org/3/library/socket.html. Accessed: June 2026.

[16] Ram Rustagi. 2026. Docker-Based Exercises for Computer Networks and Network Security. https://github.com/rprustagi/Docker-based-exercises-Network-Security.git. GitHub repository. Accessed: July 2026.

[17] Wireshark Foundation. 2026. Wireshark: The World's Leading Network Protocol Analyzer. https://www.wireshark.org/. Accessed: June 2026.