Developing an Interactive Web-Based Programming Platform for Learning Computer Networking Protocols

Dewei Zeng, Zhiyu Zhang, Jiye Chen, and Xiaojun Hei(✉)

Huazhong University of Science and Technology, Wuhan 430074, China
{zengdewei,zhiyuzhang,youtiao,heixj}@hust.edu.cn

Abstract. Computer networking protocols have become important domain knowledge for electrical engineering professionals. The learning-by-doing approach has shown its effectiveness to learn these complex protocols by reproducing research results. In this paper, we design a web-based ns-3 lab platform by integrating various open-source modules for beginners to get hands on network simulations to learn networking protocols with a smoothed learning curve. This platform consists of a vue-based front-end and a docker-based back-end to support elastic on-demand capacity expansion. We implement a simulator scheduling module based on Node.js and restify to achieve load balancing for reducing the simulation waiting time. We conduct a measurement study to evaluate the performance of this prototype system. The measurement results demonstrate the technical feasibility of the prototype design to develop a scalable but user-friendly computer network simulation platform for massive open online lab courses.

Keywords: ns-3 · Online learning · Networking protocols · Engineering education

1 Introduction

In recent years, engineering courses have shown strong tendency toward science courses, lacking of sufficient practical lab platforms. Engineering students are required to accomplish systematically-designed theory and practice modules in order to develop their system capabilities progressively in a pipeline fashion [8–10]. The "computer networking" course is an important professional fundamental course for undergraduate programs on electrical engineering with both theoretical and practical characteristics. It is an effective learning approach to reproduce research results to learn computer networking protocols [1,12]. Nevertheless, it is prerequisite to provision network facilities to support the reproduction of

D. Zeng, Z. Zhang and J. Chen—These authors contributed equally to this work.

H. Song and D. Jiang (Eds.): SIMUtools 2020, LNICST 369, pp. 611–625, 2021.
https://doi.org/10.1007/978-3-030-72792-5_48

research results. With the rapid development of the network, the complexity and customization of the network hardware equipments have been increasing significantly in recent years. Therefore, computer networking educators have been increasingly adopting network simulation labs instead of hardware-based networking labs [3–5,14].

Ns-3 is a popular open-source network simulation tool [7] which has been widely used in the networking research and teaching communities. Many third-party modules have been developed to enhance ns-3, such as ns3-gym [6] and ns3-AI [13] to foster artificial intelligence algorithms in networking research.

When beginners learn to conduct ns-3 network simulations, it is common that they find it difficult and time consuming to get started due to the complicated installation steps and largely scattered learning resources. In this paper, we are motivated to design an online ns-3 learning platform with potential easy capacity expansion, which provides the ns-3 tutorial labs for beginners with a convenient user interface but without any required system configuration. The learners are able to catch up with the latest networking protocols in a low-cost, repeatable virtual experiment environment, and potentially large-scale network simulation experiments. Our platform integrates the ns-3 tutorial guidelines, reference source codes and various programming functions, which forms a smooth learning flow of from theory learning, source coding to lab practice. In this research-oriented learning of network simulation experiments, beginners are able to concentrate on their learning without being distracted by the simulator configuration and maintenance and are motivated to explore more learning experiences independently. In summary, our contributions are listed as follows.

- We design and implement this web-based ns-3 learning platform, so that a beginner is able to study the ns-3 tutorial lab guidelines step-by-step in a learning-by-doing approach. The back-end of the platform is constructed based on the open-source ns-3 network simulator; hence, the development cost of the platform is effectively reduced but harness the progress of the active ns-3 development.
- In order to increase the scalability of the proposed learning system, we create a ns-3 docker image to generate potentially a large number of isolated ns-3 running containers, which enable the elastic on-demand capacity expansion to support users as needed. The proposed architecture implements a scheduling mechanism to balance the simulation jobs to these ns-3 running containers to minimize the waiting time for learners to receive simulation results.

The rest of the paper is organized as follows. First, we review the related work in Sect. 2. Then, we present the platform design in Sect. 3, and present the implementation details in Sect. 4. Next, we present the performance evaluation results in Sect. 5. Finally, we conclude the paper in Sect. 6.

2 Related Work

In this section, we review the representative work on the online simulation platform in engineering education for communication and networking. Derr et al.

developed a web-based simulation tool, PGCPMT, for power grid communication network simulation. PGCPMT provides a GUI interface for the network topology, allowing users to define a network by dragging and dropping network elements [2]. Zou et al. proposed a teaching platform, EasyHPC, deployed on the supercomputer Milkyway-2 [15]. The front-end of EasyHPC provides an experimental tutorial, and the back-end of EasyHPC provides the environmental support for online programming. This is a feasible case of combining an online programming platform with tutorials. Gao et al. designed a set of ns-3 labs on the EasyHPC online platform such as learning the IEEE 802.11 Wi-Fi protocol, and positive feedback was reported from the students [4]. Sljivo et al. developed an interactive web simulation tool for the IEEE 802.11ah protocol [11]. The tool provides visual results for ns-3 simulation through PyViz, and provides a monitoring view of nodes status. Gao et al. re-designed an ns-3 simulation based networking lab course by reproducing research results aiming to teach both engineering rigor and critical thinking for undergraduate students, which are crucial for their future career or research [5].

Motivated by these previous works, our platform integrates the ns-3 tutorial guidelines, relevant experimental designs and programming functions, which creates a smooth learning curve from the theory learning, lab design and coding practice. In this learning-by-doing approach, this network simulation learning platform effectively reduces the additional time and efforts for beginners to quickly get hands on the ns-3 labs.

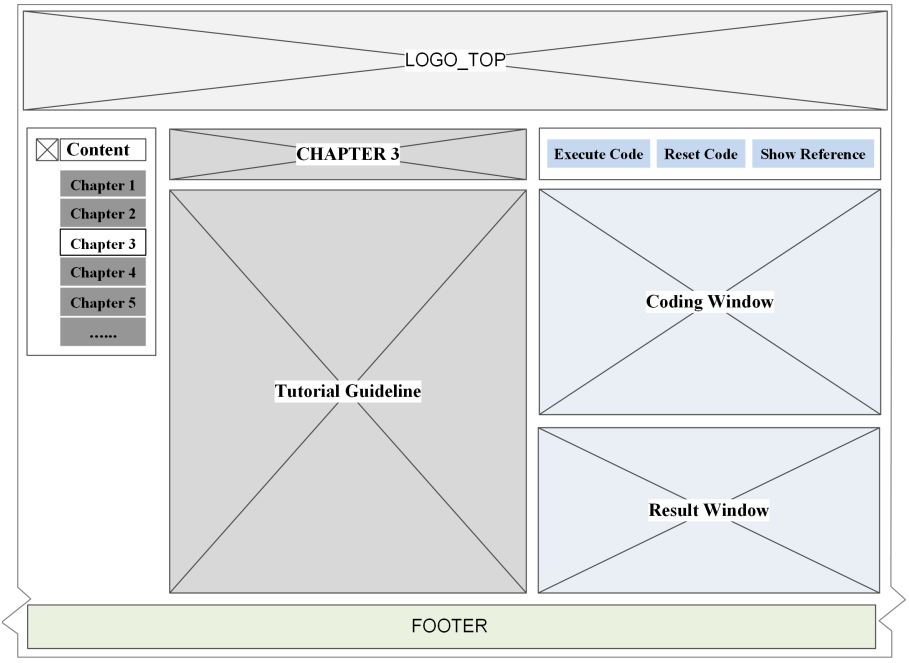


Fig. 1. User interface design

3 System Design

In this section, we first examine the requirements of an online platform in that we should provide learners with a user-friendly web-based interface, an online program execution environment, learning materials and reference codes, etc. Then, we present the system architecture design in details.

3.1 User Interface

As shown in Fig. 1, a learner edit the ns-3 source codes online with a code editor; with a single click on the execute button, the ns-3 source codes can be uploaded to the back-end ns-3 simulator compiler; the simulation experiments are conducted by the ns-3 simulator; finally, the simulation results are returned and displayed on the result page before the learner. Hence, ns-3 learners are able to enhance the comprehension in a learning-by-doing approach with minimum configuration efforts.

3.2 System Architecture

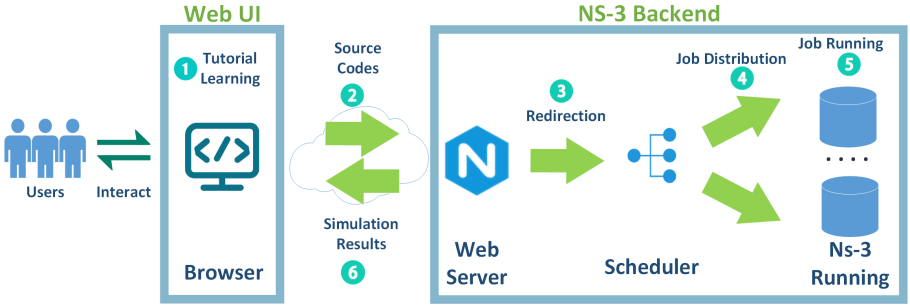


Fig. 2. System work flow

Our platform deploys a typical browser/server (B/S) architecture to meet the needs of user interaction as shown in Fig. 2. The browser provides the user interaction interface, including the tutorial guidelines, the code editor and the result display window. We store the learning materials and the reference codes on the server. When a learner needs to view the web pages, the browser sends an HTTP request to the server. After receiving the request, the server will return the corresponding web pages to the browser, and the pages are displayed on screen.

When a user needs to run codes online, the codes will be uploaded to the server by the browser, and the scheduler in the simulation server distributes the codes to an independent running environment. This selected running environment executes the codes and returns the simulation results. Finally, the results are displayed in the browser.

4 Implementation

In this section, we present the web framework to implement our system prototype. Then, we present the architecture implementation details.

4.1 Web UI

We use a popular frameworks, Vue, to build the front-end. This framework has ready-made component libraries, and we can use well-designed UI libraries to build web sites quickly. We implement the buttons and menu bar based on the antd-vue component library. Then, we download the ns-3 official web site tutorial guidelines and render them on the pages. Additionally, we integrate the code-mirror open source code editor to edit the source codes [11].

4.2 System Architecture

In order to implement the function of the online execution of the simulation experiments, we set up the ns-3 simulation environment on the server. When a learner executes the simulation codes, the browser uploads the codes to the server via HTTP, and the server process the received codes, including compilation, execution, and returning the simulation results and the execution information to the learner.

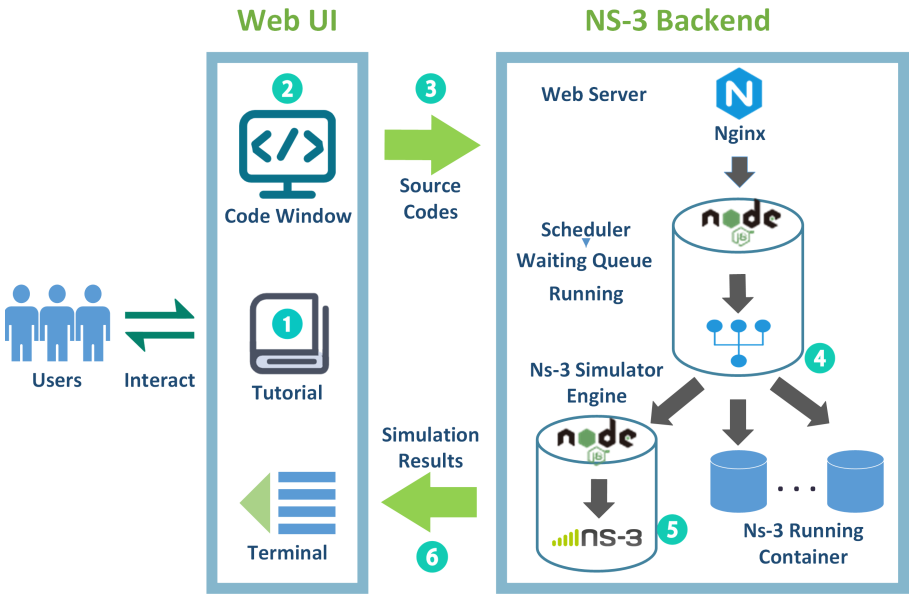


Fig. 3. System architecture

Our system implements the modules as shown in Fig. 3. In this B/S architecture, the server is composed of 3 modules: the Nginx server, the manager container and the ns-3 running container. The Nginx server is a HTTP server, which can display the web files (such as HTML, pictures) on the server to the clients through the HTTP protocol, instead of requesting resources through the server, which could reduce server pressure. It also provides a reverse proxy function to forward the HTTP request of the browser to the manager container. Note that a ns-3 process compiles its source codes in a default folder. In order to maximize the utilization of computing resources, we instrument multiple containers deployed with duplicated ns-3 environments. When the manager container receives the codes, it just forwards the codes to an idle ns-3 running container via HTTP; then, the ns-3 running container compiles the codes and executes the simulations, and returns the results to the manage container. Afterwards, the manage container returns the simulation results to the browser, and the browser displays the results in the output console for learners.

We select the Node.js and restify framework to implement our system. Node.js is used to provide the HTTP service, which is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js is a single-threaded language. It treats the received request as a task and schedules work in a single thread through a task queue, which reduces the overhead caused by thread switching, which will show a significant performance advantage when the service is mainly IO service. Our service is mainly based on IO requests that distribute tasks through HTTP, so Node.js has a great performance advantage. Restify is used to accelerate the creation of HTTP services, which is a REST service framework based on Node.js, and focuses on REST services than modules such as express, which can effectively separate front-end and back-end development, reduce project development cycles, and improve service scalability.

4.3 Simulator Scheduler

In order to increase the scalability of the system, we apply the docker to generate multiple ns-3 running containers from an ns-3 running image. We use the ready-made Nginx image and build the manage node image and the ns-3 running image. We created a Nginx server container, a manager container and a changing number of ns-3 run containers as needed through the Docker-Compose container orchestration tool[1]. Docker-Compose is a tool for defining and running multi-container docker applications by composing a YAML file to configure our ns-3 simulation service.

When the Nginx server receives a ns-3 running request, the request is forwarded to the manage container. The manage container examines whether there is an idle ns-3 container. If not, check again after blocking for a period of time. If there is an idle ns-3 container, the codes are sent to this available ns-3 running container via HTTP. The ns-3 container compiles and executes the receiving

[1] https://github.com/docker/compose.

codes, harvests the simulation results and returns the results to the management container. Finally, the management container returns the results to the learner's browser for display.

5 Performance Evaluation

5.1 Web UI

Interface Outlook. The Web UI of our platform is shown in Fig. 4. The tutorial window is on the left side and the code window on the right side.

The tutorial guidelines are transformed from the official web site of the ns-3 tutorial. We have selected 5 basic chapters, and the chapter switching can be performed by the switch button at the upper left corner.

As shown in Fig. 5, the platform includes 5 basic chapters of ns-3 tutorials for learners to start with as a use-case, including the conceptual overview, tweaking, building topologies, tracing and data collection. A learner can click the directory button in the upper left corner of the Web UI to switch chapters. In addition, the tutorial labs are also extensible, and instructors can customized tutorial labs as needed.

Code Editing. In order to allow learning to have enhanced learning experiences, our platform provides the function of online programming, allowing learners to immediately perform relevant coding training after reading relevant chapters, so as to understand ns-3 programming and network protocols in a learning-by-doing approach with immediate feedback.

As shown in Fig. 6, we integrate an online code editor on the right side of the Web UI, and a learner can edit the ns-3 codes in the code window. After a

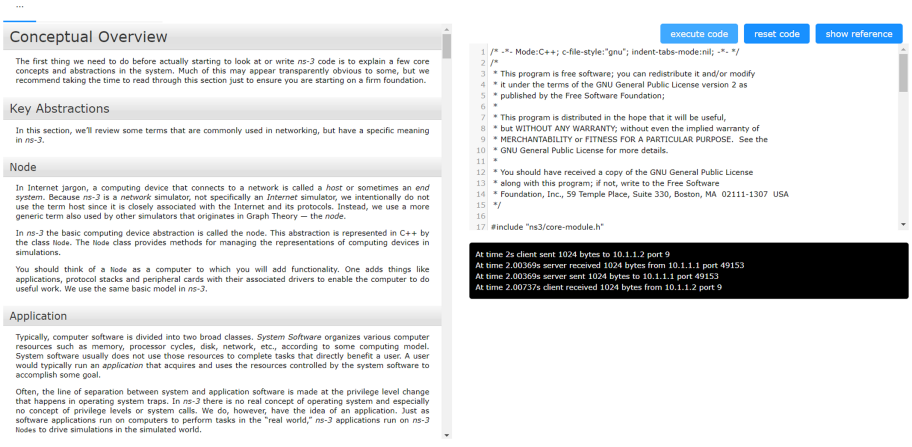


Fig. 4. Web UI overview

learner finishes the coding, he or she clicks the *Execute Code* button to submit the codes to the back-end, which will compiles the codes by the back-end ns-3 engines. The learner can also click the *Reset Code* to reset the code editor to its initial version.

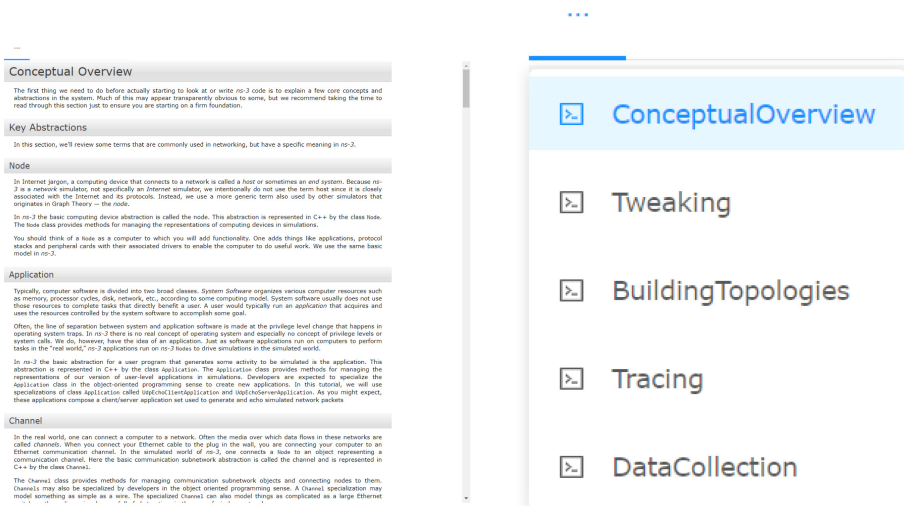
As shown in Fig. 7, the simulation results are returned to the command line of the code window, which are consistent with the real ns-3 console output as well.

Interactivity. In order to enhance interactive user experience, our platform implement the interactive design. After a learner submit the codes, the platform returns the simulation results as soon as the back-end simulator accomplishes the simulation job. Therefore, the learner can correct the errors in the codes based on the output log. If the learner still has troubles in coding correctly, she or he can click the *See Answer* button at the upper right corner of the Web UI to find out how the reference codes work.

Platform Comparison. In summary, our platform integrates three important features of learning, online programming and interactivity. We compare our platform qualitatively with several related platforms as shown in Table 1. Our platform is customized to the need of ns-3 beginners with careful design. Second, our platform instruments several necessary features for most online programming platforms. Though our platform does not yet have the full functions of graphical programming, we have been considering to upgrade the system along this direction in the future development.

5.2 Testbed Setup

In this section, we deploy this ns-3 learning system in a testbed to evaluate the its performance. As shown in Fig. 8, the server is equipped with a 1-core CPU Intel Xeon E5-2682 v4 processor, 2 GB memory, and 1 Mbps network bandwidth. We apply JMeter to conduct the loading test on the server. JMeter is a Java-based stress testing tool developed by the Apache organization, which can be used to measure the system response time of the static and dynamic resources, such as static web files, Java servlets, CGI scripts etc. on the server under different loading levels. We adopt JMeter to simulate different user behaviors by requesting the tutorial web pages, and downloading the reference codes, submitting the simulation codes for experiments. Round by round, JMeter continuously initiates simulation testing circles until lasting for 10 min. We also use JMeter to simulate concurrent requests. When multiple users are simulated, the ramp-up period between the user's requests is set at 1 s. JMeter gradually adds up to the full number of testing threads following the ramp-up period.



(a) Tutorial guidelines (b) Switching button

Fig. 5. Web UI details

Table 1. Platform comparison

Functions	EasyHPC [15]	PGCPMT [2]	Our Platform
Programmability	Yes	Yes	Yes
Results visibility	No	No	Yes
Graphic programming	No	Yes	No
NS-3 tutorial	**no**	**no**	**yes**

5.3 System Performance

Performance Metrics. We measure several important performance metrics of the ns-3 simulation services, including the response time, throughput, and CPU & Memory. The response time is defined as the time duration (second) for a user to download the tutorial pages, fetch reference source codes, execute the codes and retrieve the simulation results. The throughput is defined as the traffic volume per unit of time sent and received by a user (bps). We configure the number of threads of JMeter to simulate different number of users who request the ns-3 simulation services at the same time. We are interested to examine the CPU utilization and the memory consumption (such as swap memory, free memory, buffer size and cache size) of the back end system when the system is stressed with different loads.

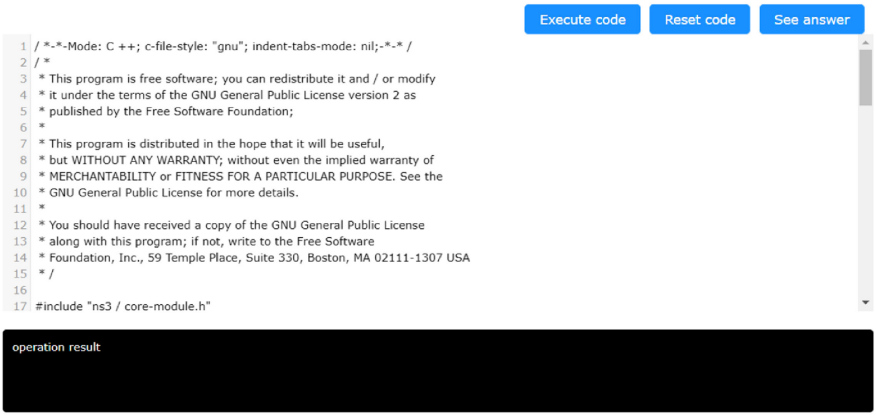


Fig. 6. An online code editor

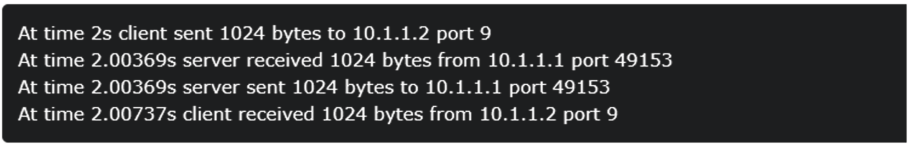


Fig. 7. Simulation results

Table 2. Average response time with different loads and containers (second)

User concurrency	1	2	4
Single container	5.21	5.51	15.01
Double containers	5.27	10.68	11.57
Triple containers	5.24	11.02	11.64

Response Time. As shown in Table 2, the average response time is 5.21 s for the case of the single user and the single ns-3 container. Additionally, the response time of the case of 4 users and the single ns-3 container is 15.01 s, which demonstrates the queueing effect of the simulator scheduler in our platform. If the number of the ns-3 running containers is not sufficient, the response time increases and impacts learning experiences significantly. When the server is lightly loaded with the 1–2 concurrent users, single ns-3 container achieves the least response time. When the concurrent users increase to 4, more ns-3 containers are required to maintain a small response time. The deployment of the container mechanism in our platform enable an elastic capacity expansion to serve an increasing number of concurrent users while maintaining a low response time.

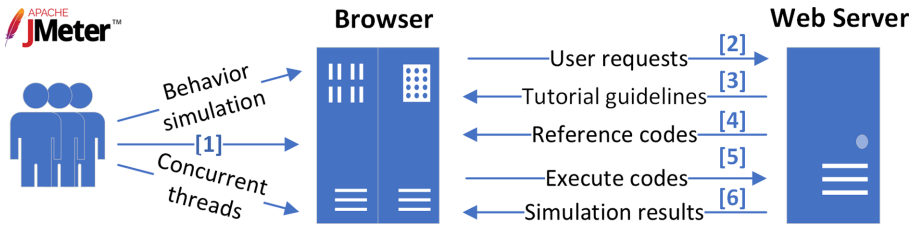


Fig. 8. Stress tests with JMeter

Table 3. Average throughput in different loads and containers (kbps)

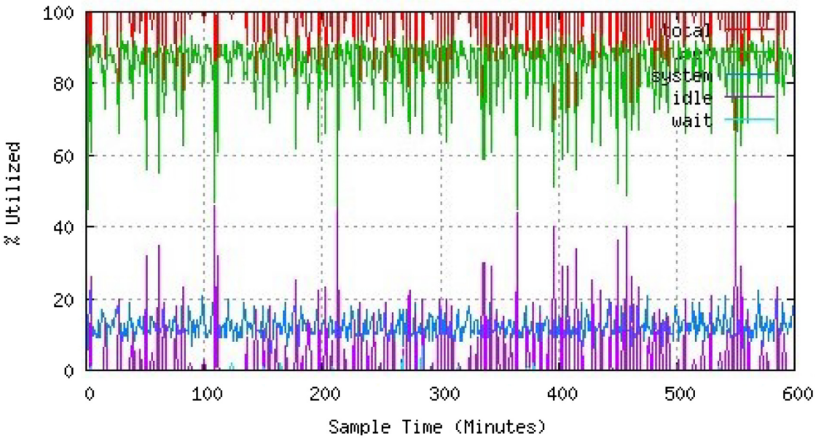
User concurrency	1	2	4
Single container	41.28	80.72	100.48
Double containers	40.96	62.24	118.4
Triple containers	41.04	59.76	117.92

Throughput. As is illustrated in Table 3, the average throughput of a single user and four users are around 41 kbps and 112 kbps, respectively, where the network bandwidth of our server is throttled by 1 Mbps. It shows that a local area network is able to serve the need of ns-3 lab course based on this platform. Thus, network bandwidth may not be the bottleneck of a high-load ns-3 lab platform while CPU and memory may bring forth the major resource bottlenecks.

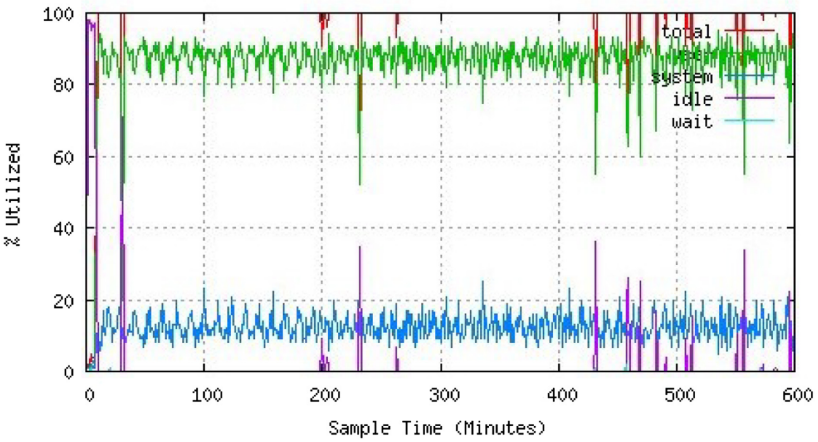
CPU and Memory. We examine the utilization of CPU and memory of the sever to analyze the resource utilization when deploying different numbers of containers when the concurrent users are simulated up to 4.

In Fig. 9, the green line depicts the percentage of the time the CPU runs user-level codes. The dark blue line depicts the percentage of the time the CPU runs system-level codes. The purple blue line depicts the percentage of the time the CPU is idle. The light blue line depicts the CPU usage for waiting I/O devices. When the concurrent user number is 4, the CPU idle time with single ns-3 container is expectedly larger than the double-container case. Because when a process executes a task, it will not completely exhaust the computing resources of the CPU, which means that the computing resources are not fully utilized. While if there are two processes compile, CPU computing resources will be more fully utilized. For triple-container, the concurrent user number is 4 means that there will be 3 task being executed together, which takes about 3 times the time to execute a single task, and the last one will be executed separately. This results in about 3/4 of the time that the CPU idle is extremely low, and the remaining 1/4 of the time CPU idle is slightly higher, just like what shown in Fig. 9. When the concurrent user number is 4, the total user-level and system-level CPU usage is beyond 80% most of the time. It shows that the performance bottleneck of the platform lies in the computing power of the CPU.

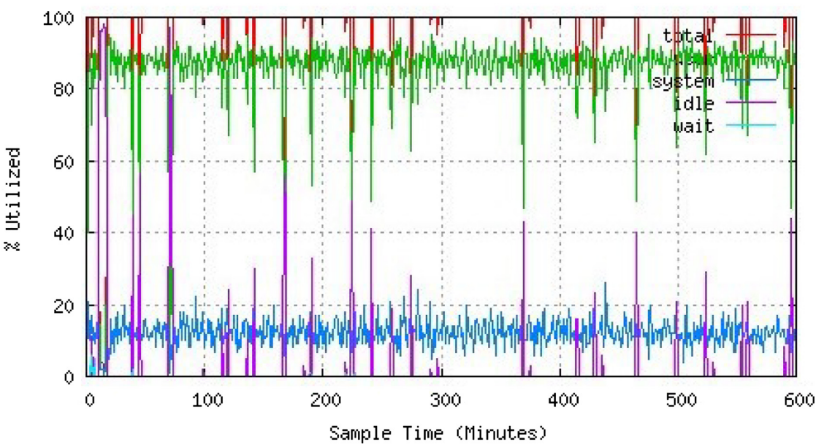
As shown in Fig. 10, with 4 concurrent users, we compare the memory consumption with different ns-3 running containers. “Swapped” depicts the amount



(a) Single ns-3 Running Container

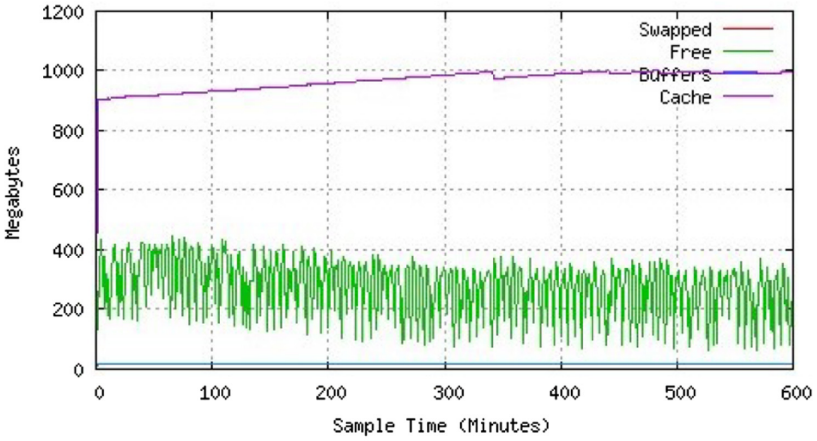


(b) Double ns-3 Running Containers

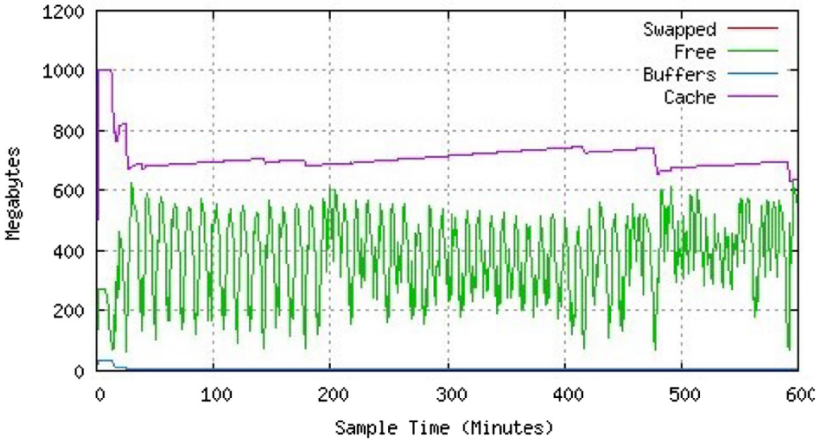


(c) Triple ns-3 Running Containers

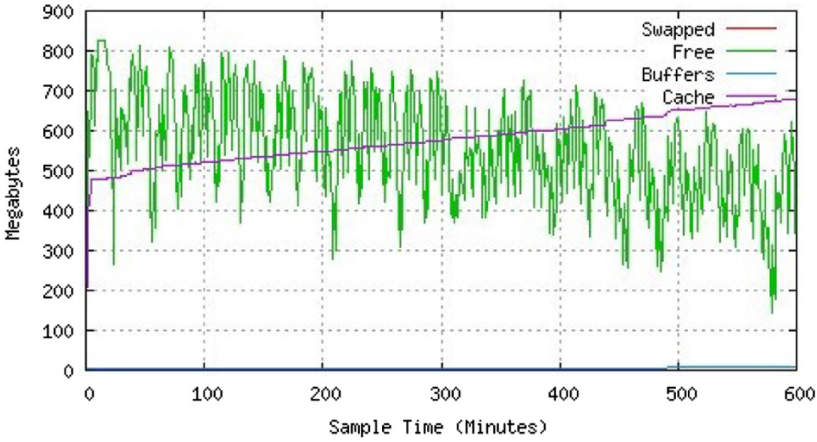
Fig. 9. CPU utilization 4 concurrent users



(a) Single ns-3 Running Container



(b) Double ns-3 Running Containers



(c) Triple ns-3 Running Containers

Fig. 10. Memory consumption with 4 concurrent users

of virtual memory used. "Free" depicts the amount of free memory. "Buffers" depicts the amount of memory used for buffers. "Cache" depicts the amount of memory used as the page cache. The fluctuating base of free memory is affected by the system environment during a test, so the memory fluctuation range and frequency can better reflect the memory usage of the task. Just like the purple curve in Fig. 9, the free memory fluctuation frequency with single ns-3 container is more frequently than the double-container case, and the range is less. This means when there are both tasks being executed at the same time, the peak memory usage will be higher and the free memory changes more smoothly. For triple-container, about 3/4 of the time that the free memory fluctuation frequency is lower and the range is higher, while the remaining 1/4 of the time that the free memory fluctuation frequency is higher and the range is lower. Consequently, when the number of ns-3 running containers executing Simultaneously increases, the CPU utilization increases, the idle time of CPU decreases, and the free memory fluctuation frequency reduces, the range increases. In summary, an increase number of containers will increase the load on the server CPU and reduce the memory consumption.

6 Conclusion

In this paper, we design and implement a web-based ns-3 learning platform [2] to provide beginners to learn computer networking protocols in an easy learning-by-doing approach. Our platform integrates various tutorial lab modules with convenient learning resources, which smoothes the learning curves. The platform supports simultaneous usage by multiple users, and returns simulation results for users with a web user interface, which effectively reduces the difficulty of getting hands on with ns-3. This platform can be used for lab course learning as well as personal learning and research. On the other hand, there are some rooms for improvement along several aspects: 1) asynchronous mechanism: the back-end uses a scheduling mechanism to distribute the user's compilation requests, and then uses the container cloud as the load balancing for ns-3 tasks; 2) topology customization: use an XML parser to implement visual network icons dragged and dropped by users for configuring network topologies in ns-3 programming labs; 3) user management: for student users, the platform provides a login mechanism so that the student users are able to code and submit labs in the back-end database; for teacher users, the platform provides the lab customization module to the support specific learning outcomes of a lab course.

Acknowledgment. The authors would like to express their gratitude to the anonymous reviewers for their constructive comments which the quality of this paper very much. This work was supported in part by the National Natural Science Foundation of 61972172) and the teaching research fund by the Huazhong University of Science and Technology (no. 2018077).

[2] http://cloud.eic.hust.edu.cn:8585.

References

1. David, A., et al.: Reproducible computer network experiments: a case study using popper. In: Proceedings of the 2nd International Workshop on Practical Reproducible Evaluation of Computer Systems, pp. 29–34 (2019)
2. Derr, K.: Ns-3 web-based user interface: power grid communications planning and modeling tool. In: Proceedings of the Workshop on Ns-3, pp. 93–100 (2016)
3. Gao, Y., Peng, J., Yin, Y., Hei, X., Wang, X.: Improving a software/hardware integrated computer networking laboratory course. In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 1189–1192 (2018)
4. Gao, Y., Peng, J., Yin, Y., Hei, X., Wu, D.: Developing wireless networking labs for MOOC learners on an online programming platform. In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 1154–1157 (2018)
5. Gao, Y., Zhang, C., Zhong, G., Hei, X.: Learning networking by reproducing research results in an ns-3 simulation networking laboratory course. In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) (2019)
6. Gawłowicz, P., Zubow, A.: Ns-3 meets OpenAI Gym: the playground for machine learning in networking research. In: Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 113–120 (2019)
7. Gupta, S., et al.: Open-source network simulation tools: an overview. Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET) **2**(4), 1629 (2013)
8. Hei, X., Cheng, W.: Work in progress: fostering a telecommunication engineering pipeline: a curriculum design. In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 258–261 (2015)
9. Hei, X., Cheng, W.: Developing a telecommunication engineering pipeline of communication networks. In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 185–189 (2016)
10. Hei, X., Wen, H., Cheng, W., Huang, X.: Boosting computer-assisted telecommunication engineering education in internet thinking. In: Proceedings of ACM Turing Celebration Conference - China, pp. 123–124 (2018)
11. Šljivo, A., Kerkhove, D., Moerman, I., De Poorter, E., Hoebeke, J.: Interactive web visualizer for IEEE 802.11ah ns-3 module. In: Proceedings of the 10th Workshop on Ns-3, pp. 23–29 (2018)
12. Yan, L., McKeown, N.: Learning networking by reproducing research results. SIGCOMM Comput. Commun. Rev. **47**(2), 19–26 (2017)
13. Yin, H., et al.: NS3-AI: Fostering artificial intelligence algorithms for networking research. In: Proceedings of the 2020 Workshop on Ns-3, pp. 57–64 (2020)
14. Yin, Y., Gao, Y., Hei, X.: Performance evaluation of a unified IEEE 802.11 DCF model in ns-3. In: Song, H., Jiang, D. (eds.) Simulation Tools and Techniques, pp. 395–406 (2019)
15. Zou, Z., Zhang, Y., Li, J., Hei, X., Du, Y., Wu, D.: EasyHPC: an online programming platform for learning high performance computing. In: IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), pp. 432–435 (2017)