An Infrastructure to Deliver Synchronous Remote Programming Labs

Miguel Garcia, Jose Quiroga, Francisco Ortin

University of Oviedo, Computer Science Department, c/Federico García Lorca 18, 33007, Oviedo, Spain

An Infrastructure to Deliver Synchronous Remote Programming Labs

Miguel Garcia, Jose Quiroga, and Francisco Ortin

Abstract—**With the abrupt nationwide lockdown caused by the COVID-19 pandemic, many universities suspended face-to-face activities. Some of them decided to continue their academic courses, adapting traditional approaches to online learning. An important challenge was to deliver programming labs over the Internet without important methodological changes, that might imply modifications of the learning outcomes. Most of the existing approaches to remote programming labs are based on asynchronous learning, where students work autonomously and contact the lecturers if they have any issues. The existing systems to provide synchronous programming labs are restricted to a single programming language or application type, and show significant interaction limitations. Therefore, we defined an infrastructure that allowed us to deliver synchronous programming labs over the Internet during the COVID-19 lockdown, as we used to do face-to-face. After using it for both programming labs and exams, students showed a high level of satisfaction. Compared to previous years, the use of our system produced no statistically significant difference in student's grades, pass and fail rates, or the number of students taking the lab exam. The network bandwidth, CPU, and memory resources consumed are sufficiently low to have allowed all the students to use it without any issues. Regardless of the pedagogical and methodological approach selected, our infrastructure provides the synchronous and remote delivery of programming labs, similar to the original face-to-face approach. Its features make it appropriate to deliver synchronous remote classes where strong lecturer–student interaction is required, and all the student work can be done with their computers.**

Index Terms—**Remote programming lab, synchronous online learning, computer monitoring system, Web conferencing platform, resource consumption, Veyon.**

I. Introduction

COVID-19 was originated in Wuhan (China) and spread rapidly across the globe, taking most of the world by surprise. In the absence of a vaccine, social distancing emerged as the most widely adopted strategy for its mitigation [1]. The social distancing measures caused the decision of governments to shut down schools and universities in most countries. In many places, nationwide lockdowns were imposed, suspending all the face-to-face activities of educational institutions.

With students and lectures confined at home, most Spanish universities decided to continue their academic courses. Lecturers adapted their courses, methodologies, and evaluation systems to deliver their classes online. Those changes should not modify the learning outcomes to be achieved by students, and they should be assessed accordingly [2].

At the Spanish University of Oviedo, a Programming Technology and Paradigms course is delivered as part of a Software Engineering degree [3]. In that course, the students learn object-oriented and functional paradigms, concurrent and parallel programming, and the basic meta-programming services provided by most dynamic languages [4]. The course is delivered through lectures and laboratory classes, summing 58 class hours (30 hours for programming labs and 28 for lectures) along the second semester (6 ECTS). Lectures introduce the concepts of each paradigm, and labs are mainly aimed at solving programming problems, using different language features and paradigms [3].

The online delivery of lectures was undertaken with the aid of Web conferencing platforms (we used Microsoft Teams, mostly, and BigBlueButton). The recorded classes and additional course material were uploaded to the course learning management system (LMS) (Moodle). When necessary, the University of Oviedo provided students with the necessary resources to be able to attend online lectures. In this way, we were able to adapt our face-to-face learning approach to online lecturing.

We found, however, laboratory classes much more difficult to be delivered online. In our labs, we pose different programming activities, that require strong lecturer−student interaction. Instructors need a system to not only communicate with students, but also see the code they are typing. When a student makes an important mistake, or they are not doing what they should, the lecturer could be able to see it and help them out straightaway—that is how we do it in face-to-face labs. Sometimes, it might also be beneficial to take control of the student's computer (after their explicit consent) to assist them with their work. This kind of system would provide students with robust assistance and guidance, while they are in the

This work has been partially funded by the Spanish Department of Science, Innovation, and Universities: project RTI2018-099235-B-I00. The authors have also received funds from the University of Oviedo through its support to official research groups, GR-2011-0040. (*Corresponding author: Francisco Ortin.*).

Miguel Garcia, Jose Quiroga, and Francisco Ortin are with the Computer Science Department, University of Oviedo, c/Federico Garcia Lorca 18, 33007, Oviedo, Spain (e-mail: garciarmiguel@uniovi.es; quirogajose@uniovi.es; ortin@uniovi.es). Francisco Ortin is also an adjunct lecturer of the Department of Computer Science, Cork Institute of Technology, Rossa Ave., Bishopstown, Cork, T12 P928, Ireland.

programming labs. Likewise, the ability to monitor and record students' computers in programming exams represents a valuable tool for cheating detection—a worrying factor in online teaching [5].

Most of the existing approaches to online programming labs are based on detailed specifications of programming assignments that the students undertake asynchronously [6]. If they have questions about the assignments, they mainly use email and discussion forums to ask lecturers and other students about the problems found [6]. Different tools allow students to upload their code and receive some feedback about it [7]; virtual programming labs that provide Web IDEs with features aimed at improving programming [8]; and multiple systems for the automated online evaluation of programming assignments [9]. However, such approaches do not mimic the way face-to-face programming labs are delivered in the physical computer laboratory. Therefore, the use of those approaches would represent important changes in the original methodology, most likely altering the learning outcomes if the whole course were not modified.

For all these reasons, we defined an infrastructure to deliver synchronous online programming labs in the same way it is commonly done in face-to-face sessions, regardless of the pedagogical and methodological approach used by lecturers. The infrastructure comprises a modification of a computer monitoring system; a virtual private network (VPN) configuration system to facilitate the use of our infrastructure over the Internet; a collection of scripts that makes it very easy for students to (un)install, start, and stop the system; a Web conferencing platform; and some other scripts to allow the lecturer to create the programming labs. We successfully used it for both programming labs and exams, with a high level of student satisfaction and reasonable resource consumption. Our system is particularly suitable for synchronous remote labs and classes that mimic face-to-face interaction, and all the student work could be done with their computers.

The rest of this paper is structured as follows. The next section discusses related work and the architecture of our infrastructure is depicted in Section III. Section IV describes different use cases of our system. We evaluate our infrastructure in Section V and conclusions are presented in Section VI.

II. Related Work

There are different works about remote and virtual laboratory implementations in different disciplines [10]. Most of such works are focused on the asynchronous approach, where there is not real-time student−instructor interaction through the lab sessions [10].

One synchronous remote lab platform is that implemented by Böhne, Rütters, and Wagner [11]. They developed and evaluated a Web-based lab environment that supports synchronous tele-tutorial assistance by a human tutor. Their work emphasizes the importance of a remote tutor, especially in situations where learners have problems and questions. Without the possibility to get immediate feedback, many learners may become demotivated and abort their learning. Their Web system allows the compilation and execution of Java code and communication via audio, video, and text chat. When learners want to share their screen with the tutor, they use VNC [12]. They conducted an experiment with 19 electrical engineering students. The students worked on different programing tasks, while a tutor assisted them via videoconference, text chat, and desktop sharing. The evaluation showed that synchronous tele-tutorial support has the potential to assist students effectively during the remote laboratories [13]. The main difference with the infrastructure presented in this article is that Web lab uses a constrained Java programming environment. Students can only implement embedded Java code, and it is not allowed the use of any IDE, library, compiler, or component installed in the local computer. Additionally, lectures cannot see what all the students are doing in the labs.

Jara *et al.* developed synchronous collaborative virtual and remote laboratories within a Web course [14]. They modified the Moodle LMS to offer the Web environment, and used Easy Java Simulations (EJS) for the virtual and remote laboratories with collaborative support. EJS is an open-source Java tool for the creation of discrete computer simulations [15]. All the code should be implemented as Java applets. With this system, the instructor prepares virtual remote labs and add them to the online Web course. Any student enrolled in the course can be invited to the synchronous lab. Lab attendants communicate through the online user messages feature provided by the LMS. Compared to our approach, this communication mechanism limits highly interactive labs. Moreover, the approach of Jara *et al.* is restricted to a particular programing language (Java), programming environment (EJS), and application type (applets). Popović and Naumović took the approach defined by Jara *et al.* and used EJS embedded in Moodle to deliver optional control theory labs [16]. An anonymous survey showed that 91% of students were satisfied with the virtual remote labs.

Bakonyi, Illés, and Verma used the Veyon computer monitoring system [17] for delivering non-remote programming labs [18]. They obtained some of the benefits we were looking for, but not in a remote environment. Using Veyon, the instructor can show the lab attendants the work of one student to emphasize good programming practices or solve common errors. Lecturers can also share their screen with one single student, when it should not be seen by the rest of them (e.g., a solution excerpt). If necessary, instructors may lock the students' computers to draw their attention. Unfortunately, the use of Veyon is much easier in a local area network than over Internet, since open ports, firewall exceptions, remote user authentication, student IP identification, and encrypted communication are not important issues (see Section III-B). Therefore, we present in this article an approach to overcome these issues. We will see how Veyon represents an important asset to deliver remote programming labs.

There are multiple works about virtual labs [10]. In a virtual lab, a distributed software simulates laboratory environments whereby students conduct experiments in a virtual space [10]. Prieto-Blazquez *et al.* propose the design of virtual programming labs (VPLab) [6]. One of the key components of VPLab is its technological resources. Those are technology

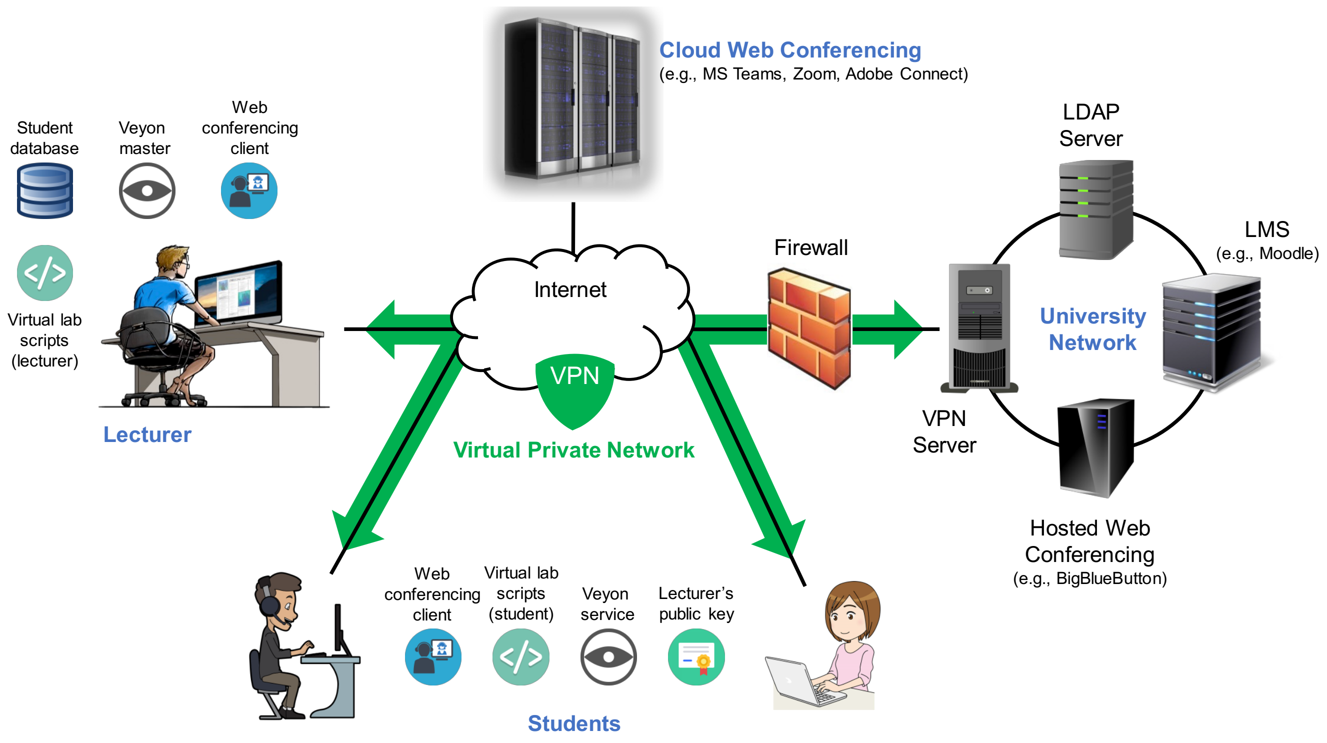


Fig. 1. Architecture of the synchronous remote programming lab infrastructure.

artifacts that can be used to simulate virtual laboratories, including a virtual communication environment (VCE), a simulator (SIM), a remote laboratory (REM), a virtual machine (VM), and an automatic assessment tool (AET). The students work autonomously and they ask instructors if they have any issue with the activities. Evaluation is performed with an automatic assessment tool that provides the test reports to the lecturer. They conducted a survey showing that the students grant more importance to human factors than to technological elements [6]. Tareq M. Alkhaldi adds pedagogical considerations to virtual labs, designing a framework to deliver a systems-level course in computing [19]. An evaluation showed that students improved their assessment scores and they obtained higher learning outcomes [19]. Besides the software limitations of these systems, they do not support the strong student−instructor interaction we were looking for.

A-VPL is a virtual programming lab Web system implemented at Athabasca University, Canada [8]. In Web-based distance education, remote virtual labs represent a challenge for both students and instructors. Fixing code errors is commonly a difficult task for beginners, and they need aid from instructors. A-VPL is a Web application where registered students can program in multiple languages, such as Java, C++, and JavaScript [20]. All the participants in a virtual lab are shown, and a chat area allows the class participants to communicate with each other. A-VPL provides neither audio nor video support, and participants' screens could not be shared.

VPL is a Virtual Programming lab plugin for the Moodle LMS [21]. VPL provides an online code editor (Java applet) that supports Ada, C, C++, C#, Fortran, Haskell, Java, Matlab, Octave, Pascal, Perl, Php, Prolog, Python, Ruby, Scheme, SQL, and VHDL. Programs can be compiled and run in a sandboxed environment executed in a *jail* Linux server [22]. VPL provides a powerful automatic grading system [23]. Instructors define how the student program is evaluated, allowing different kinds of tests. They can also define the rubric used to grade each assignment, which is automatically incorporated in the student record. VPL also provides a plagiarism test to detect programs with a high level of similarity [24]. The main inconvenience of VPL is that it does not support communication among lab attendants.

eLaboratory combines remote lab technologies and collaboration-based eLearning to improve the learning process in remote labs [25]. The eLaboratory architecture consists of four modules: remote experiment, engineering portal, telepresence, and application publishing modules. The remote experiment module allows students to access the hardware resources in the physical lab. The engineering portal module is the central Web component, which enables students and instructors to access the eLaboratory. Telepresence module comprises a video and a collaborative communications layer. The application publishing module allows students to access the laboratory software. The approach of eLaboratory is based on asynchronous remote access to laboratories. In the evaluation, the students indicated that the main limitations were the user interface and the insufficient communication support between students and teaching assistants [25]. This work was later adapted to test electric motors over the Internet [26].

WebVPL is a Web-based virtual programming lab for on-line distance learning [27]. WebVPL provides the students with access over the Internet to a collection of lab server computers hosting programming software. The design of the WebVPL system includes a user interface, an agent-based client, a mechanism to locate user-requested software

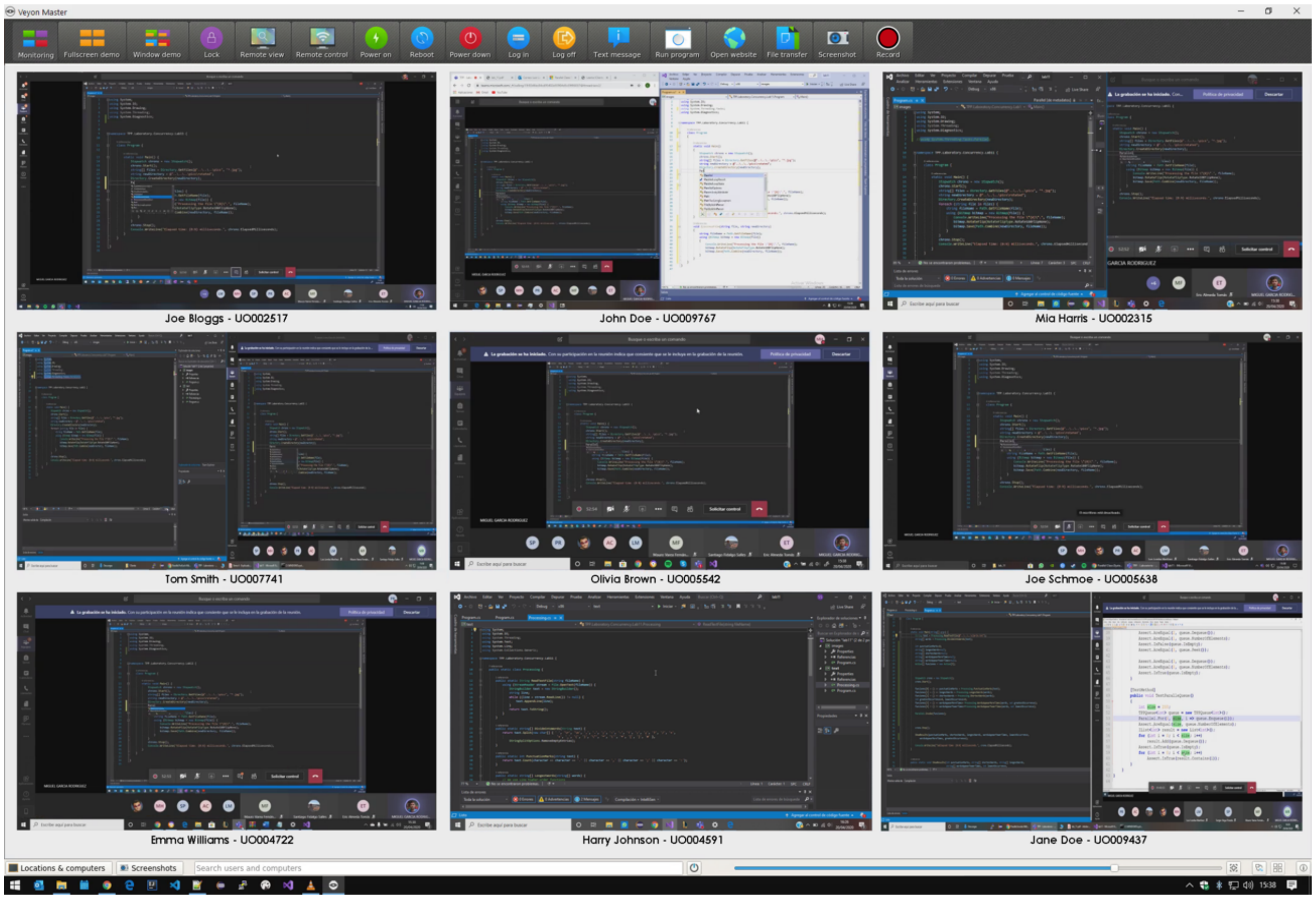


Fig. 2. Screenshot of an example remote lab with 9 students (student IDs and names were changed for the sake of anonymity).

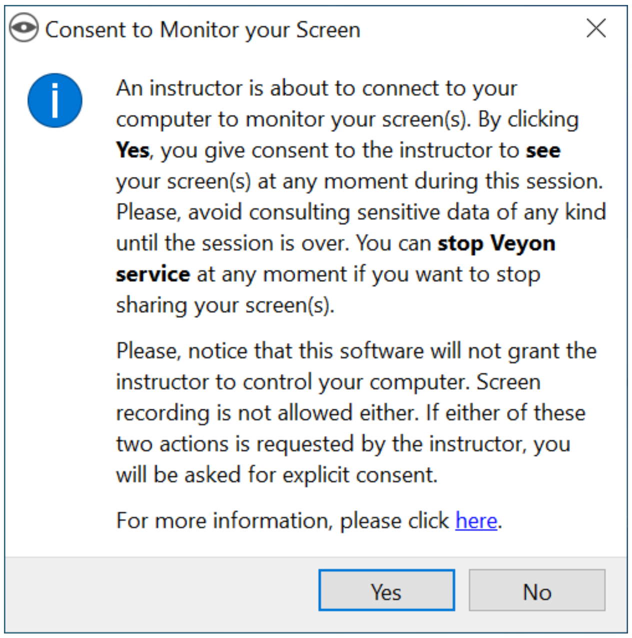
packages/services, and the interfaces to various programming software packages. Demaidi *et al.* took Moodle and WebVPL to deliver a blended-learning C programming course at An-Najah National University [28]. Students' performance was significantly higher than the previous traditional approach. Students found the system easy to use, suitable for writing code, and practical for homework submissions. Unfortunately, WebVPL provides no interaction among users.

Although not in a didactic setting, the work and technologies used in the field of distributed teamwork are significantly related to synchronous remote teaching [29]. There has been a recent increase in the use of Web conferencing tools such as Zoom, Abobe Connect, Google Meet, and Webex Meetings for remote teaching [30]. This kind of software allows instructors and students to conduct training sessions remotely, via the Internet. Teamwork communication platforms (e.g., Microsoft Teams, Slack, and Webex Teams) commonly include additional features to improve teamwork, such as persistent chat rooms or channels, third-party application integration, direct messaging, and private groups. Teamwork communication platforms have been used as both web conferencing and collaborative learning tools [31]. Virtual classroom software packages (e.g., BigBlueButton, Adobe Connect, and Blackboard Collaborate) are customizations of the previous tools for online learning. For the particular case of programming courses, source code repositories hosting services, such as GitHub and Bitbucket, have been utilized for managing student course work, and as communication and collaboration tools, in an asynchronous fashion [32].

III. Architecture

Fig. 1 shows the architecture of the remote programming lab infrastructure we designed. Lecturers and students (participants) are connected to the system through the Internet, but a VPN is created to include all the lab participants in the same private network. A VPN server, placed in the University network, accepts connections from the lab participants. The VPN server permits the authentication of students in different ways, including the use of the university LDAP server. Student monitoring is undertaken with a fork we implemented of the Veyon project [33]. Such fork provides video recording of all the students attending the remote lab and some extensions to comply with the general data protection regulation (GDPR) of the European Union [34].

Communication among students and lecturers is mainly supported by a Web conferencing system. We used both cloud-hosted Microsoft Teams and an instance of BigBlueButton deployed in servers inside our university network. We developed different scripts for both students and lecturers, providing different functionalities such as (re)starting, stopping, (un)installing the remote lab, gathering the information of all the lab attendants, and remote lab creation


Consent to Monitor your Screen

An instructor is about to connect to your computer to monitor your screen(s). By clicking **Yes**, you give consent to the instructor to **see** your screen(s) at any moment during this session. Please, avoid consulting sensitive data of any kind until the session is over. You can **stop Veyon service** at any moment if you want to stop sharing your screen(s).

Please, notice that this software will not grant the instructor to control your computer. Screen recording is not allowed either. If either of these two actions is requested by the instructor, you will be asked for explicit consent.

For more information, please click here.

Fig. 3. Window asking for explicit consent when the lecturer is about to view the student's screen.

(just for instructors).

A. *Computer Monitoring System*

An important element of the proposed infrastructure is the software used to monitor students' computers, while they are attending a remote lab or taking an exam. For this purpose, we used the Veyon open source monitoring system for Windows and Linux [35].

With Veyon, the lecturer can see all the screens of the students in the remote lab, as shown in Fig. 2. Lecturers can display one remote computer screen in full size, in a separate window. It also allows taking control of a student's computer if needed, and locking all the computers to draw students' attention to the lesson being delivered. Some other features are provided, such as transferring files, sending text messages, opening websites, and broadcasting screen contents.

Veyon consists of two components: service and master for, respectively, students and lecturers. Veyon service is a non-graphical application that runs as a server, sending monitoring information to the master. Veyon master is a graphical application used by the lecturer to monitor student's work, offering the additional features mentioned above.

We created a fork of the Veyon open source project to add some functionalities to it [33]. An important feature we demanded lab exams is the capability of recording the students' work during one session. This feature is important in our case, because the University of Oviedo requests some kind of proof to penalize cheating in exams. In fact, they recommend recording the online exams when possible. Therefore, we initially tried to record Veyon master sessions using a video recording software, but different issues appeared. First, views of labs with more than 12 students are not sufficiently detailed. Second, when the lecturer zooms the view of one student or interacts with him/her, the software does not record the rest of the students. We also considered the use of different monitors, but it turned out to be more difficult than expected for labs with more than 12 students.

Therefore, we modified the implementation of Veyon to include video recording as part of its functionality [33]. The `VeyonMaster.json` configuration file is modified to specify the parameters about how the video should be recorded. In that file, the lecturer indicates video frame size (in pixels), frames per second, and location. It also allows the storage of a sequence of screenshots instead of a video. We added a new button for video recording to the toolbar (Fig. 2). When the button is clicked, the system records one video per lab attendant.

Monitoring and recording the student's activity, and taking control of a student's computer, involve privacy issues that must be considered. According to the EU regulation 2016/679 about general data protection (GDPR), those actions require the students to provide consent explicitly [34]. Thus, we included in our modification of Veyon different functionalities to comply with GDPR [33]. The first time the lecturer is about to view the student's screen, the informative message in Fig. 3 is showed to the student. If consent is given, the lecturer is granted view (but not control or recording) privileges. Later, if the lecturer tries to control the student's computer or record his/her activity, another window appears asking for consent. If consent is not provided, that action is not performed. Moreover, as informed by the window in Fig. 3, the students are provided with a script to stop the monitoring service at any time (Section III-C).

B. *Virtual Private Network*

The use of Veyon through the Internet involves multiple issues. First, the Veyon service to be installed in the student's computer listens for incoming connections in its primary service port (its default value is 11100). To make it work, students should change their router settings to open/forward the TCP port used by the Veyon service. Although that is not actually an impediment, we had reservations about whether all the students would be able to do it successfully. Second, Veyon master (lecturer) needs to know the IP of all the Veyon services (students) included in the remote lab. However, many ISPs do not provide static IP addresses to home customers. Another drawback is that the Veyon primary service port exposes all the transmitted information with no encryption.

For all these reasons, we decided to use a VPN in order to make it easier the use of Veyon over the Internet. As shown in Fig. 1, we hosted an instance of the SoftEther open-source VPN server in the university network [36]. SoftEther provides AES 256-bit and RSA 4096-bit traffic encryption. It also implements a versatile user authentication service that may rely on the university LDAP server through RADIUS, Active Directory, and NT Domain controllers [36]. Therefore, we can assign the VPN server the tasks of student authentication, using their university credentials. Moreover, students get rid of modifying their router settings.

In this way, students only need two actions to join a remote lab: 1) connect to the VPN through any VPN client (most of them use the one included in Windows 10); and 2) run the `start` script provided as part of our infrastructure (see Section III-C) that launches the Veyon service.

C. *Remote Lab Scripts*

Our remote lab infrastructure is provided with two sets of Windows and Linux scripts, for both students and lecturers (Fig. 1). What follows is a brief description of the five scripts provided for students:

1) (`un`)`install`. The `install` script installs the Veyon service, opens the 11100 port in the firewall, includes the public certificate of the lecturer delivering the lab, and sets the authentication method as public-key file authentication.
   In order to access the student's computer, the accessing lecturer must first authenticate himself. Access without authentication is not supported in Veyon. The simplest approach is to include the lecturer's public key file in the `install` script and set the authentication mode of the Veyon service to key-file authentication. In this way, only the lab lecturer assigned to each student could monitor their work.
   The `uninstall` script performs the reverse process.
2) (`re`)`start`. One script runs the Veyon service as a background process, and the other one restarts it.
3) `stop`. This script stops the Veyon service running in the student's computer. In this way, students control exactly when they could be monitored. In the `install` script, we set the `autostart` property to false, meaning that the student must run the Veyon service explicitly (i.e., with `start`).

In the case of lecturers, the scripts to be run need information about students, which are taken from the university enterprise resource planning (ERP). That information is stored in a database (student database in Fig. 1) that includes student name and surname, their unique university ID, and the lab they belong to.

These are the four scripts we provide for lecturers:

1) (`un`)`install`. Very similar to student scripts. The main difference is that this script installs the lecturer's public and private keys, so he/she must provide these two files before installation.
2) `create_labs`. This script connects to the VPN server to know all the computer IPs connected to the VPN. For each IP, it searches in the student database for the University ID used by each student connected to the VPN. Then, it calls the `parse` script (below) to create the remote lab configuration file that is passed to the Veyon master application. The outcome is that the Veyon master shows the lecturer all the students' screens and identification data (name, surname, and ID) attending the remote lab. An example lab for 9 students is shown in Fig. 2, where student name, surname, and ID are written below each screen.
3) `parse`. This script parses all the information returned by the VPN server. For each IP, it retrieves the associated student ID and takes from the student database their name, surname, and the lab they belong to. Finally, it writes all that information in a CSV file used to configure Veyon master through its common line interface (CLI) [35].

D. *Web Conferencing Platform*

The same as with online lectures, a Web conferencing platform is used to deliver remote labs synchronously. The University of Oviedo provides both cloud-hosted Microsoft Teams and a BigBlueButton instance deployed in various servers inside the university network (we mainly used Teams). Lecturers use a Web conferencing platform to explain the work to be done in the remote lab, interact with students through audio and chat, speak to a single student without disturbing the other ones, share lab resources, allow students to upload their work, and share the lecturer's screen and whiteboard [37].

IV. Case Scenario: Programming Technology and Paradigms

As mentioned, we used the proposed remote lab infrastructure to deliver the programming labs of the Programming Technology and Paradigms course of a Software Engineering degree [4]. On March 13, the Spanish Government announced the state of alarm, which implied mandatory confinement. Next week, we delivered labs just using Teams, and started the definition of the remote lab infrastructure presented in this paper. We first used it the following week, improving it day after day. On May 2 remote labs ended, and we started using our infrastructure for programming lab exams.

There were 136 students enrolled in the course, distributed in 11 labs (12.4 students per lab). Four different lecturers acted as lab instructors. As mentioned, all the labs are aimed at solving programming problems using object-oriented and functional paradigms, and concurrent and parallel programming [4]. Students must use the C# multiparadigm programming language and Visual Studio.

What follows is a brief description of how we used the different elements of the architecture presented in Fig. 1, for some recurring scenarios.

A. *Lab Introduction*

At the beginning of each lab, lecturers explain the programming problems to be solved. They use different features of the Web conferencing platform (e.g., screen and whiteboard sharing, audio interaction, and chat participation). By taking a look at the students' screens (Veyon), the lecturer can warn students and ask them to pay attention. Instructors can even lock students' computers to draw their attention.

B. *Lab Work*

After explaining the programming lab, students work on the lab activities. Veyon is used by the lecturer to check if any student has important issues, communicating just with him/her through Teams. If needed, and after explicit consent, the lecturer takes control of the student's computer to help him/her out.

It sometimes happens that many students have the same mistake (noticed with Veyon). In that case, the lecturer uses the

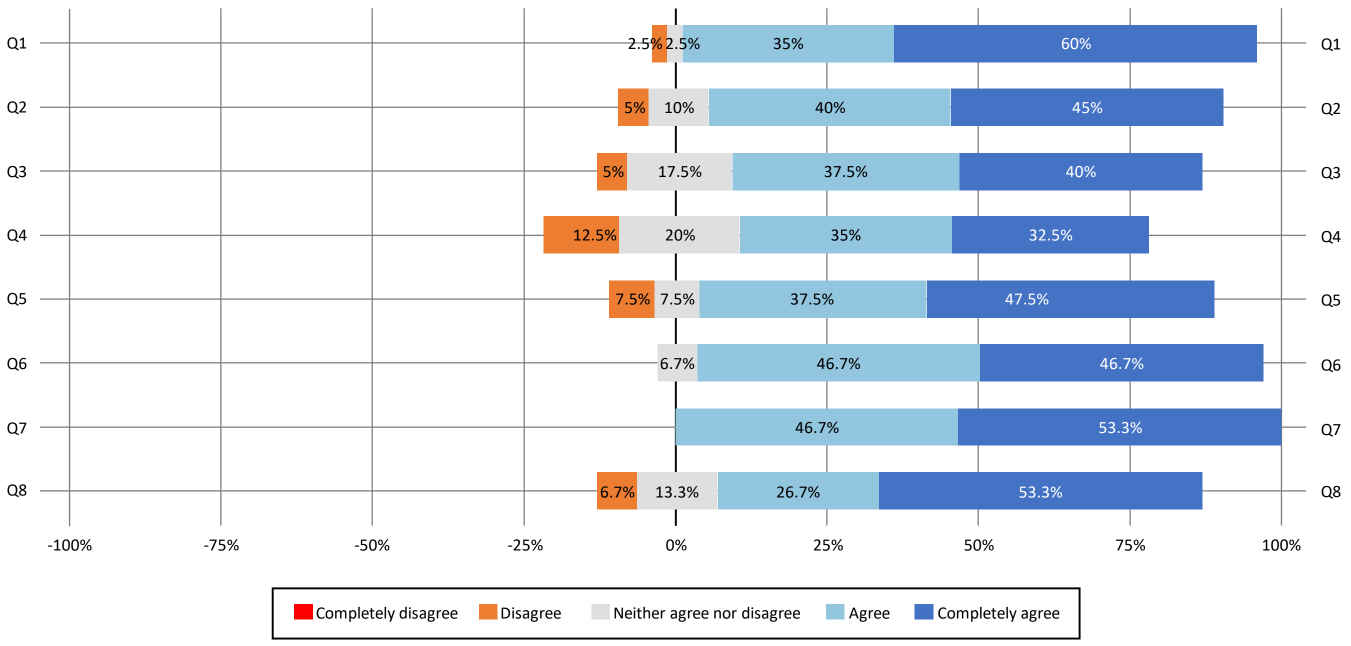


Fig. 4. Student's answers to the Likert scale questionnaire (N = 50).

TABLE I
SUMMARY OF STUDENTS' ANSWERS TO THE QUESTIONNAIRE

Question	Arithmetic mean (μ)	Standard deviation (σ)	Mode	Median
Q1	4.53	0.68	5	5
Q2	4.25	0.84	5	4
Q3	4.13	0.88	5	4
Q4	3.88	1.02	5	4
Q5	4.25	0.90	5	4
Q6	4.40	0.63	5	4
Q7	4.53	0.52	5	5
Q8	4.27	0.96	5	5
Download Mbps	89.90	87.27	100	67.45
Upload Mbps	57.72	63.38	100	35.5

N = 50; 1 = completely disagree, 2 = disagree, 3 = neither agree nor disagree, 4 = agree, 5 = completely agree.

Web conferencing platform to clarify that issue to all the lab attendants.

Throughout the programming lab, students call the lecturer to ask for help (Teams), while their screen is being monitored (Veyon).

C. *Lab Exam*

Lecturers check what the students are doing in the exam. If some student accesses one resource that is not allowed or performs any other forbidden action, he/she is warned with a popup message (Veyon) or orally (Teams). In case the lecturer realizes one student does not understand the exam, that student is contacted to clarify the exam.

D. *Lab Assessment*

While evaluating the students' work, lecturers may find some signs of plagiarism or cheating. If that is the case, the recorded videos generated by our Veyon fork can be analyzed to find some evidence. As mentioned, our university requests proof to give a failing grade for plagiarism, and encourages the use of recorded videos for online exams.

V. EVALUATION

We present an evaluation of our remote lab infrastructure, including students' opinion, resource consumption, and the analysis of various data compared to previous years.

A. *Students' Opinion*

After using the infrastructure for both programming labs and exams, we asked the students to fill in a questionnaire. In this way, the students evaluated our system by answering various questions related to the different hypotheses of our work. The anonymous questionnaire was published online with Google Forms. Out of 99 students attending the remote labs and taking the programming exam (73 males and 26 females), 50 filled in the survey.

The questionnaire consists of eight questions in a 5-point Likert scale, ranging from 1 = "completely disagree" to 5 = "completely agree". It is also asked the download and upload speed of their Internet connection—a link to a Web speed test is provided [38]. The purpose of these two last questions is to check if poor network connections could involve issues with the use of our remote lab infrastructure. These are the questions asked:

1) The system used for the remote labs is very easy to install and use.
2) The system is not intrusive. That is, it lets me work with my computer as usual.
3) I have not noticed that the system slows down my Internet connection significantly.
4) I think the remote lab system prevents students from cheating in the exams.
5) I know exactly when my computer is being monitored, so my privacy is assured at all times.

TABLE II
COMPARISON WITH PREVIOUS ACADEMIC YEARS

	Academic years	2011-2012	2012-2013	2013-2014	2014-2015	2015-2016	2016-2017	2017-2018	2018-2019	2019-2020	95% IC, 2011-12 to 2018-19
	N (enrolled)	80	81	138	133	153	188	147	140	136	(94.1, 161.2)
Labs	Took lab exam	39	49	80	93	123	153	121	108	99	(51.9, 123.5)
	Pass	29	38	54	63	61	112	95	63	71	(33.9, 84.7)
	Fail	10	11	26	30	62	41	26	45	28	(10,5, 42.9)
	Taking lab exam	48.8%	60.5%	58.0%	69.9%	80.4%	81.4%	82.3%	77.1%	72.8%	(57%, 80.4%)
	Pass rate	74.4%	77.6%	67.5%	67.7%	49.6%	73.2%	78.5%	58.3%	71.7%	(58.4%, 76.9%)
	Fail rate	25.6%	22.4%	32.5%	32.3%	50.4%	26.8%	21.5%	41.7%	28.3%	(21.1%, 39.7%)
	Average mark	6.5	6.4	6.1	6.2	4.9	6.0	6.3	5.8	6.1	(5.54, 6.47)
Final Eval.	Took any exam	48.8%	61.7%	58.0%	69.9%	80.4%	81.4%	82.3%	84.3%	72.8%	(57.3%, 82%)
	Pass rate	87.2%	76.0%	47.5%	72.0%	53.7%	77.1%	91.7%	55.1%	75.8%	(53.3%, 83.4%)
	Fail rate	12.8%	24.0%	52.5%	28.0%	46.3%	22.9%	8.3%	44.9%	24.2%	(10.4%, 40.5%)
	Average mark	6.5	6.1	5.1	6.2	5.1	6.1	6.3	5.9	6.0	(5.41, 6.38)

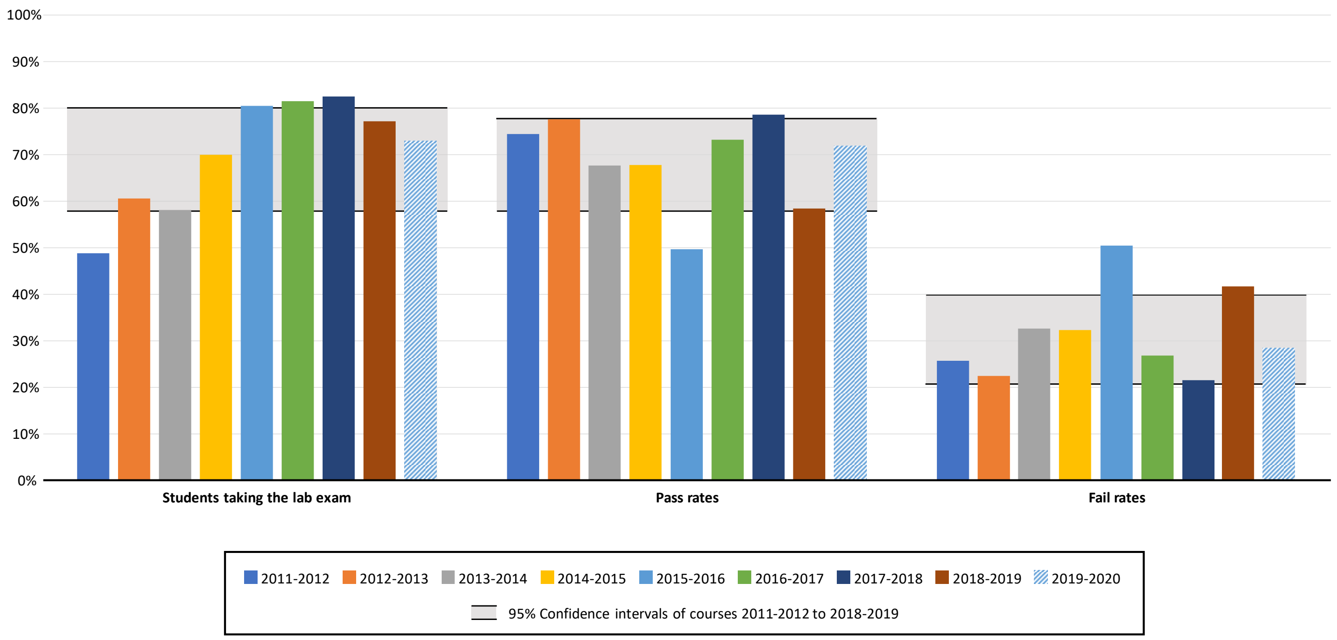


Fig. 5. Students taking the lab exam, and pass and fail rates for the last academic years.

6) Compared to previous face-to-face labs, the content of online lessons and their learning outcomes have not been reduced in remote labs.
7) In my opinion, the infrastructure used by the lecturers to deliver remote labs has been appropriate to achieve the objectives of each session.
8) In the remote labs, lecturers have successfully used different online technologies to help me out with the lab activities.

Fig. 4 and Table I show the students' answers. The Cronbach's α coefficient [39] obtained is $\alpha = 0.8543$, showing good reliability of the questionnaire [40]. For all the questions, the most common response (mode) is "completely agree". All the questions but one (Q4) have an average value greater than 4, so students "agree" with the sentences stated in the questionnaire. Two questions, Q1 and Q7, are closer to "completely agree": the first one, about how easy it is to install and use; for Q7, the students strongly agree that the infrastructure is appropriate to achieve the objectives of each session. The question about the use of remote lab to prevent cheating (Q4) was the one with the lowest average result (3.88), but on average they "agree" with it (4 is the closest value in the Likert scale).

Regarding the Internet connection, the median bit rate is 67.45/35.5 Mbps and the average value is 89.9/57.7 Mbps. There is a weak correlation between the question about noticing Internet speed slowdown (Q3) and download speed (Spearman's correlation coefficient $r_s = 0.374$, $p < 0.01$), and between Q3 and upload speed ($r_s = 0.288$, $p < 0.01$) [41]. The only students answering "disagree" in Q3 have Internet connections equal or slower than 3/2 Mbps.

B. Comparison with Previous Years

We also analyze students' marks in the last nine years, to see if there is any significant difference with the last academic course. Table II presents the number of students enrolled in the course, how many took the lab exam, pass and fail rates, and average marks. Both Table II and Fig. 5 show the 95% confidence intervals (*Student's t* distribution) from 2011-2012 to 2018-2019, not including the last academic year when we used the remote lab infrastructure (2019-2020). We can see how all the values of the last academic course are within the 95%

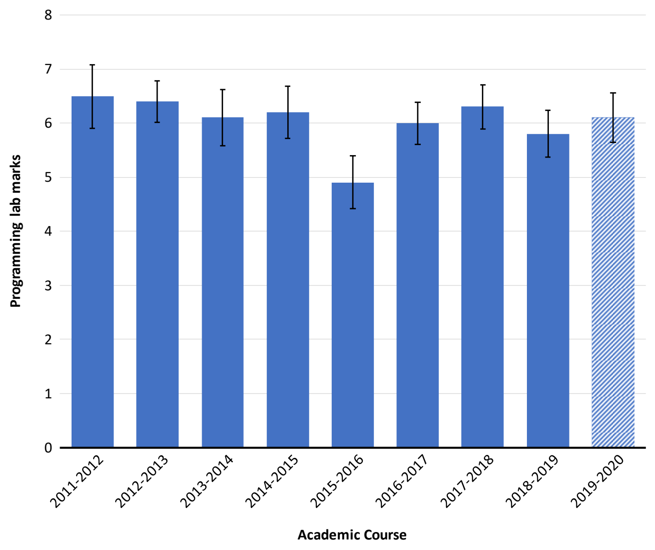


Fig. 6. Student's lab marks for the last academic years.

confidence intervals, so the remote lab approach does not cause any significant difference with previous years [42]. In the last course, the percentage of students taking the lab exam (72.8%) and the pass rate (71.7%) are greater than the average values for the previous eight years (69.8% and 68.3%, respectively).

Fig. 6 presents the 95% confidence intervals of student's marks for the programming labs, for each year. To find statistically significant differences, we perform an independent sample *t* test to each previous academic year and the last one—lab marks are normally distributed for all years (Shapiro−Wilk $p > 0.8$). There are significant differences only for the 2015-2016 academic course ($\alpha = 0.05$, $p > 0.05$), which also has significant differences with all the remaining years. The average lab mark for the last course is 6.1, while the average value for the rest of the academic courses is 6.0. Therefore, our infrastructure shows no significant influence on students' marks for programming labs.

We also analyze whether the use of our infrastructure to deliver remote labs influences the final evaluation of students (the final grade for this programming course is 70% lab mark and 30% theory exam). The last four rows in Table II show data regarding the final evaluation, including the number of students taking any exam (lab or theory), pass and fail rates, and marks of the final evaluation. As in labs, the values of the last year are within the 95% confidence intervals of the previous eight academic courses. Thus, our remote programming lab approach has not caused significant differences in final grades. Indeed, the rate of students taking any exam, the average mark, and the pass rate of the last year are slightly higher than the average values of the previous courses.

C. *Resource Consumption*

We measure memory, CPU, and network consumption of our remote lab infrastructure.

1) *Methodology*

A programming lab with 15 students and one instructor was created to measure the resources consumed by the different elements of the architecture detailed in Fig. 1. All the attendants used a full HD screen resolution (1920x1080 pixels), audio was enabled, and no webcams were used. The lecturer shares his/her screen and the students ask questions with their microphones. We measured two configurations, with two different Web conferencing platforms: Microsoft Teams and BigBlueButton.

The computer used by the lecturer was a 3.6 GHz Intel Core i7 7820X (8C/16T) system with 16 GB of DDR4 3200 MHz RAM, running Windows 10 pro 1909. Students used different computers and laptops, all of them running Windows 10.

We developed a portable .NET core 3.0 application in C# to measure the resources consumed by our infrastructure—its source code is available for download at [43]. For network consumption, we used the TraceEvent library developed by Microsoft [44], which allows tracing and collecting many data about processes. Memory and CPU usage are measured with the `Process` class in the `System.Diagnostics` namespace of the .NET platform, which provides interaction with local and remote processes, event logs, and performance counters.

CPU consumption was measured with the following equation:

$$total\ processor\ time / clock\ time / number\ of\ cores \quad (1)$$

Processor time is the amount of time that the microprocessor actually spends working on a task (it is the sum of user and privileged processor time). Clock time (wall time) is the elapsed execution time for the process, from its start to its end. The division of these two values gives us the percentage of CPU used by the process. The previous expression is divided by the number of cores (`ProcessorCount` property of the `Environment` class), because `TotalProcessorTime` represents the sum of processor times for all the cores.

For memory consumption, we used the maximum size of working set memory employed by a process since it was started (the `PeakWorkingSet` property). The working set of a process is the set of memory pages currently visible to the process in physical RAM memory [45]. Those pages are resident and available for an application to be used without triggering a page fault. The working set includes both shared and private data. The shared data comprises the pages that contain all the instructions that the process executes, including those from the process modules and the system libraries.

Event tracing for Windows (ETW) is a kernel-level tracing facility to log kernel and application-defined events, including those produced by network communication [46]. TraceEvent is a library to make it easy the use of ETW [44]. This library provides the `TcpIpSend` and `TcpIpRecv` delegates of the `KernelTraceEventParser` class that are called when a TCP/IP package is sent and received by a process. We used such functionality to measure the network data sent and received by the different processes comprising our remote lab infrastructure.

2) *Results*

The first six rows in Table III show the resources consumed by each element of the infrastructure. For memory and CPU consumption, we can see how Veyon (both service and master) consume significantly fewer resources than the two Web conferencing platforms (Teams and BigBlueButton). Microsoft Teams consume more CPU, memory, and network bandwidth than BigBlueButton. Differences are particularly significant for

TABLE III
RESOURCE CONSUMPTION OF THE DIFFERENT ELEMENTS OF THE INFRASTRUCTURE AND DIFFERENT CONFIGURATIONS

		CPU usage (%)	Memory (MB)	Network download (Kbps)	Network upload (Kbps)
Elements of the architecture	Veyon service (student)	0.31%	98	46.3	93.2
	Veyon master (15 students)	0.22%	202	1290.3	5.9
	Teams student	2.26%	1375	893.6	89.6
	Teams lecturer (15 students)	3.29%	1462	79.3	664.2
	BBB student	2.25%	686	458.4	67.8
	BBB lecturer (15 students)	3.13%	761	70.3	507.8
Configurations	Student with Teams	2.57%	1473	939.9	182.8
	Student with BBB	2.56%	784	504.7	161.0
	Lecturer with Teams (15 students)	3.51%	1664	1369.5	670.1
	Lecturer with BBB (15 students)	3.35%	963	1360.5	513.7

"BBB" stands for BigBlueButton.

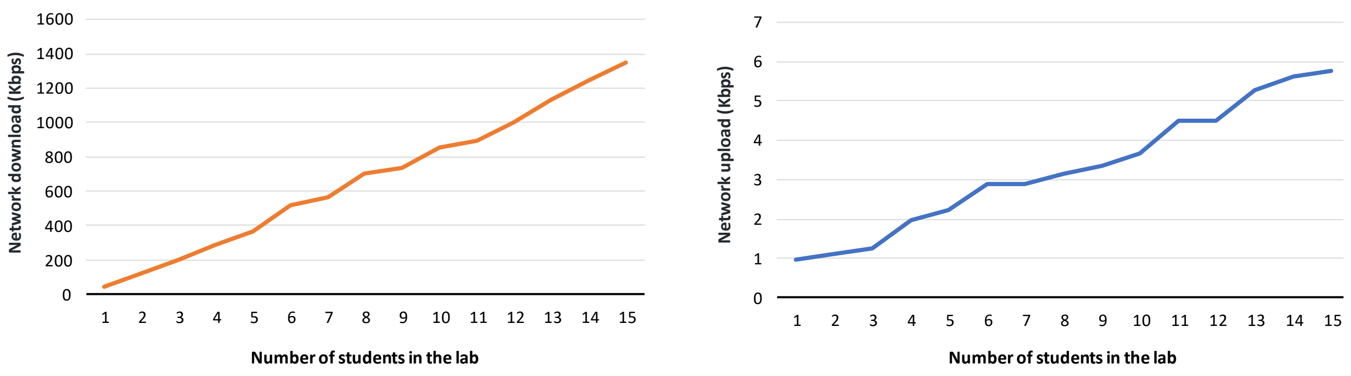


Fig. 7. Network download and upload traffic generated by Veyon master, for an increasing number of students (all of them with full HD screen resolution).

memory (96.2% higher) and student download traffic (94.9% more).

The last four rows in Table III show resource consumptions of the two configurations, for both students and lecturers (15 students per lab). CPU usage is low, ranging from 2.56% (student) to 3.51% (lecturer). The required memory resources go from 0.78 GB (student with BigBlueButton) to 1.66 GB (lecturer with Teams). Regarding network connection, the maximum bandwidth required for students is 0.94/0.18 Mbps (0.5/0.16 Mbps if BigBlueButton is used). Lecturers require 1.37/0.67 Mbps for remote labs with 15 students using full HD resolution.

3) *Scalability*

We measure how values in Table III vary depending on the number of students per class. Memory, CPU, and network traffic of both Web conferencing platforms remain constant. This is because audio and video are broadcasted from the lecturer's computer to all the students at the same time[1]. Although Veyon service for students remains constant, Veyon master needs more bandwidth as the number of students per lab increases.

Fig. 7 shows the download and upload bandwidth per student consumed by Veyon Master. For 15 students, the lecturer requires a 1.3/0.006 Mbps Internet connection. To analyze the scalability of Veyon master, we built two regression models that estimate the download and upload bandwidths per student. As shown in Fig. 7, both bandwidths show a linear increase in the number of students in the lab (R^2 coefficient of determination for download and upload regression models are, respectively, 0.981 and 0.944). The download and upload bandwidth increase per student (i.e., the slope of both models) are, respectively, 90.8 and 0.37 Kbps. Thus, for a remote lab with 100 students, the instructor requires a 9.08/0.37 Mbps Internet connection.

The use of a VPN makes all the network traffic pass through the VPN server. This could represent scalability issues if multiple classes are delivered at the same time over the same VPN. Under these circumstances, it is advisable to use the clustering capability provided by many VPN servers, including SoftEther [36]. Load balancing is performed across multiple VPN servers, decreasing the load of each server, and increasing the overall throughput of the system.

VI. CONCLUSION

The infrastructure defined allows the synchronous delivery of programming labs over the Internet with easy installation and configuration, and affordable resource consumption. We used it during the COVID-19 lockdown for programming labs and exams, and it allowed us to keep the face-to-face method we were utilizing before the abrupt confinement. Our technological proposal is focused on synchronous online labs, regardless of the pedagogical and methodological approach selected by lecturers. Students can use any programming language, IDE, and tool installed on their computers.

Students showed high satisfaction with the infrastructure. They agree it is easy to install and use, is not intrusive, causes

[1] As mentioned, students use no webcams in the configurations measured.

no significant slowdown in their Internet connection, ensures their privacy, and prevents them from cheating in exams. In their opinion, the online approach does not reduce the contents and learning outcomes of the labs, and the technologies used are appropriate to achieve the lab objectives and to help them out with the lab activities. The use of the remote infrastructure caused no significant difference in students' grades, the pass and fail rates, or the number of students taking the lab exam. The resource requirements of the platform allowed all the students to attend the labs, using different computers and bandwidths.

Our system can be used in other scenarios different from programming labs. Its features make it suitable for any synchronous remote lab and class where all the student work can be done with their computers. It is particularly appropriate in those scenarios where strong lecturer–student interaction is required, and it is beneficial to monitor students' work. Examples are other computer science courses (software design, databases, or artificial intelligence), engineering courses (electronics, electricity [47], or process control [11]), and labs where simulations are performed with software tools (biology [48], chemistry [49], or physics [50]). Indeed, our system is currently been used in other two courses (programming languages design [51] and operating systems) that are been delivered online because of the pandemic. Moreover, it has also been installed in all the laboratories of the School of Computer Science Engineering at the University of Oviedo to keep the social distance between lecturers and students.

We plan to work in the creation of different predictive models to identify undesired student behavior, such as lack of attention and cheating [52]. Since we modified Veyon to record student's activity [33], we could deploy such models in our system to warn lectures as they are delivering the labs or the exam is being done. We also plan to include in our programming labs the output of other research works aimed at measuring the quality of source code [53] and detecting bad programming practices [54], so that students could learn how to improve their code autonomously.

All the infrastructure presented in this article is available for download [43]. We include the source code of all the Windows and Linux scripts (Section III-C), the Veyon fork we implemented to record student sessions (Section III-A), and the .NET core application developed to measure CPU, memory, and network bandwidth consumption (Section V-C).

REFERENCES

[1] R. Singh and R. Adhikari, "Age-structured impact of social distancing on the COVID-19 epidemic in India," 2020, *arXiv:2003.12055*.

[2] E. Nordmann, C. Horlin, J. Hutchison, J. A. Murray, L. Robson, M. Seery, and J. MacKay, "10 simple rules for supporting a temporary online pivot in higher education," *PLOS Comput. Biol.*, vol. 16, no. 10, pp. 1–18, Oct. 2020, doi: 10.1371/journal.pcbi.1008242.

[3] F. Ortin, J. M. Redondo, and J. Quiroga, "Design and evaluation of an alternative programming paradigms course", *Telematics Inform.*, vol. 34, no. 6, pp. 813–823, Sep. 2017, doi: 10.1016/j.tele.2016.09.014.

[4] F. Ortin, J. M. Redondo, and J. Quiroga, "Design of a programming paradigms course using one single programming language", in *Proc. World Conf. Inf. Syst. Technol.*, Recife, Brazil, 2016, pp. 179–188, doi: 10.1007/978-3-319-31307-8_18.

[5] M. Asadullah and Shibli Nisar, "An automated technique for cheating detection," in *Proc. Int. Conf. Innov. Comput. Technol.*, Dublin, Ireland, 2016, pp. 251–255, doi: 10.1109/INTECH.2016.7845069.

[6] J. Prieto-Blázquez, J. Herrera-Joancomartí, and A. E. Guerrero-Roldán, "A virtual laboratory structure for developing programming labs," *Int. J. Emerg. Technol. Learn.*, vol. 4, pp. 47–52, 2009, doi: 10.3991/ijet.v4s1.789.

[7] M. Cardoso, A. V. de Castro, and A. Rocha, "Integration of virtual programming lab in a process of teaching programming EduScrum based," in *Proc. Iberian Conf. Inf. Syst. Technol.*, Caceres, Spain, 2018, pp. 1–6, doi: 10.23919/CISTI.2018.8399261.

[8] H. Wang and D. Philips, "Implement virtual programming lab with cloud computing for Web-based distance education," in *Cloud Computing for Teaching and Learning: Strategies for Design and Implementation*. Hershey, PA, USA: IGI Global, 2012, pp. 95–110, doi: 10.4018/978-1-4666-0957-0.ch007.

[9] K. M. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," *Comput. Sci. Educ.*, vol. 15, no. 2, pp. 83–102, Feb. 2007, doi: 10.1080/08993400500150747.

[10] T. Alkhaldi, I. Pranata, and R. I. Athauda, "A review of contemporary virtual and remote laboratory implementations: Observations and findings," *J. Comput. Educl*, vol. 3, pp. 329–351, Jun. 2016, doi: 10.1007/s40692-016-0068-z.

[11] A. Böhne, K. Rütters, and B. Wagner, "Evaluation of tele-tutorial support in a remote programming laboratory," Learn. Lab Lower Saxony (L3S), Hanover, Germany, 2004.

[12] "VNC, secure and reliable screen sharing," 2021. [Online]. Available: https://www.realvnc.com

[13] A. Böhne, N. Faltin, and B. Wagner, "Synchronous tele-tutorial support in a remote laboratory for process control," in *Proc. World Innov. Eng. Educ. Res.*, Valencia, Spain, 2004, pp. 317–329.

[14] C. A. Jara, R. Heradio, L. Torre, J. Sanchez, S. Dormido, F. Torres, and F. A. Candelas, "Synchronous collaboration with virtual and remote labs in Moodle," in *Proc. Symp. Adv. Control Educ.*, Novgorod, Russia, 2012, pp. 270–275, doi: 10.3182/20120619-3-RU-2024.00030.

[15] "EJS, Easy Java/JavaScript Simulations," 2021. [Online]. Available: https://fem.um.es/Ejs

[16] N. Popović and M. B. Naumović, "Virtual laboratory and learning management system in optimal control theory education," *Int. J. of Elect. Eng. Educ.*, vol. 53, no. 4, pp. 357–370 , Apr. 2016, doi: 10.1177/0020720916639321.

[17] T. Junghans, "Veyon – cross-platform computer monitoring and classroom management," 2021. [Online]. Available: https://veyon.io

[18] V. Bakonyi, Z. Illés, and C. Verma, "Towards the real-time analysis of talks," in *Proc. Int. Conf. Comput., Automat. Knowl. Manage.*, Dubai, United Arab Emirates, 2020, pp. 322–327, doi: 10.1109/ICCAKM46823.2020.9051507.

[19] T.M. Alkhaldi, "Design and evaluation of a technological-enhanced lab environment for a systems and network administration course," Ph.D. dissertation, Univ. Newcastle, Newcastle , UK, 2019.

[20] "Virtual programming lab,", 2021. [Online]. Available. http://vpl.athabascau.ca

[21] J. C. Rodríguez-del-Pino, E. Rubio-Royo, and Z. H. Figueroa, "A virtual programming lab for Moodle with automatic assessment and anti-plagiarism features," in *Proc. of the Int. Conf. on e-Learn., e-Bus., Enterprise Inf. Syst., e-Government*, Las Vegas, USA, 2012, pp. 1–6.

[22] J.C. Rodríguez-del-Pino, "VPL jail system's documentation," 2021. [Online]. Available: https://vpl.dis.ulpgc.es/documentation/vpl-jail-system-2.6.0

[23] D. Thiébaut, "Automatic evaluation of computer programs using Moodle's virtual programming lab (VPL) plug-in," *J. Comput. Sci. Colleges*, vol. 30, no. 6, pp. 141–151, Jun. 2015.

[24] J.C., Rodríguez-del-Pino, "Integridad académica en la docencia universitaria actual con énfasis en el plagio del código fuente: modelo, propuesta de intervención y herramientas,", Ph.D. dissertation, Univ. Palmas Gran Canaria, Spain, 2016.

[25] M. G. Helander and R. Emami, "Engineering eLaboratories: Integration of remote access and eCollaboration," *Int. J. Eng. Educ.*, vol. 24, no 3, pp. 466–479, 2008.

[26] M. A. Tedesco and R. Emami, "A modular and turn-key remote-access hardware-in-the-loop platform for testing electric motors," *J. Adv. Mech. Des., Syst., Manuf.*, vol. 8, no.1, pp. 1–8, Mar. 2014, doi: 10.1299/jamdsm.2014jamdsm0008.

[27] J. Cao, A. Chan, W. Cao, and C. Yeung, "Virtual programming lab for online distance learning," in *Proc. Int. Conf. on Adv. Web-based Learn.*, 2002, pp. 216–227, doi: 10.1007/3-540-45689-9_18.

[28] M. N. Demaidi, M. Qamhieh, and A. Afeefi, "Applying blended learning in programming courses,", *IEEE Access*, vol. 7, pp. 156824–156833, Oct. 2019, doi: 10.1109/access.2019.2949927.
[29] "UNITWIN/UNESCO Chair Holders Institutional Responses to COVID-19," 2021. [Online]. Available: https://apa.sdg4education2030.org/sites/apa.sdg4education2030.org/files/2020-05/HED%20UNITWIN%20_COVID19_SURVEY%20REPORT%20 02.04.20.pdf
[30] S. Cornelius, "Facilitating in a demanding environment: Experiences of teaching in virtual classrooms using web conferencing," *Brit. J. Educ. Technol.*, vol. 45, no. 2, pp. 260–271, Mar. 2014, doi: /10.1111/bjet.12016.
[31] X. Zhang, Y. Meng, P. Ordóñez de Pablos, and Y. Sunc, "Learning analytics in collaborative learning supported by Slack: From the perspective of engagement," *Comput. Human Behav.*, vol. 92, Mar. 2019, pp. 625–633, doi: 10.1016/j.chb.2017.08.012.
[32] R. Glassey, "Adopting Git/Github within teaching: A survey of tool support," in *Proc. Conf. Global Comput. Educ.*, Chengdu, China, 2019, pp. 143–149, doi: 3300115.3309518.
[33] M. Garcia, J. Quiroga, and F. Ortin, "Veyon Fork to comply with the EU general data protection regulation and record remote lab sessions," 2021. [Online]. Available: https://github.com/ComputationalReflection/veyon
[34] Official journal of the European Union, (2016, Apr. 27). *Regulation (EU) 2016/679 of the European Parliament and of the Council, on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (general data protection regulation)*. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679
[35] T. Junghans, "Veyon documentation, release 4.3.5," 2021. [Online]. Available: https://github.com/veyon/docs/releases/download/v4.3.5/veyon-admin-manual-en_4.3.5.pdf
[36] D. Nobori, T. Sugiyama, G. Hatakeyama, and C. Smith, "SoftEther VPN project, " 2021. [Online]. Available: https://www.softether.org
[37] M. Bower, "Synchronous collaboration competencies in Web-conferencing environments – their impact on the learning process," *Distance Educ.*, vol. 32, no.1, pp. 63–83, May 2011, doi: 10.1080/01587919.2011.565502.
[38] "OOKLA, the global broadband speed test," 2021. [Online]. Available: https://www.speedtest.net
[39] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, pp. 297–334, Sep. 1951, doi: 10.1007/BF02310555.
[40] J. R. Warmbrod, "Reporting and interpreting scores derived from Likert-type scales," *J. Agricultural Educ.*, vol. 55, no. 5, pp. 30–47, 2014, doi: 10.5032/jae.2014.05030.
[41] M. M. Mukaka, "Statistics corner: A guide to appropriate use of correlation coefficient in medical research," *Malawi Med. J*, vol. 24, no. 3, pp. 69–71, Sep. 2012.
[42] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous Java performance evaluation," in *Proc. Annu. Conf. Object-Oriented Program. Syst. Appl.*, New York, USA, 2007, pp. 57–76, doi: 10.1145/1297105.1297033.
[43] M. Garcia, J. Quiroga, and F. Ortin, "An infrastructure to deliver synchronous remote programming labs (support material website)," 2021. [Online]. Available: http://www.reflection.uniovi.es/download/2020/tlt
[44] "Microsoft TraceEvent library," 2021. [Online]. Available: https://github.com/Microsoft/perfview/blob/master/documentation/TraceEvent/TraceEventLibrary.md
[45] J. M. Redondo and F. Ortin, "Efficient support of dynamic inheritance for class- and prototype-based languages," *J. Syst. Softw.*, vol. 86, no. 2, pp. 278–301, Feb. 2013, doi: 10.1016/j.jss.2012.08.016.
[46] "Microsof event tracing for Windows," 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/etw/event-tracing-portal
[47] M. A. Marques, M. C. Viegas, M. C. Costa-Lobo, A. V. Fidalgo, G. R. Alves, J. S. Rocha, and I. Gustavsson, "How remote labs impact on course outcomes: Various practices using VISIR," *IEEE Trans. Educ.*, vol. 57, no. 3, pp. 151–159, Aug. 2014, doi: 10.1109/TE.2013.2284156.
[48] Z. Hossain, X. Jin, E. W. Bumbacher, A. M. Chung, S. Koo, J. D. Shapiro, C. Y. Truong, S. Choi, N. D. Orloff, P. Blikstein, and I. H. Riedel-Kruse, "Interactive cloud experimentation for biology: An online education case study," in *Proc. Annu. Conf. Human Factors in Comput. Sys.*, Seoul, Korea, 2015, pp. 3681–3690, doi: 10.1145/2702123.2702354.
[49] B. F. Woodfield, H. R. Catlin, G. L. Waddoups, M. S. Moore, R. Swan, R. Allen, and G. Bodily, "The virtual ChemLab project: A realistic and sophisticated simulation of inorganic qualitative analysis," *J. Chem. Educ.*, vol. 81, no. 11, pp. 1672–1678, Nov. 2004, doi: 10.1021/ed081p1672.
[50] Y. Ding and H. Fang, "Using a simulation laboratory to improve physics learning: A case exploratory learning of diffraction grating," in *Int. Workshop Educ. Technol. Comput. Sci.*, Wuhan, China, 2009, pp 3–6, doi: 10.1109/ETCS.2009.523.
[51] F. Ortin, D. Zapico, and J. M. Cueva, "Design patterns for teaching type checking in a compiler construction course," *IEEE Tran. Educ.*, vol. 50, no. 3, pp. 273–283, Aug. 2007, doi: 10.1016/j.chb.2017.08.012.
[52] A. Arinaldi and M. I. Fanany, "Cheating video description based on sequences of gestures," in *Proc. Int. Conf. Inf. Commun. Technol.*, Malacca City, Malaysia, 2017, pp. 1–6, doi: 10.1109/ICoICT.2017.8074679.
[53] F. Ortin, O. Rodriguez-Prieto, N. Pascual, and M. Garcia, "Heterogeneous tree structure classification to label Java programmers according to their expertise level," *Future Gener. Comput. Syst.*, vol. 105, pp. 380–394, Apr. 2020, doi: 10.1016/j.future.2019.12.016.
[54] O. Rodriguez-Prieto, A. Mycroft, and F. Ortin, "An efficient and scalable platform for Java source code analysis using overlaid graph representations," *IEEE Access*, vol. 8, pp. 72239–72260, Dec. 2020, doi: 10.1109/access.2020.2987631.

Miguel Garcia was born in La Caridad, El Franco, Asturias, Spain in 1979. He received his B.S. degree in computer science in 2005. In 2008 he was awarded an M.S. in Web engineering and, in 2010, an M.S. by research in computer science. In 2013 he presented his Ph.D. dissertation at the University of Oviedo, Spain.

He worked in the CTIC foundation from 2006 to 2009. Since 2009, he has been an Assistant Professor with the Computer Science Department, University of Oviedo. His research interests include big data, software development, machine learning, and programming languages.

Jose Quiroga was born in Oviedo, Asturias, Spain in 1982. He received his B.S. degree in computer science in 2004. In 2009 he was awarded an M.S. in computer engineering. In 2016 he presented his Ph.D. dissertation at the University of Oviedo, Spain.

He worked in the CTIC foundation from 2007 to 2012. Since 2012, he has been an Assistant Professor with the Computer Science Department, University of Oviedo. His research interests include machine learning, distributed multimedia systems, dynamic languages, and big code.

Francisco Ortin was born in Oviedo, Asturias, Spain in 1973. He received his B.S. degree in computer science in 1994. In 1996 he was awarded an M.S. in computer engineering. In 2002, he was awarded his Ph.D. degree at the University of Oviedo, Spain.

From 1998 to 2002, he was an Assistant Professor with the Computer Science Department, University of Oviedo. In 2002, he became an Associate Professor, and from 2018 he works as a Full Professor. Since 2016, he is also an Adjunct Lecturer at the Computers Science Department of the Cork Institute of Technology, Muster Technological University, Ireland. His research interests include big code, dynamic languages, type systems, aspect-oriented programming, and runtime adaptable applications.