On enabling remote hands-on Computer Networking Education: the NITOS testbed approach

Nikos Makris[⋆†], Virgilios Passas[⋆†], Apostolos Apostolaras[⋆†],
Theodoros Tsourdinis[⋆], Ilias Chatzistefanidis[⋆] and Thanasis Korakis[⋆†]
[⋆]Dept. of Electrical and Computer Engineering, University of Thessaly, Greece
[†]Centre for Research and Technology Hellas, CERTH, Greece
Email: nimakris@uth.gr, vipassas@uth.gr, apaposto@uth.gr,
ttsourdinis@uth.gr, ichatsist@uth.gr, korakis@uth.gr

2023 IEEE Integrated STEM Education Conference (ISEC) | 979-8-3503-0001-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISEC57711.2023.10402230

***Abstract*—Education in recent years has slowly transitioned to an online model, allowing massive access to online courses virtually from anywhere. The adoption of such educational models was boosted by the global pandemic in 2020, with universities and other degree programs quickly transitioning to such schemes. Although such a model is apt for lecture-based courses, hands-on training remains a puzzle on how it can transition to remote learning. In this work, we describe and evaluate our scheme for integrating testbed resources in online-taught networking-related courses in University of Thessaly, Greece. The framework is based on Kubernetes and is able to deliver hands-on labs related to networking as micro-services over the testbed architecture with minimal overhead on the lab setup from the instructor. The proposed approach has been applied in the networking-related courses of the curriculum during the 2020-2021 and 2021-2022 academic years, educating more than 800 students on computer networking concepts in practice. The paper describes the framework and a benchmarking evaluation, which proves the capacity of the framework to serve up to 5 times higher numbers of students, compared to prior methodologies and practices, without any infrastructure upgrades.**

***Index Terms*—Online Education, hands-on labs, testbed experimentation, Kubernetes, micro-services**

I. Introduction

The evolution of networking technologies and testing platforms with open-source software and platforms creates fertile ground for the integration of hands-on based courses for the computer networking curricula in under- and post-graduate levels. Education on Computer Networking can benefit from this approach, allowing students to get an actual experience on real networks, through their configuration and debugging in real-time, by thoroughly comprehending and applying the theoretic concepts taught through lectures [1]. In turn, the hands-on training can expose them to real-world problems, help them identify the industry needs, and subsequently present them with better offerings in the job market.

At the same time, the appropriate tools for deploying massive clusters of inter-networked services has been enabled through the rise of micro-services and accompanying tools [2]. Through the adoption of the container technology (e.g. docker [3]), and advances in the orchestration software such as the Kubernetes (K8s) [4] framework, the effortless deployment is enabled for isolated inter-networked containers with specific characteristics, instantiated even on the same hardware. Such functionality can be highly beneficial for educational purposes, given the abundance of testing facilities e.g. available testbed platforms such as NITOS [5], Fed4FIRE [6], ORBIT [7] and PAWR [8] that can provide the respective compute infrastructure. In this work, we capitalize on an existing testbed facility located in University of Thessaly in Greece, and present our contributions in developing the respective functionality for supporting computer networking courses with hands-on labs, realized as docker micro-services. The proposed approach is a paradigm shift on the manner that such networking courses have been taught in similar facilities, as they have been usually relying on physical access to the testbed resources, which can be of limited scale in most cases, hindered by the available resources and the overhead for the lab configuration for several attendees. Through the extended use of the Kubernetes framework, we are able to deploy massively labs that support either the basic networking courses or the more advanced, covering also complex wireless networking experiments, while significantly easing the overhead placed on the instructors to configure and parameterize each lab.

From the side of the students, the only requirement is a computer with a network connection to the testbed that hosts the experiments. The developed framework is generic enough to be applied to any other Linux-like cluster of nodes, with only a small number of experiments requiring the use of dedicated hardware. Through multiple designed activities, the participating students experience the role of a network provider, with activities including even the deployment of their own 4G network. The developed functionality can serve as the bridging technology between MOOC platforms and actual testbed-driven experiments, allowing for the meaningful interpretation of collected results and comprehension of networking protocols in practice.

In this context, a framework enabling the wide effortless access to actual testbed resources has been developed in University of Thessaly, enabling the execution of several lab-based experiments over actual networking equipment, and integrated into the curriculum of three different networking courses: 1)Introduction to Computer Networking, 2) Inter-network protocol design, and 3) Advanced topics on Computer Networking. The framework enables students to get hands-on experience on the theoretical concepts that have been lectured, while the tutors

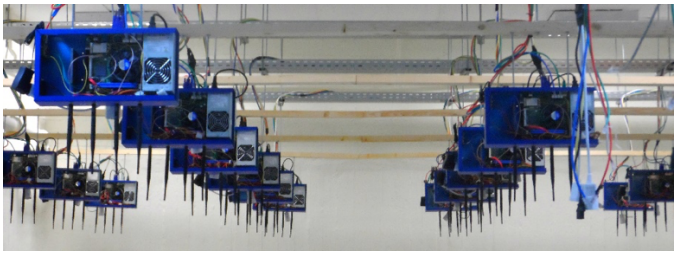


Fig. 1: NITOS testbed overview; all the physical nodes are available to be used as bare metal machines, hosting labs as micro-services. In the figure, an overview of the indoor testbed nodes is shown. All the nodes of the testbed are interconnected under the same network.

enjoy effortless deployment of several advanced hardware labs, that would otherwise be effort- and time-consuming to develop and deploy. The developed approach was tested during the 2020-2021 and 2021-2022 academic years, serving more than 800 students that attended these classes remotely, and is currently in service for the academic year of 2022-2023.

The rest of the paper is organized as follows: Section 2 presents the hardware elements that are provided in a testbed setup, allowing the setup and configuration of complex experiments over the infrastructure. Section III presents our framework and the developed labs that integrate with the courses. Section IV shows some benchmarking results for the platform, proving that the capacity of the framework is very high, enabling the support of even more students concurrently. Finally, in Section V we discuss our work and in Section VI we conclude, presenting some future directions.

II. Testbed Deployment

A. Testbed Description

The target facility used for the development, application and evaluation of the hands-on experiments for the networking courses is the NITOS testbed (http://nitos.inf.uth.gr), located in University of Thessaly, Greece. The testbed is providing in a 24/7 fashion remotely accessible resources, targeting at experimentally driven research in wireless and wired networks. In this section, we provide a very brief description of the capabilities of the testbed. The testbed is providing access free-of-charge to over 100 static physical nodes, equipped with key networking technologies:

1) All the nodes are high-end PCs, equipped with Core-i7 processors and 8 GBs of RAM each, featuring at least two IEEE 802.11 a/b/g/n/ac cards, compatible with Open Source drivers (e.g. ath9/10k) used for WiFi research.
2) Two commercial off-the-shelf LTE access points are available for experimentation, along with a commercial off-the-shelf Evolved Packet Core (EPC). Both femtocells and EPC are programmable through the available testbed services. About half of the nodes are equipped with LTE dongles, enabling the establishment of an operator-grade 4G network, using testbed specific SIM cards.
3) Over 20 different SDR devices are installed in the testbed, which are compatible RF front-ends for open source implementations of 4G/5G and beyond networks such as OpenAirInterface [9].
4) Six mmWave devices are installed in the testbed, reachable from all the nodes, supporting the creation of high-throughput wireless point-to-point links. The nodes support beam steering, allowing the formation of different mmWave topologies.
5) All the testbed nodes are interconnected through three hardware OpenFlow switches, organized in a tree topology.

Figure 1 illustrates the deployed testbed infrastructure. The testbed is organized in three different setups: An indoor RF-isolated setup, an outdoor configuration prone to uncontrolled external interference, and an office setup with mild interference settings. Resources can be mixed from different locations in order to create a versatile experimentation environment.

Testbed-driven research has gained lots of attention during the last decade, as it is able to overcome ambiguities and misconfigurations when developing new solutions and protocols, that are commonly induced through simulation-based tools [10]. Examples of such ambiguities among different simulation tools are prominent especially in wireless network simulators, regarding the configuration of the physical parameters of the medium and its environment (e.g. signal propagation models, assuming that the terrain is flat, etc.). Boosted by the adoption of open-source software, and the availability of different protocol implementations, researchers can take advantage of experimenting with their novel concepts in practice and real-world environments, competing with existing solutions under the same settings.

B. Testbed provisioning for Labs

The testbed offers a number of different tools for experimenting with the infrastructure. These tools rely on a scheduler interface, through which the experimenters can book the testbed nodes for a predefined timeframe, and get bare metal access on the nodes. However, for the needs of quickly deploying experiments that can be used by a large mass of users (e.g. up to 400 students participating in a single course), the access scheme is changed, and provisioning of the lab materials takes place through a different set of tools.

Given the aforementioned needs for deploying large scale experiments in a fast manner, the use of micro-services becomes apparent. Their lightweight nature, along with the ability to instantiate large swarms of services even in a single compute node make them the preferred choice compared to Virtual Machines. Although engines like Docker or LXD can provide the baseline functionality for deploying such services, the use of a higher level orchestration tool is desired in order to allow efficient, scalable and isolated scheduling and deployment across large clusters of nodes. To this aim, although several tools exist, like for example Nomad [11] or Eclipse Fog05 [12], we select to leverage the Kubernetes (K8s) framework as it is at the moment one of the most

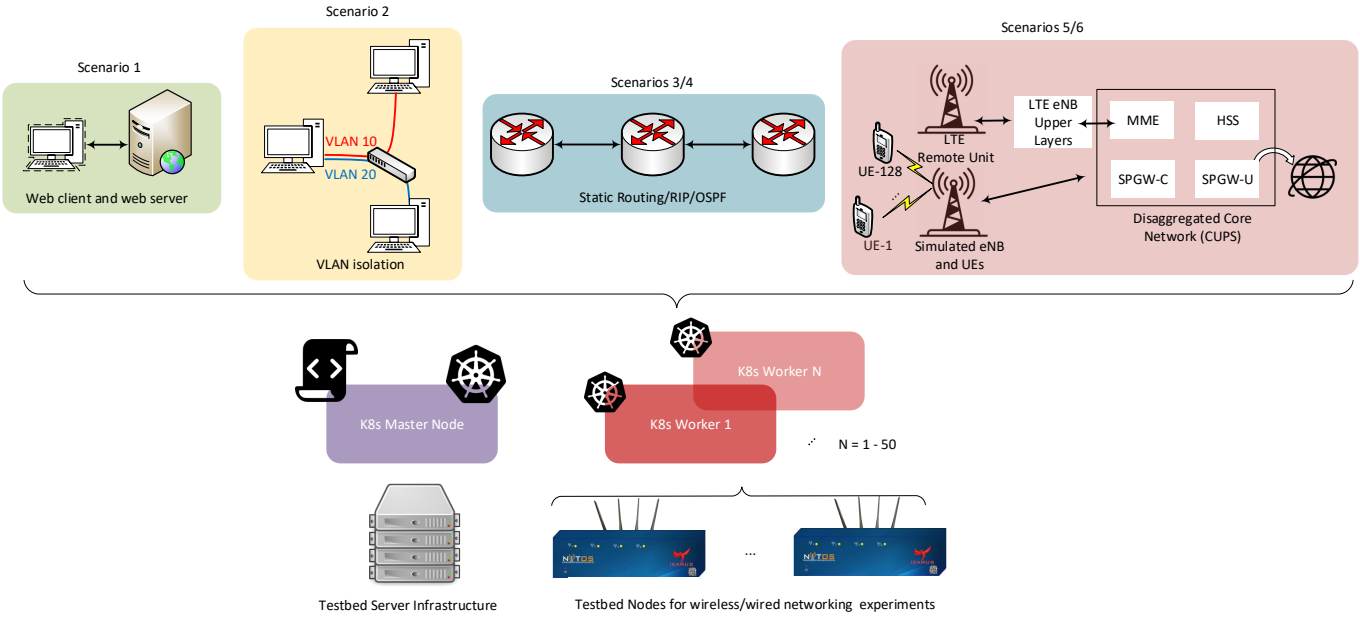


Fig. 2: Testbed configuration for running the networking lab scenarios

widely adopted tools for the orchestration and management of container-based deployments.

Kubernetes is using a control-plane and worker setup, where the control-plane node is used to orchestrate containers over a pool of different workers. K8s is offering isolation of the different containers through different namespaces, or using network slicing with plugins like Calico[1] or Flannel[2]. The labs are deployed as a separate set of containers, over the same infrastructure for multiple users, as shown in figure 2. The control-plane node is running as a Virtual Machine (VM) in the testbed infrastructure, which has direct network access to the wireless nodes of the testbed. Each of the testbed nodes is configured as a worker node for the K8s control-plane node, and therefore containers can be orchestrated on top of them. The methodology for running each of the experiments is the following. The instructor goes through the normal process of reserving the nodes through the testbed tools, and prepares the worker nodes by using the testbed tools to load an image on them (using the cOntrol and Management Framework [13]). After this initial preparation, the instructor uses the control-plane K8s VM to deploy the respective scenarios over the testbed. The students access only their designated set of pods, and do the lab assignments.

The developed approach is a significant improvement on the effort that the instructor has to devote for preparing and running the designated labs. Prior to the adoption of the framework, the instructor had to go through the reservation of the testbed nodes, loading specific images on the nodes (dedicated to each lab), while also deploying the appropriate tools and bootstrapping functions so that the labs are automatically executed. Several other tools and firewalls needed to be employed on each node in order to enable the isolation of the flows among different students. A similar process needs to be dedicated when preparing a new lab, for preparing the baseline image. Through the adoption of the K8s platform, the instructor only has to prepare the respective micro-services for the framework, and the framework will take care of deploying and isolating them from other instances. Moreover, the adoption of such a framework creates fertile ground for the deployment of advanced monitoring solutions, such as the Prometheus operator [14], that provide fine-grained information on the allocation of resources, the consumption of resources per each container, etc. This can take place in real-time, during the lab execution, centralized and visualized through tools like Grafana [15]. In the following section we present some indicative material taught in each lab segment.

[1]https://docs.projectcalico.org/
[2]https://github.com/flannel-io/flannel

III. Lab Development and Deployment

The developed labs are used in a number of different computer networking courses, either introductory to networking concepts, or more advanced regarding wireless network operation. The labs are developed in six different scenarios, around the four topics outlined below, and illustrated in figure 2.

A. Web-services

In this lab, students delve into the basics of a web transaction and learn about the client-server model. The students get access to a pair of containers, one of which is running a NGINX web server, and a second one used as the web-client. By using tools provided in the container (*wget, curl, Chromium browser*), they send requests to the web server and observe the responses. As a second step, they implement a

web client using TCP/IP sockets, and retrieve the served web page through their own client. Delving into the specifics of the HTTP protocol, the students use the *tcpdump* tool to observe how the HTTP protocol is expressed at the packet level, and the overall network exchange.

B. Flow Isolation

In this lab, the students learn how different flows in the network can be isolated from each-other by using the VLAN technology. Each student gets access to three containers (e.g. A,B,C), which are configured to communicate over different VLANs. For example, container A is communicating with container B over VLAN-X, and with container C over VLAN-Y. In such a setup, traffic transmitted from container B is not captured by any of the container C network interfaces. Using simple traffic generators (e.g. *ping* command) and the network capture *tcpdump* tool, the students generate traffic and observe how the VLAN configuration enforces the isolation of traffic flows in the network.

C. Network Routing

This lab is dedicated to routing networking packets. The lab is divided to two different scenarios as follows. In the first scenario, the students get access to three containers, configured to use two different networks. One of the containers has access to both networks, and can act as a router. The goal of the lab is for the students to experiment with the *ip route* commands and configure the routing rules, that will enable the different containers to communicate with each other. In the second scenario, the deployed containers launch the *Quagga* software, which enables the execution of a networking protocol (e.g. RIP/OSPF/BGP) over the different containers. Using network capture tools such as *tcpdump* and *Wireshark*, the students observe the specific message exchanges for each protocol between the different containers, complementing the knowledge they get through the theory sessions of the course.

D. Cloud-native Radio Access Network

This lab is oriented for courses dealing with wireless networking, demonstrating how network softwarization enables the on-the-fly instantiation of entire cells in a mobile broadband networking scenario. Through the lab, the students are provided with a larger number of containers as follows: 1) a set of four containers, configured to run a disaggregated instance of an Evolved Packet Core (EPC) Network for a 4G networking cell - the EPC is configured to run with the latest feature of Control and User Plane Separation (CUPS) of the 4G Architecture, 2) one container running the upper part of a disaggregated base station (Cloud-RAN), using a Software Defined Radio (SDR) front-end for transmitting data over the air, 3) one container running a simulation environment for a base station and up to 128 users attaching to it. All the containers are based on software from the OpenAirInterface (OAI) platform. The students run two different scenarios: 1) using the EPC containers, and the base station setup, which enables them to observe the cell transmitting over the air using spectrum sensing tools, or 2) using the EPC containers and the simulator container, where they get to attach a number of different users and observe the signalling exchanged from the user, to the base station and subsequently to the EPC, for admitting a user to the network. This lab takes place in either computer networking introductory courses to demonstrate an all-software based network, or in more advanced networking lessons where the users learn the basics of operation for the 4G wireless network. It is currently being further extended in order to teach different aspects of signaling in cellular networks, as well as different interfaces, GPRS tunnels, etc.

IV. Evaluation

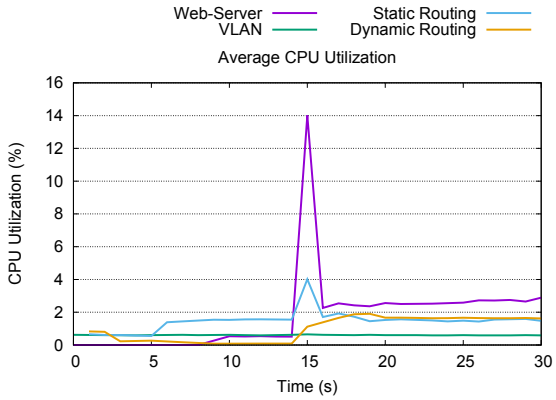
In this section we present some benchmarking results for the developed labs. For the first three labs, we use a single worker cluster and instantiate the maximum number of containers that can be supported. The limitations on the number of the supported containers over each worker stem from the actual K8s platform (v1.18) [3], as it has been designed in a manner to allow large deployments of services, while also ensuring that they are working as expected. As this number for the K8s version that we are using is limited to 100 containers per each worker, this corresponds to 50 isolated labs for the web-services lab, and 33 for the flow isolation and network routing labs. For the cloud-native RAN lab, we use either a single cluster node for the cases of a simulated RAN, or a two-/three- node setup for the real network deployment labs.

By deploying massively the designed labs in the most constrained environment, we aim to quantify the maximum number of students that can be supported in the testbed through platform. Our evaluation is based on monitoring how resource metrics (CPU, memory and network utilization) on the host node are affected. The Prometheus monitoring suite is employed for retrieving telemetry data with a granularity of 5 seconds between consecutive measurements from the container-hosting nodes.

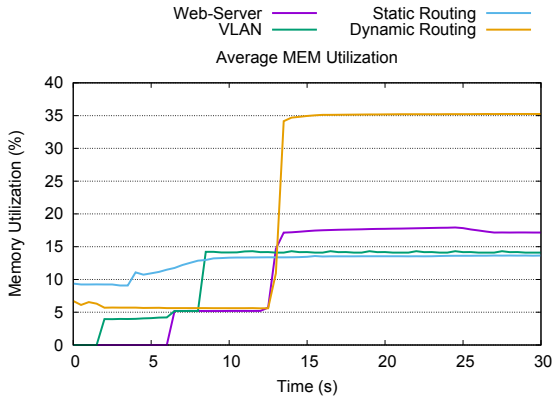
Figures 3 and 4 show the resource consumption for the first three labs. At around 15secs of the experiment, we start the network exchange and start the services providing the functionality needed for the labs. For the web services case, we see that CPU and memory consumption rises around that point, with CPU on the host node being utilized up to 14% by the 100 containers. On the other hand, memory usage peaks at around 18% of the 8 GBs present on the host node. Network utilization on the other hand, is constantly rising its rate, peaking at around 70KBs/minute; this means that for every minute, the network exchange is rising with 70KBs more than the previous measurement. Given the high capacity links that are interconnecting the nodes, such a rise is not a concern.

For the flow-isolation lab (VLAN), we see that it is not demanding as all the resources seem to remain under-utilized for the entire period of the experiment. The routing lab is broken down to the two different scenarios, the static and the dynamic routing. For both cases, no notable resource

[3]https://kubernetes.io/docs/setup/best-practices/cluster-large/



(a) Average CPU Utilization on the host node



(b) Average Memory Usage on the host node

Fig. 3: CPU and Memory statistics for the four under-study scenarios

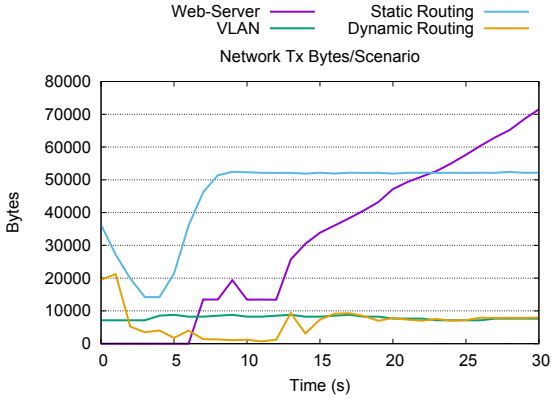


Fig. 4: Net usage statistics per scenario

consumption is shown. For the dynamic routing case, at the time during which the routing algorithm starts to work, we see an increase in the memory usage up to 35% of the host. However, as these results demonstrate measurements from containers that run on the maxed-out node (up to 100 containers on the node), performance is not a concern for these types of labs.

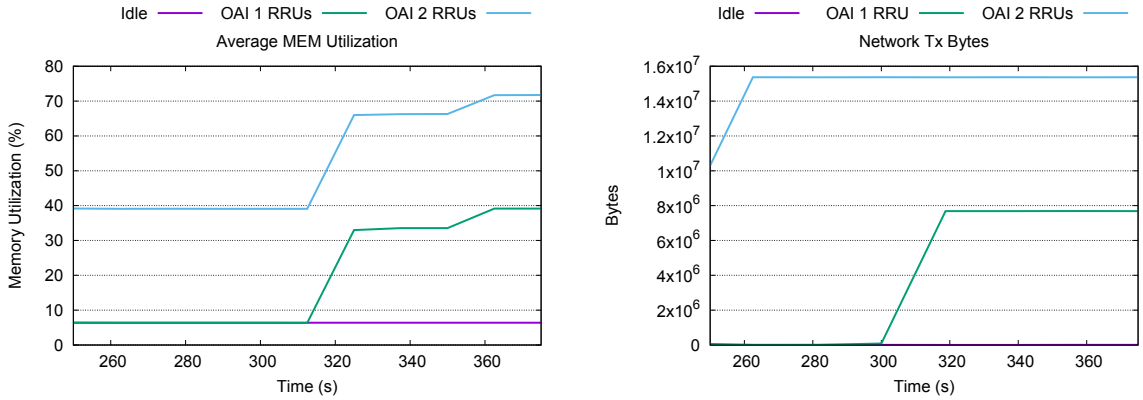
Regarding the evaluation of the cloud-native RAN labs, we present similar benchmarking results in figures 5 and 6. Figure 5 presents the memory and network traffic exchange for the case of instantiating one or two disaggregated base stations along with the respective core network, where the Radio Unit/s (RRU) is/are running on a separate node. We use the 3GPP Option 7.2 split [16], which means that processing tasks up to the modulation and signal encoding for the base station are captured in our measurements. Thus, the figures depict the memory and network exchange of the processing tasks of the base station, before the transmission of data over the air. Based on the node capacities, we observe that memory utilization reaches approx. 36% for each base station. Therefore, each node can support up to two base stations, in terms of processing.

Figure 6 shows the respective memory utilization for the case of running a simulated RAN (OAISIM) and User Equipment (UE). As we see, the network deployment consumes approx. 33% of the host node. At the point denoted with a red line, we start and attach sequentially up to 128 UEs to the network. We see that there is a significant allocation of memory at around 96 UEs, and the overall allocation of resources is reaching up to 52% of the available memory on the host node. This essentially means that one node can be used for orchestrating experiments for two different students, when attaching up to 96 UEs per each student.

V. Discussion

The presented results show that per each lab a different number of hosting nodes is needed. For instance, the first three labs do not exceed the limits of resources per each node, and are restricted only by the K8s framework that allows only up to 100 dockers per each compute node to be instantiated. On the other hand, the more advanced labs on cloud-native RANs can host up to two labs on each node, with some limitations (e.g. reducing the number of UEs attaching to the network). The first set of labs are more applicable to introductory networking courses, which have a massive number of students attending, approx. 400 students/year. For such courses, the current setup allows the configuration of the hands-on labs with less than 20 physical nodes in total. The more advanced labs on cloud-native RANs are instructed in the more advanced networking courses, which are not mandatory courses, and therefore have a smaller number of students (approx. 50 per year). This allows the execution of the labs with less than 25 physical nodes, which is exactly half of the offered resources in the NITOS testbed. Based on the current offering of testbed resources, we conclude that the testbed can be used for serving concurrently up to 2500 students for the simpler experiments, and up to 100 for the advanced ones.

Nevertheless, the most significant gain on the developed methodology is the configuration and deployment overhead for the hands-on experiments. When using the methodologies, instructors needed to invest in time for pre-provisioning the lab environment, by employing different tools for configuring



(a) Average Memory Usage on the host node

(b) Network bytes exchanged over the host node

Fig. 5: Memory and Network statistics for the cloud-native RAN lab

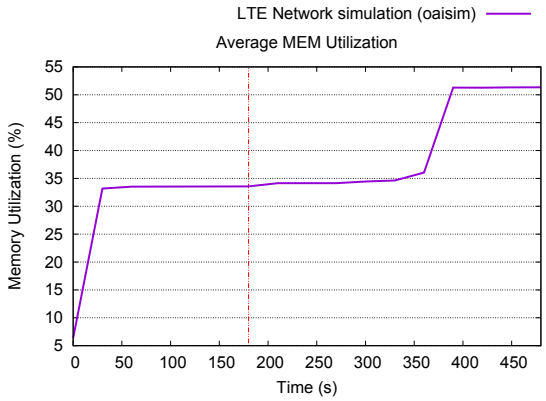


Fig. 6: Average Memory Usage on the host node: red-dotted line denotes the point after which we attach 128 UEs sequentially

the experimentation environment. Through the aforementioned contributions, the labs can be deployed in a single-click manner, regardless of the number of students that participate in the courses, thus relieving the overhead in preparing the labs from the instructors.

Nevertheless, although the adoption of such framework setting new paradigms in the manner of experimenting with the testbed, the proposed setup suffers from several limitations. First and foremost, as some of the experiments need dedicated hardware (specific SDRs), the number of parallel instances is bounded by the maximum number of available devices. Such limitation can be be resolved through the adoption of virtualization over the SDR units, that would allow parallel usage of the same device from multiple processes/users on top, yet no such viable solution is currently available. Similar limitations exist for the maximum number of containers that can be hosted per node or per cluster, as they are limited by the core K8s framework. Although the framework is constantly extended to support more resources, such limits still exist and can limit the maximum number of parallel instances.

The proposed framework can help towards bridging the gaps between traditional lecturing and hands-on training for future network engineers. As more courses are taught online, the training aspect of the teaching process might be underdeveloped, since the norm is that the students do not have access to costly equipment. On the other hand, frameworks such as the aforementioned one can help towards providing training resources from available testbeds, that are equipped with cutting-edge equipment, and integrate them in the learning process. Such hands-on training can help towards materializing the knowledge from lecture/online-taught courses, and possibly integrate them within a MOOC platform; this will enable the execution of repeatable and reproducible experimental results in real-time through the platform. Such an effort is currently in progress for the aforementioned courses (http://web.nitlab.inf.uth.gr/advanced_topics/). Similarly, the developed Kubernetes deployment files are publicly available [4], in order to allow reproducibility of the results in other clusters as well. Through the constant upgrade of the testbed, the overall vision is that such courses will be provided easily through exclusively online tools in the coming years. This in turn opens up other possibilities for the further exploitation of similar research infrastructure in the future. As there are emerging discussions on the exploitation and sustainability of such research infrastructure in the beyond 5G era, the educational approach and the switch to similar tools utilizing micro-services is something that is worth considering [17], [18]; such tools are able to cover the vast majority of experimentation needs, based on the types of experiments already conducted in the facilities while enabling more experiments to be executed concurrently and making more efficient use of the resources.

VI. Conclusion

In this work, we presented our contributions towards integrating hands-on labs in remote education. Towards this goal, we build on top of an existing testbed infrastructure, that is remotely accessible, and enhance it with a framework that

[4]https://repo.nitlab.inf.uth.gr/educational-labs/ece-networking-labs

allows the effortless deployment of massive numbers of hands-on labs for networking courses. The framework is based on the Kubernetes platform, while the labs run as micro-services, and can be deployed in a single-click manner. We evaluated and benchmarked the developed labs in the testbed environment. Our analysis showed that the current testbed infrastructure can serve more than 2500 users concurrently for the simple types of labs, and up to 100 concurrently for the more advanced resource-intensive ones. In the future, we foresee the integration of the lesson examination process with the testbed, as well as the development of more courses, enabling advanced concepts to be taught, including Multi-access Edge Computing with follow-me approaches (cases when the edge service is migrated according to the location of the client), wireless backhauling using novel wireless communication methods (e.g. in the mmWave wireless spectrum), and different levels of base station disaggregation.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Horizon 2020 Programme for research, technological development and demonstration under Grant Agreement Number No 101008468 (H2020 SLICES-SC). The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only.

REFERENCES

[1] L. Xu, D. Huang, and W.-T. Tsai, "V-lab: a cloud-based virtual laboratory platform for hands-on networking courses," in *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, 2012, pp. 256–261.

[2] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, *Microservices: Yesterday, Today, and Tomorrow*. Cham: Springer International Publishing, 2017, pp. 195–216. [Online]. Available: https://doi.org/10.1007/978-3-319-67425-4_12

[3] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.

[4] D. Bernstein, "Containers and cloud: From LXC to docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[5] N. Makris, C. Zarafetas, S. Kechagias, T. Korakis, I. Seskar, and L. Tassiulas, "Enabling open access to LTE network components; the NITOS testbed paradigm," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–6.

[6] P. Demeester, P. Van Daele, T. Wauters, and H. Hrasnica, "Fed4fire: the largest federation of testbeds in Europe," in *Building the future internet through FIRE*, 2016, pp. 87–109.

[7] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 3. IEEE, 2005, pp. 1664–1669.

[8] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejski, M. Sherman, Z. Kostic, X. Gu *et al.*, "Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–13.

[9] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.

[10] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET Simulation Studies: The Incredibles," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 4, p. 50–61, oct 2005. [Online]. Available: https://doi.org/10.1145/1096166.1096174

[11] Hashicorp, "Nomad framework," 2021, [Online], https://www.nomadproject.io.

[12] Eclipse Foundation, "Eclipse Fog05," 2021, [Online], https://projects.eclipse.org/projects/iot.fog05.

[13] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: a control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 54–59, 2010.

[14] N. Sukhija and E. Bautista, "Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 257–262.

[15] E. Betke and J. Kunkel, "Real-time I/O-monitoring of HPC applications with SIOX, elasticsearch, Grafana and FUSE," in *International Conference on High Performance Computing*. Springer, 2017, pp. 174–186.

[16] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[17] N. Makris, C. Zarafetas, A. Valantasis, T. Korakis, and L. Tassiulas, "Integrating nfv-mano with wireless services: Experiences and testbed development," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2018, pp. 1–5.

[18] N. Makris, C. Zarafetas, A. Valantasis, and T. Korakis, "Service Orchestration Over Wireless Network Slices: Testbed Setup and Integration," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 482–497, 2021.