WNSWE: Web-based Network Simulator for Web Engineering Education

Sebastian Heil[1 [0000−0003−2761−9009] (✉)], Lucas Schröder[1 [0009−0009−5540−4495]], and Martin Gaedke[1 [0000−0002−6729−2912]]

Technische Universität Chemnitz, 09111 Chemnitz, Germany
sebastian.heil@informatik.tu-chemnitz.de
lucas.schroeder@informatik.tu-chemnitz.de
martin.gaedke@informatik.tu-chemnitz.de

Abstract. The complexity of current Web Information Systems represents a challenge when educating the next generation of engineers capable of dealing with all aspects of their life cycle. Network simulators are used in university education to address this challenge and provide students with the necessary comprehensive understanding of the interplay of their inherently distributed system components and the respective protocols at different network layers. However, existing simulators only partially satisfy educational requirements, having limitations in the level of control of educators to focus on selected aspects on the desired conceptual level, support for demonstrations with interventions at runtime, and the flexibility to set up, use, and program the simulators at home and on mobile devices. Thus, we propose WNSWE, a Web-based Network Simulator designed for Web Engineering Education, detailing its client-side architecture, program model, and interactive capabilities. To evaluate WNSWE, we applied it to 26 educational scenarios and report on 3 case studies of employing it for classroom teaching and asynchronous assignments in different university courses and teaching settings, covering a range of topics from basic networking to cloud and security of distributed systems. Furthermore, WNSWE is provided as open educational resource (OER) and we outline our roadmap for its extension.

Keywords: Network Simulator · Web Engineering · OER · Simulation-Based Learning · e-Learning · SDG4 · CS2023 · WebAssembly

1 Introduction

Engineering Web Information Systems requires profound knowledge of the entire underlying technological platform. As Web-based systems are inherently distributed, the complexity of their design, implementation, testing, optimizing their performance, hardening their security, operations and resolving errors puts high demands on Web Engineers. The integration of emerging technologies and paradigms such as microservices, blockchain and AI-as-a-Service components have further increased systems' complexity. At the same time, they

represent a large share of commercial software production and companies worldwide are seeking for experts.Universities address this demand by dedicated study programs.Transforming students into highly qualified Web Engineers exceeds a mere training of Web development using the latest frameworks. Thus, curricula comprise of various modules to teach the necessary basics for gaining a deep understanding of the interplay of distributed system components and protocols at different layers of the network as well as within the application layer itself.

Studying these subjects can be challenging for students due to the complexity of the protocols and their interactions. The communication models comprise of various message exchanges not easy to observe directly without dedicated instrumentation and hardware (e.g. network sniffers, programmable routers), especially while studying outside of university premises at home. Network simulators [17,15,4] are employed in academic education to address these issues and provide hands-on experience to internalize the required concepts [8]. However, most focus on providing a detailed analysis tool for the TCP/IP stack in large network architectures [17,15,8], only partially satisfying the didactic requirements for Web Engineering education. In particular, they have a lack of coverage of custom application layer protocols (such as MQTT), limited flexibility to restrict the simulations only to the relevant communication aspects of interest for a specific teaching session, missing support for allowing students to implement the behavior of components on their own, a lack of interactive control to intervene during the simulation for demonstrations of behaviors not predefined by the scenarios during teaching sessions (e.g. man-in-the-middle attacks), and a technical setup requiring installation on a PC with limited compatibility to different operating systems and mobile devices.

Students and educators benefit from a network simulator specifically created to address the requirements of Web Engineering education in universities. It can be used for both demonstrations in synchronous teaching sessions in classrooms or online as well as for asynchronous study activities such as assignments, self-study units or exam preparations. The importance of interactive e-learning in university education allowing students to internalize contents by actively creating and interacting with learning materials has increased. Approaches such as Challenge-based learning have a positive effect on motivation, knowledge retention and study success [11]. Simulation-based Learning (SBL) [3] is a popular approach in computer science didactics.

Thus, the objective of this work is to create a Web-based network simulator for educational use in universities which fulfills the following requirements:

- programmable behaviors of all system components to allow educators to restrict the focus on specific communication aspects and represent arbitrary protocols across all network layers, including non-standardized custom application layer protocols
- inspection and modification of all parts of the system and the messages interactively at runtime to enable demonstrations and experimentation with error scenarios, including removal of messages, alteration of message contents, and changes of connections

- platform-independence and support for mobile devices with sufficient screen size such as tablets through a web-based solution not requiring installation and usable in web browsers without permanent internet connectivity
- availability of the simulator for educators and students as open educational resource (OER) [16]

In section 2, we present the architecture and implementation details of our simulator. We position our solution against related work in section 3. The results of an extensive empirical evaluation of the simulator through the usage in three different modules are presented in chapter 4. Chapter 5 concludes the paper and provides an outlook on future work.

2 WebAssembly-based Client-Side Network Simulator for Web Engineering Education

2.1 Network Simulator Architecture

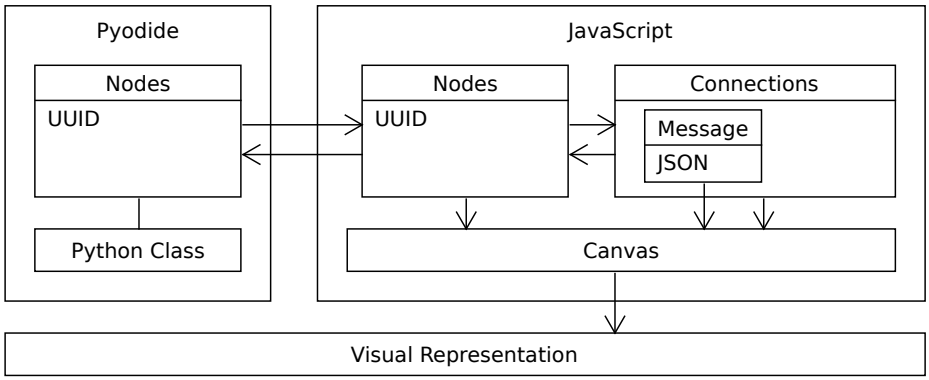

Fig. 1. WNSWE architecture. Each node has a JavaScript object and twin Python object, passing messages. Nodes, connections and messages in transit are visualized using an HTML canvas.

WNSWE is a client-side web application[1], which, after initial loading, does not require further network access for its core functionality. The core concepts of the network simulator are nodes, weighted connections and messages, forming a network graph. The behavior of the nodes can be configured to send custom messages via these connections to their neighboring nodes. The weight of a connection determines how long the messages takes to travel across the connection. The receiving node can then handle the message according to its own behavior. This enables the replication of distributed algorithms and protocols in an

[1] cf. implementation repository at https://github.com/LucasSchroeder99/WSNWE

abstracted fashion. Each node runs Python code, through which it can send messages with custom JSON content along a connection to a neighboring node. The code can be edited by the user, allowing learners to replicate, change or expand algorithms. Within the simulation, each node runs its own, asynchronous Python code via Pyodide. Most of Python's standard library is available and, if needed, more complex node behavior may be achieved via concurrently running, asynchronous coroutines. The overall architecture is shown in figure 1 The nodes and connections are defined by educators or learners via a canvas-based visual representation, which also shows the messages in transit. This way, the distributed events are visualized and the global state of the distributed system can be grasped more easily. This is facilitated by the capability of nodes and messages to be colored according to their state or type.

2.2 Programming Model for component behavior

WNSWE enables programming the behavior of nodes in the network graph. This is achieved by allowing the user to edit the Python code run by each node of the network graph. This not only allows to recreate almost arbitrary algorithms and protocols, but also enables the use of WNSWE within tasks or assignments in different didactic contexts. For example, an algorithm may in advance be implemented by the instructor, with deliberate gaps, which then have to be filled by students. To allow nodes to both specify a unique behavior for the scenario while also making use of code shared with other nodes, each node is an instance of a specific Python class, with an added unique `run()`-method. It is automatically executed for all nodes at the start of the simulation. This special method is a function added to the instance of the node object, but not the class, allowing multiple nodes of the same class to have differing behavior while reusing the same class code. This allows the class to contain shared code, for example code to create or parse abstract Ethernet frames, while still specifying a specific behavior required for the scenario in the `run()`-method, such as having a specific node initiating the communication. Fundamentally, each Python node object makes use of pre-defined, asynchronous `send()`- and `receive()`-methods, which facilitate the interaction with their respective JavaScript twin object.These methods are part of a special `Endpoint`-class, from which all other node classes inherit. This allows to hide the actual code required to interface with the JavaScript side of the simulator. By using asynchronous methods, blocking or parallel method behavior can be simulated. Beyond the two communication methods, further utility functionality exists, for example to change the color of a node or to query the current time within the simulation. If required, globally shared data structures can also be employed, which, although not realistic, may aid educators in achieving didactic reduction. To specify specific pre-configured scenarios for learners, the network graph and the created Python code can be exported as a scenario JSON file for later use.

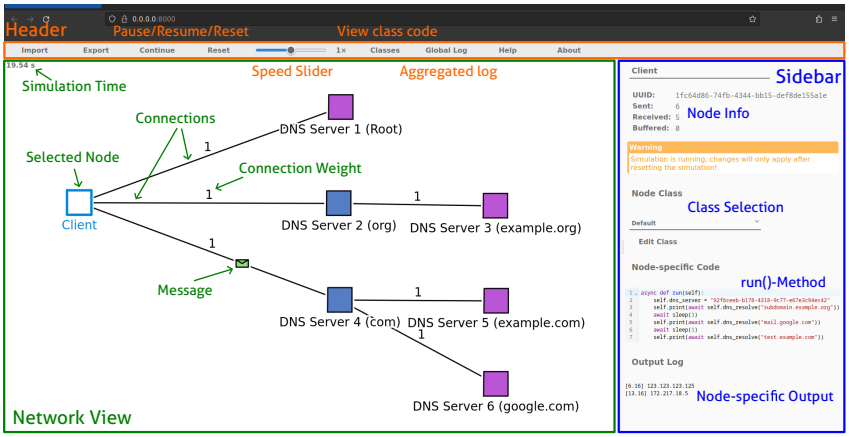

Fig. 2. A screenshot of the WNSWE GUI, showing a scenario about iterative and recursive DNS resolution. The main areas, specifically the header, sidebar and network view, are highlighted and important elements labeled.

2.3 Client-side GUI and Runtime functionalities

The network simulator has a graphical user interface that is divided into three parts, as shown in the screenshot in figure 2. These are the header – containing the essential simulation controls – highlighted in orange, the sidebar – containing information specific to the currently selected element of the network graph – in blue, and as the main part the visual representation of the network graph and its global state, in green. The *header* at the top of the GUI, contains several buttons that allow controlling the simulation and provide information. They can be used to export or import scenario files and to start, pause, resume or reset the simulation. Furthermore, a speed slider is available which allows changing the speed of messages and timeouts within the simulation from 0.1x to 10x. Additionally, the Python code for node classes can be accessed and edited via the "Classes" button. A help section is also accessible, which also includes a simple documentation for the pre-defined Python methods, such as the aforementioned `send()`- and `receive()`-methods and other utility functionalities. On the right side, there is a *sidebar*, the contents of which depend on the element currently selected in the Network View. It shows basic information about the corresponding element and allows to make some changes to it. For instance, it displays the content of messages, the `run()`-methods of nodes or the weight of connections. The main part of the simulator, the *Network View*, is a canvas containing the visual representation of the network graph with all nodes, connections and messages at the current point in time of the simulation. While the simulation is running, changes are visualized accordingly, and messages are visible. By using a mouse or touch, the viewport and zoom level can be controlled and via a context menu changes can be made, such as adding, editing or deleting nodes or connections.

Similarly, messages in transit can be inspected, their contents altered or the entire message deleted. When selected, the JSON contents of the message are displayed in an editor within the sidebar. Any changes to the contents update the message accordingly. Lastly, connections can be added, changed or deleted, even at runtime. This enables limited changes to the network graph during the simulation, without changing the already running Python code, e.g. simulating a physical loss of connectivity.

3 Related Work

Network Simulators have been used as tools by network engineers and researchers for a long time. They allow cost-efficient experimentation with various hardware and software configurations in order to evaluate network architecture and protocols. This has made them relevant for educational purposes [14], following the idea of Simulation-based learning [3]. We follow the taxonomy by Gomez et al. [8] and focus on Network Simulators, as opposed to Network Emulators and Network Testbeds, since these are the most accessible for educational use.

Network Simulators such as OMNeT++ [17], ns-3 [15] or OPNET [2] provide comprehensive network planning and optimization functionality, allowing to analyze various behavioral and performance aspects of large-scale network topologies by simulating real hardware. Depending on their simulation model, they are distinguished into time-based and event-based simulators [8]. Established non-educational Network Simulators are used in university education [14]. However, their focus is on traffic realism, scalability and a high level of technical detail down to the physical layer, resulting in a relatively steep learning curve. Their programming models primarily require the use of system programming languages like C++ [17,15,2] and they exhibit a relatively high interaction complexity [8]. Due to these characteristics, their educational utility in Web Engineering programs is only limited.

Network Simulators for Education are specifically built for use in academic teaching contexts. They differ from general Network Simulators by emphasizing usability, user-friendly graphical user interfaces, and customization for specific educational scenarios, allowing learners to focus more on the content matter and less on learning how to use the tools [8]. For instance, Cisco Packet Tracker [4] provides a visual network topology editor and supports simulation of the most common network protocols, and additionally supports collaborative features enabling group learning. Several case studies [1,13] are reporting on the use of Cisco Packet Tracker in Computer Network courses for university students, highlighting benefits like facilitating the understanding of network concepts and allowing students to experiment at home without need for the hardware and fear of damaging equipment. The Graphical Network Simulator (GNS-3) [6] is an open-source network simulation platform especially used in certification and cybersecurity training contexts, but also in university programs [18,14]. It particularly emphasizes the relevance of visualizing simulated network components and connections in education. It provides a detailed coverage of network protocols

across all layers, supported by an active marketplace which allows contributors to share custom implementations, and can be extended using the HTTP API or by emulators like Qemu[2] or Virtualbox[3]. Due to its high level of abstraction and wide adoption, python is commonly used for scripting with GNS-3.

While network simulators for education better address the requirements of learners in terms of ease-of-use and extensibility, their main focus is on detailed networking education. Suitability for university programs training the next generation of Web Information Systems Engineers is only limited, as the required competencies do not lie in the area of network administration, but rather a profound conceptual understanding of the layers and protocols underneath the Web ecosystem. Furthermore, they require installation and cannot be directly used by students on various platforms including mobile devices. In contrast, WNSWE better supports didactic reduction [5], allowing to limit the learners focus on specific aspects and a conceptual understanding, rather than the complete and detailed scope required for training network engineers. An exhaustive and more detailed overview of Network Simulators in Education can be found in the surveys by Gomez et al. [8] and Prvan et al. [14].

Web-based Network Simulators. A few network simulators are implemented as web applications, making use of the established and uniform platform of the web in order to achieve wide compatibility and an installation-free setup. This lowers the initial hurdle and allows learners to use the simulators at home or on mobile devices with sufficient screen size such as tablets. One example is the Educational Network Simulator [7]. Targeted at secondary schools, it comprises of ready-to-use implementations for the demonstration of basic networking concepts and interaction with protocols like DNS, DHCP and HTTP, but does not support customization through programmable behaviors. Like WNSWE, it is implemented as client-side web application, in this case based on JavaScript. PT Anywhere [12] is another web-based Simulator created in the context of EU FP7 project FORGE promoting experiment-driven education[4]. Developed in cooperation with Cisco Networking Academy, it is based on the Cisco Packet Tracker [4], but provides a Web frontend. The project emphasizes the importance of allowing learners to access simulations at home in mobile environments. While WNSWE shares the low installation effort and compatibility with mobile devices of existing Web-based network simulators, it supports a much wider scope and is extensible compared to the Educational Network Simulator, and does not require a server and thus internet connection like PT Anywhere. On the didactic side, the focus of WNSWE is on concptual-level teaching for Web Engineering education in contrast to the network engineering focus of Educational Network Simulator and PT Anywhere.

[2] https://www.qemu.org/
[3] https://www.virtualbox.org/
[4] cf. Future Internet Research Experimentation (FIRE)

4 Experiments

In this section, we present the results from our evaluation of using WNSWE in university education. The evaluation is two-fold, focusing on feasibility and utility. For the feasibility evaluation, we tested the applicability to a wide range of Web Information Systems topics by applying WNSWE to model 26 scenarios, as described in subsection 4.1. Additionally, we conducted a pilot evaluation of utility through three case studies in real-world educational settings, which we report about in subsection 4.2.

4.1 Feasibility Study

The evaluation objective of the feasibility study is to test the applicability of WNSWE to various education topics relevant for Web Information Systems engineers. We used the joint ACM/IEEE-CS/AAAI Computer Science Curriculum (CS2023) [10] as basis for the selection of the scenarios. Overall, 26 scenarios have been modeled in WNSWE. All scenarios are available as OERs in the replication package [9] of this paper.

4.2 Case Studies

WNSWE was evaluated in 3 different case studies in different university courses. To test the utility in different real-world educational settings, the case studies were selected to represent a diverse range of characteristics in education level, teaching mode, didactic method, and topic. They cover education of undergraduate as well as master students, asynchronous assignments as well as synchronous classroom teaching sessions, and instructive demonstrations as well as HAITI-based tasks.Each case study corresponds to a different module taught at Chemnitz University of Technology. Feedback from the students and educators in the case studies was used to already make limited improvements on WNSWE.

C1 Networks. Case study C1 took place in the context of a computer networks module focusing networking basics on all 7 ISO-OSI layers and is taught by two educators of 5 and 12 years teaching experience. The module is open to bachelor students of computer science with a distributed systems specialization, as well as to other computer science bachelors and biomedical engineering students. WNSWE was used in 2 asynchronous assignments and 9 classroom sessions. The asynchronous assignments were tasks in the context of the HAITI method, for which students had 2 weeks time to prepare. These tasks comprised of TCP Acknowledgements and the Email protocols SMTP and POP3. In synchronous teaching, WNSWE was used for 12 demonstrations as well as for 1 in-class group live task (IP Router behavior). The demonstrations were on various topics across different ISO/OSI layers, starting from layer two upwards. The behavior of a network switch and a translating bridge were shown for layer 2, while Segmentation, ICMP and traceroute behavior and RIP were shown for layer 3. For layer 4, TCP state machine and acknowledgements were shown, for the application layer, recursive and iterative DNS resolution and email protocols.

C2 Cloud. Case study C2 was set in the Cloud and Web Applications module, which focuses on the engineering of Web-based systems in the context of cloud computing, taught by an educator of 4 years teaching experience. The module is attended to by Master students of Web Engineering as well as electively of Computer Science. WNSWE was used for an in-class group live task in which students were tasked to implement the behavior of a multi-step processing pipeline. Students were divided into three teams, each team working on one step of the process. Each team worked on one scenario, in which a test message was sent from the incoming queue, had to be processed and sent to the outgoing queue. After each team finished that exercise, a toggle within the scenario activated the communication with the message broker, allowing messages processed by one team to be sent directly to the next team, across multiple devices.

C3 Security. The third case study was conducted in a module on the Security of Distributed Systems, which focuses on security threats in modern Web Information Systems and techniques and protocols to address them. It is taught by two educators of 6 and 4 years of experience respectively. Course attendees are Master students of Web Engineering as well as electively other Computer Science master students. WNSWE was used for an in-class instructive demonstration on HTTPS and TLS and a homework assignment on the Kerberos protocol. In the TLS demonstration, the messages and their respective contents within the establishment of a secure connection were shown. In the homework on Kerberos, an abstract implementation of the Kerberos protocol was given, which already performed authentication and requested a ticket for one service. Students were asked to extend the solution to perform a request to a second service after the first one, making use of the existing ticket granting ticket.

4.3 Discussion

WNSWE has been used in three different Web Engineering related courses at Chemnitz University of Technology, in various didactic settings. Its capabilities to represent distributed algorithms and protocols have been utilized both in demonstrations by educators, as well as in tasks assigned to and solved by students. During this, areas for future improvement have been identified and some changes are already integrated. As an educational tool, the simulator allows instructors to create scenarios for demonstration or for tasks and assignments. However, these scenarios need to be designed carefully to fulfill their intended purpose. It has been noticed that when initially confronted with WNSWE as a new tool, students required some time to get used to its interface, and some tasks had to be simplified to adjust their difficulty towards the intended level. Further utility functionality has been added to the Python code as a result of implementing various algorithms and protocols in the context of the feasibility study, to better handle common tasks. Examples of added functionality include running asynchronous functions concurrently or with a timeout, while retaining proper output and error reporting to the node's output log. While WNSWE is currently in a reasonably usable state, further polishing and additional features are being considered. This includes more runtime capabilities, such as manually spawning

message on connections or pre-determined scenario events, for example to have connections disappear or reappear at specific times during the runtime of the scenario. Furthermore, while WNSWE already runs some checks to detect possible coding errors specific to the network simulation, such as the `run()`-method being renamed, these checks could be extended to cover further common issues and to provide better support when creating scenarios or solving tasks.

5 Conclusion

In this paper we presented WNSWE, a Web-based network simulator designed to support the academic education of Web Engineers capable of dealing with the increasing complexity of Web Information Systems in the AIaaS era. We reasoned about the utility of network simulators in Web Engineering education to empower educators making use of new didactic methods from Challenge/Simulation-based learning, and analyzed shortcomings of existing network simulators. Supported by WebAssembly, WNSWE is a client-side Web application with minimal setup requirements, that supports programmable individual behavior of all components and interactive inspection and modification at runtime, and is available for educators and learners as open educational resource (OER). We applied WNSWE to 26 educational scenarios of the CS2023 curriculum and reported 3 case studies of its application in different real-world university courses. Future work includes the following three main directions. We are extending WNSWE to support the recording and replay of events from user interactions at runtime, allowing to pre-define modifications of the network elements in the scenario descriptions. An improved support for the integration of existing non-simulated components in the simulated architectures, e.g. dedicated configurations for interacting with existing Web, DNS, or Mail servers etc. Furthermore, we plan to extend our experimentation with WNSWE by applying it in other courses and collaborating with educators involved in Web Engineering programs at other universities in order to perform a comprehensive evaluation study.

Acknowledgments. This work is supported by the European Union's HORIZON Research and Innovation Programme under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Allison, J.: Simulation-Based Learning via Cisco Packet Tracer to Enhance the Teaching of Computer Networks. In: Proc. of the 27th ITiCSE. vol. 1, pp. 68–74. ACM, New York, NY, USA (jul 2022). https://doi.org/10.1145/3502718.3524739
2. Chang, X.: Network simulations with OPNET. In: Proc. of WSC '99. vol. 1, pp. 307–314. ACM Press, New York, New York, USA (1999). https://doi.org/10.1145/324138.324232

3. Chernikova, O., Heitzmann, N., Stadler, M., Holzberger, D., Seidel, T., Fischer, F.: Simulation-Based Learning in Higher Education. Review of Educational Research **90**, 499 – 541 (2020). https://doi.org/https://doi.org/10.3102/0034654320933544
4. Cisco Networking Academy: Cisco Packet Tracer - Networking Simulation Tool (2022), https://www.netacad.com/courses/packet-tracer
5. Futschek, G.: Extreme Didactic Reduction in Computational Thinking Education. In: World Conference on Computers in Education. pp. 6–12. Ifip Tc3, IFIP, Torun, Poland (2013)
6. Galaxy Technologies, L.: Getting Started with GNS3 | GNS3 Documentation (2024), https://docs.gns3.com/docs/
7. García Ochoa de Aspuru, J., Quinson, M.: Educational Network Simulator Project (2015), https://malkiah.github.io/NetworkSimulator/
8. Gomez, J., Kfoury, E.F., Crichigno, J., Srivastava, G.: A survey on network simulators, emulators, and testbeds used for research and education. Computer Networks **237**, 110054 (dec 2023). https://doi.org/10.1016/j.comnet.2023.110054, https://linkinghub.elsevier.com/retrieve/pii/S1389128623004991
9. Heil, S., Schröder, L., Gaedke, M.: Replication package for WNSWE: Web-based Network Simulator for Web Engineering Education (1.0.0) (2024). https://doi.org/10.5281/zenodo.13693453, Zenodo
10. Kumar, A.N., et al.: Computer Science Curricula 2023. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/https://doi.org/10.1145/3664191
11. Leijon, M., Gudmundsson, P., Staaf, P., Christersson, C.: Challenge based learning in higher education. Innovations in Education and Teaching International **59**(5), 609–618 (sep 2022). https://doi.org/10.1080/14703297.2021.1892503, https://doi.org/10.1080/14703297.2021.1892503https://www.tandfonline.com/doi/full/10.1080/14703297.2021.1892503
12. Mikroyannidis, A., Gómez-Goiri, A., Smith, A., Domingue, J.: PT Anywhere: a mobile environment for practical learning of network engineering. Interactive Learning Environments **28**(4), 482–496 (2018). https://doi.org/10.1080/10494820.2018.1541911
13. Noor, N.M.M., Yayao, N., Sulaiman, S.: Effectiveness of Using Cisco Packet Tracer as a Learning Tool. International Journal of Information and Education Technology **8**(1), 11–16 (2018). https://doi.org/10.18178/IJIET.2018.8.1.1004
14. Prvan, M., OžEGOVIć, J.: Methods in Teaching Computer Networks;: A Litereature Review. ACM Transactions on Computing Education **20**(3), 1–35 (sep 2020). https://doi.org/10.1145/3394963
15. Riley, G.F., Henderson, T.R.: The ns-3 Network Simulator. In: Modeling and Tools for Network Simulation, pp. 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12331-3_2
16. UNESCO: Recommendation on Open Educational Resources (OER) (2019), https://unesdoc.unesco.org/ark:/48223/pf0000373755/PDF/373755eng.pdf.multi.page=3
17. Varga, A., Hornig, R.: An Overview of the OMNeT++ Simulation Environment. In: Proc. of the 1st International Conference on Simulation Tools and Techniques for Communications Networks and Systems. ICST, Marseille, France (may 2008). https://doi.org/10.4108/ICST.SIMUTOOLS2008.3027
18. Wangchuk, T.: Study on the Usability of GNS3 for Teaching and Learning System and Network Administration. International Journal of Science Technology & Engineering **4**(10) (2018), https://ijste.org/Article.php?manuscript=IJSTEV4I10018