

DVCL: A Distributed Virtual Computer Lab for Security and Network Education

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Open Universiteit
op gezag van de rector magnificus
prof. mr. A. Oskamp
ten overstaan van een door het
College voor promoties ingestelde commissie
in het openbaar te verdedigen

op vrijdag 22 juni 2018 te Heerlen
om 13.30 uur precies

door

Jens Haag
geboren op 9 augustus 1976 te Düsseldorf

Promotor

Prof. dr. M.C.J.D. van Eekelen

Open Universiteit
Radboud Universiteit

Copromotors

Prof. dr. S. Karsch

TH Köln

Dr. ir. H.P.E. Vranken

Open Universiteit

Leden beoordelingscommissie

Prof. Di Battista

Roma Tre University

Prof. dr. J. Keller

FernUniversität in Hagen

Prof. dr. H. L. Stahl

TH Köln

Prof. dr. J.T. Jeuring

Open Universiteit

Universiteit Utrecht

Bibliografische Information der *Deutschen Nationalbibliothek*

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <<https://dnb.de>> abrufbar.

ISBN: 978-3-86387-892-4

Zugel.: Open Universiteit Nederland, Univ., Diss., 2018

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Die Wiedergabe von Gebrauchsnamen, Warenbezeichnungen, usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zur Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen.

This document is protected by copyright law.

No part of this document may be reproduced in any form by any means without prior written authorization of the publisher.

Coverbild © Antonioguillém- Fotolia.com

Alle Rechte vorbehalten | all rights reserved

© Mensch und Buch Verlag 2018

Choriner Str. 85 - 10119 Berlin
verlag@menschundbuch.de
www.menschundbuch.de

Contents

1	Introduction	1
1.1	Computer Labs in Education	3
1.2	Technical Background of this Thesis	8
1.2.1	User Mode Linux (UML) and Netkit	8
1.2.2	Virtual Computer Security Lab (VCSL)	12
1.3	Organization of this Thesis	14
1.3.1	Involved People and Environments	14
1.3.2	Contribution and Organization of this Thesis	16
I	Design Issues of DVCL	19
2	Distributed Virtual Computer Lab	21
2.1	Core DVCL Architecture	22
2.2	DVCL Example Setups	27
2.3	Conclusion	32
3	Adding a Central Authority	35
3.1	Core CA Architecture	36
3.2	CA Example Setup	38
3.3	Discussion on Scalability	41
3.4	Conclusion	41
4	Applicability Enhancements	45
4.1	Security	46
4.2	Graphical User Interface	52

4.3	Conclusion	59
II	Educational Aspects of DVCL	61
5	Course Evaluation	63
5.1	Learning Situation and Environment	64
5.2	Networking Assignment Example	65
5.3	Evaluation	66
5.4	Conclusion	73
6	Electronic Exercise Assistant	77
6.1	A Typical Exercise Example	78
6.2	Performance of a Human Course Advisor	79
6.3	Technical Feasibility	81
6.4	Educational Feasibility	88
6.5	The Electronic Exercise Assistant	95
6.6	Example for Educational Feasibility	98
6.7	Conclusion	102
7	Educational Enhancements	105
7.1	Classroom Settings	106
7.2	Virtual Classroom Prototype	111
7.3	Conclusion	114
	Summary	117
	Bibliography	121
	Curriculum Vitae	133

Chapter 1

Introduction

Knowledge distribution and knowledge acquisition are essentials for educational establishments. Usually they are built on two components: *Theory* and *hands-on experience*¹. We find this in many academic disciplines, one discipline is the field of information technology (IT) resp. computer science. Here, the theory is traditionally taught in lessons or by textbooks. The knowledge, that people learn, is often illustrated, deepened and anchored by carrying out practical (hands-on) exercises [Yang and Reddington, 2014, Crowley, 2004, Peltsverger and Zhang, 2014, Catuogno and De Santis, 2008, Yuan, 2017, Yuan et al., 2011, Xu et al., 2012, Sarkar, 2006].

While the content of a discipline is usually fixed in the curriculum, a challenge is how to organize the hands-on experience. Especially in the field of communication technology and IT security, which are sub-disciplines of computer science, special requirements occur with respect to the hands-on learning environment, e.g. the isolated real-world character and manageability.

Usually, an exercise is derived from and also targeted at a real-world setting, e.g. setting up and configuring hosts and networks or attacking a system within a network. This requires, that the learning environment feels like and behaves as close as the real-world setting to the learner. To gather experiences, learners should be able to try out things they learned

¹By “hands-on” we mean tasks in which learners observe or manipulate real objects or material.

in theory. This includes the way of doing things right as well as doing things wrong. In the scope of communication technology, this could mean that students will set up a network environment, either with a valid or an invalid configuration. With respect to IT security, this could mean that someone will protect a system and another one tries to attack it. Depending on the aim of a certain task, everything is working fine, the environment gets down or damaged or something in between can happen. However, this requires that the learning environment is isolated from the real world in order to prevent interferences. Also, the manageability of the hands-on learning environment is crucial for both lecturer as well as learner. E.g. the lecturer may use the environment in different courses with several learners, so a time-saving clean-up and reset process of the environment is required to get an equal initial state for each learner in each course. For the learner, a great user experience as well as a short training period might result in an increased acceptance level which finally can lead to an improved learning outcome.

A common way to organize hands-on experiences is to use a *computer laboratory* or *computer security laboratory* for education. A computer lab in the scope of communication technology and IT security typically consists of a computer or a group of computer systems usually connected into a network. The systems as well as the network behave as in the real world but they are not connected to it - they are neither part of a critical infrastructure nor can connect to any. In other words, the system and also the network are *isolated* from the outside world. Many tasks can be realized using a computer lab which otherwise would not be possible, are forbidden or can cause damage. In the scope of communication technology and IT security, the use of a computer lab is a suitable and common way for hands-on experiences. Nevertheless, this situation still holds challenges and thus can be improved.

The following section 1.1 discusses computer labs in education and covers physical and virtual labs, the educational context and also the research challenge. The technical background specific to this thesis is introduced in the subsequent section 1.2. Section 1.3 introduces the organization of this thesis and covers involved people and environments as well as the contribution and organization of this thesis.

1.1 Computer Labs in Education

Computer labs are of great value in courses that teach security of computer systems and networks. The knowledge that students learn from textbooks or in lessons, is illustrated, deepened, and anchored by carrying out practical exercises in such a lab. In addition, students usually like carrying out practical exercises next to studying theory, thereby improving their motivation and results. E.g. it is exciting to do hacking exercises that would be forbidden or illegal in real systems.

Physical and Virtual Labs

The idea of using an isolated network as an environment to perform IT related tasks for the purpose of research or education is widely recognized [Bishop and Heberlein, 1996, Agarwal et al., 2001, Taylor et al., 1996, Lo et al., 2014, Bardas and Ou, 2013]. There are two general approaches to create such an environment:

The first approach is to create or use an isolated, physical network with physical hosts that is separated from an operational network such as a campus network [Bishop and Heberlein, 1996, Jakab et al., 2009]. This isolation may be achieved by physical separation of the networks or by using components like firewalls to restrict data flow between network areas [Yang et al., 2004]. Within this isolated network the students can perform exercises and work with a real-world like network setup. Remote access to such a network may be granted by using remote access technologies such as Virtual Private Network (VPN) [Yoo and Hovis, 2004]. Administration and maintenance of such a lab however is labour-intensive. Students work in the lab with super-user rights and can modify system configurations at will. After a session, it is necessary to clean up system configurations, which may even require reinstalling operating systems.

The second approach makes use of virtualization technologies to create an isolated, virtual network with virtual hosts. Literature refers to such an environment in the context of education or e-learning usually as a virtual lab [Brian Hay, 2006, Bullers et al., 2006, Damiani, 2006, Keller and Naues, 2006, O’Leary, 2006, Li, 2009a]. This approach significantly reduces the amount of physical hardware resources (e.g., switches, routers, hosts), since the required resources are created by virtualization. Cleaning up or

reinstalling a virtual lab simply means reloading the virtual environments, which can even be an automated task.

Literature also reports two main approaches to provide an isolated network. In the first one, the environment is located at a central place, usually at the university [Drigas et al., 2005, Border, 2007, Hu et al., 2005, Keller and Naues, 2006, Krishna et al., 2005, Lahoud and Tang, 2006] and students can get physical or remote access by using a secured network connection. A central place could also be a cloud [Ellabidy and Russo, 2014, Mhd Wael Bazzaza, 2015, Salah, 2014] or a federated lab [Peterson et al., 2003, Berman et al., 2014]. Although such labs may be accessed remotely at any time from any place, they are generally not easily scalable. Allowing an arbitrary number of students to participate at the same time requires students to reserve timeslots in advance for working in the lab. This may impose restrictions for students in distance education, who usually study in evening hours and weekends. Provisioning a remote lab for peak access outside office hours, may result in a largely over-dimensioned lab with a low average degree of utilization and hence a waste of resources.

Second, the environment is provided as a preconfigured, stand-alone software package which can be installed and used by students on any computer, usually their private computer [Li, 2009b, Vranken and Koppelman, 2009, Li, 2010, Seeling, 2008]. This gives the students the opportunity to safely carry out assignments wherever and whenever they want to.

Educational Context

The Open Universiteit (Netherlands) as well as the Cologne University of Applied Sciences (Germany) offer courses in the field of computer networks and IT security. While the Open Universiteit is a university for distance education, the Cologne University of Applied Sciences is a traditional on-campus university. Both universities offer practical courses in networking and IT security, but the setting is mostly different.

The Open Universiteit offers courses meant for distance teaching. For courses in computer networks and IT security, the Open Universiteit implemented a virtual computer security lab consisting of an isolated, secured software environment that each individual student can easily install on his PC. In this virtual lab, the student can configure and simulate virtual

hosts, connect them into virtual computer networks and safely carry out experiments related to security. It is neither required to provide a computer lab at the campus (or in the cloud) nor required for students to travel there [Vranken and Koppelman, 2009].

The Cologne University of Applied Sciences offers courses for classroom teaching. This university provides the students a classic computer lab to work in. All students of a certain Bachelor program have to take part in the course “Communication technology and networks”, where they learn about concepts and standards of computer networks, hosts and intercommunications. The course consists of lectures in a classroom, accompanied and supplemented by a practical course. Students are required to pass the practical course as a prerequisite to take the exam. According to the curriculum, the practical course’s outcome is either pass or fail while the exam is being graded. The practical course is organized as follows: Students have to register to take part. In a kick off meeting they will get to know the course advisers, the assignments and the computer laboratory. The course advisers are members of the academic staff at the university, have expert knowledge about the course content and are able to support and guide the students. The assignments are related to certain theoretical concepts of the lecture and facilitate that students have to apply their previously acquired knowledge. The aim of the practical course is to make sure, that every student has learned the concepts related to the assignment, has an understanding of the solution and is able to reproduce and defend the solution. To successfully complete the course each student has to demonstrate and defend the solved assignment in a final bilateral expert talk with a course advisor. The course advisor knows the solution and possible ways of solving. He is able to judge whether a student has successfully acquired and applied theoretical concepts of the lecture.

Research Challenge

Providing a safe playground is only one requirement of a computer lab. The challenge at this point is to figure out what environment fits best in a certain learning scenario. This requires to find a balance between technical opportunities, educational requirements and also the demands of the learners. This context is illustrated in figure 1.1 on the following page. Unfortunately, this balance cannot be found as the ultimate solution.

E.g. in [Lahoud and Tang, 2006], “*it was able to address the needs of distance learning students taking security classes*” but “*a broadband-based Internet service is recommended for such lab experiments in order to take full advantage of its functionality*”. In [Seeling, 2008], “*The main benefits resulting from this approach are that students can progress at a time and pace of their choice and the reduced costs in infrastructure for schools*”, but “*one potential drawback of this approach is the hard drive space required on the computers of instructor and students.*”. [Salah, 2014] states, that “*At the end of the one-semester course, the overall reaction from the students and instructor was very positive.*”, but “*MITM (Man-In-The-Middle) attacks are not feasible in the cloud*”. We proceed on the assumption that each setting will have its own strengths and weaknesses.

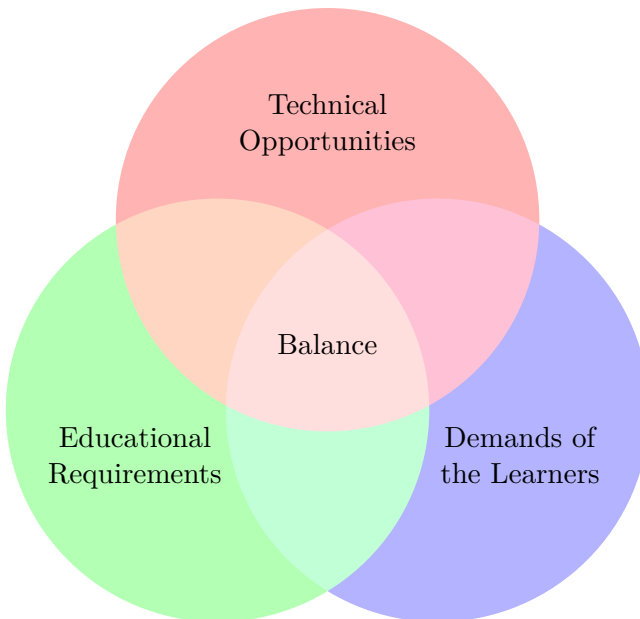


Figure 1.1: The challenge is to find a balance for a certain setting

The aim of the thesis is to improve education. The general idea is to explore the possibilities when combining aspects of two established labs for hands-on experiences in networking and it security courses. A prerequisite

is, that the labs are predominantly different but have an essential overlap in educational requirements. We combine strengths of both labs². We assume, that this can significantly improve the learners hands-on experience.

In 2009, the Open Universiteit as well as the Cologne University of Applied Sciences started a research cooperation in order to improve the quality of learning by means of better support to students and to grant them more flexibility with respect to their learning environment. Both computer labs - the physical lab at the Cologne University of Applied Sciences as well as the virtual computer security lab at the Open University - work fine within their own scope. E.g., the freedom for students to learn at any time and place meets exactly the requirements of a university for distance education. Also, the possibility to intercommunicate face-to-face with other students or with the course advisor is natural at an on-campus university. It becomes interesting when we try to cover or cross both scopes. E.g. students, working in the evening hours at home using the virtual lab, want to be able to solve a networking exercise in a group or the support of a course advisor is needed. Also, students at the on-campus university want to be able to learn outside of the opening hours of the lab. As previously mentioned, each setting will have its own strengths and also weaknesses.

For our research, we identified the following challenges in creating a virtual environment for security and network education. These challenges are

- enabling group work,
- supporting ease of use,
- identifying key aspects for acceptance of the system and
- addressing those key aspects in the system.

The first challenge is targeted at enabling group work in a virtual lab, which was designed to run as a single isolated instance on a student's computer, completely independent of other systems, networks or individuals. This is effectively achieved by extending the existing virtual lab to a

²A similar approach can be found in the concept of blended learning [Bonk et al., 2005]. In blended learning, online digital media will be combined with traditional classroom methods.

distributed virtual computer lab for security and network education. The second challenge is fulfilled by simplifying the way how students will organize their group and how they connect their labs, followed by some applicability enhancements of the new system. The third challenge is dealt with by interrogating students to discover their preferences in learning behaviour. The fourth challenge is achieved in two ways. Firstly, by introducing an electronic exercise assistant, which is able to support and guide students as well as verify a final solution even if a human course advisor is not available. Secondly, we deal with it by creating a virtual computer lab as a social place, where students e.g. meet, form learning groups, enrol for an assignment, talk and discuss.

1.2 Technical Background of this Thesis

The following section will introduce the virtualization environment called User Mode Linux (UML). UML is a special kind of virtualization and is also the base where Netkit is built upon. Netkit will also be introduced afterwards because Netkit is the base where the VCSL is built upon. Finally, this cohesion is the base where this thesis is built upon.

1.2.1 User Mode Linux (UML) and Netkit

The following part is an abstract of [Rimondini, 2007] and also refers to [Dike, 2006].

User Mode Linux (UML) User-Mode Linux³ (UML)[Dike, 2006, Dike, 2000] is a port of the standard Linux kernel⁴ which is designed to run as a userspace process. Being a kernel in itself, UML comes with its own kernel subsystems, including scheduler, memory manager, filesystem, network and devices. In this sense, an instance of UML provides a virtualized environment in which everything (processes, memory, filesystem, etc.) is controlled by itself instead of the host kernel. In practice, UML appears

³The User-Mode Linux Kernel Home Page, <http://user-mode-linux.sourceforge.net>, Online, accessed March 2017

⁴The Linux Kernel Archives, <https://www.kernel.org>, Online, accessed March 2017

as a userspace process on the hosting machine and acts as a kernel for its own processes.

Development In the past UML used to be available in the form of a patch to a standard Linux kernel. Most recent kernels already include user-mode code, thus a UML kernel can be simply built by specifying `um` as target architecture while compiling a vanilla kernel. Since 2002, there also exists a set of additional patches called SKAS that can be optionally applied to the host kernel to change the way UML behaves. The patches have beneficial effects in terms of both security and performance.

Virtualization of CPU and RAM There is an important difference in the approach adopted by full virtualization products, e.g. VMware and that adopted in UML. Full virtualization products usually attain virtualization by directly interfacing with the host hardware, and provide an abstraction layer implementing an architecture that may also be different from the one of the host they are running on. In the case of UML, virtualization takes place within the host kernel rather than at the hardware layer. In other words, UML provides simulated hardware constructed on the basis of services provided by the host kernel. Essentially, UML is a port of the Linux kernel to the Linux system call interface rather than to a specific hardware interface. User space code simply runs natively (no emulation), while processes in kernel mode see a special environment which limits access to host resources. This makes the virtualization faster and more responsive, and is the reason why UML is considered a lightweight emulator. The drawback of this approach is that it only allows to run emulated Linux boxes.

Virtual machines are equipped with a disk, whose raw image is a file in the host machine; they have their own memory region, whose size can be set upon startup; and they can be configured with an arbitrary number of virtual network interfaces which are connected to a virtual hub.

Virtualization of Hard Drive Disk UML mounts a virtual disk device provided by the user and boots the Linux distribution it finds inside it. Virtual disks are managed by a User-mode Block Device (UBD) driver, which uses a standard file on the host filesystem as storage area. This file

can be considered as a disk image. This disk image can be filled with Linux software in a way that is very similar to what would happen on a real host. Indeed, the disk image can be made available as a loopback device on the host machine by using the `losetup`⁵ command. Once done, it can be initialized by using appropriate filesystem tools (e.g. `mkfs`) and populated by using tools such as `debootstrap` on Debian or `pacman` on ArchLinux.

Being a lightweight environment, it is possible to implement complex setups consisting of several instances of UML based virtual machines. Since each virtual machine writes on its own filesystem, this potentially implies using several disk images, which size is often not negligible (hundreds of megabytes). However, the block driver technology is able to support sharing of a filesystem among different virtual machines. This is achieved by writing changes to the disk image into a different file, a technique that is also known as Copy-On-Write (COW). Therefore, a typical setup of a complex emulated scenario consists of a single large disk image file containing a model filesystem and several small COW files that store the changes to the model filesystem. Thus, filesystem information for a virtual machine can only be reconstructed based on both its own COW file and the disk image. Each COW file can only be used together with the backing file it was created from. However, by using the UML utility `uml_moo`, it is also possible to merge the two and get a standalone disk image that contains all the filesystem information for that virtual machine. Furthermore, COW files are implemented as sparse files. A sparse file is one which efficiently uses the filesystem by allocating space only when data is actually written to the file.

Virtualization of Network Interfaces UML allows to configure virtual machines with an arbitrary number of network interfaces. By using appropriate UML command line arguments, these interfaces can be attached to a `uml_switch` process running on the host, which simulates the behaviour of a network switch or network hub. In this way, different virtual machines attached to the same switch can exchange data with each other. More specifically, UML virtual network interfaces can be attached to a UNIX socket. In turn, an `uml_switch` can be attached to the same

⁵Linux Manual Page: "`losetup` is used to associate loop devices with regular files or block devices, to detach loop devices and to query the status of a loop device."

socket and forward packets among the virtual machines that are connected to that socket. Figure 1.2 shows an example of two connected UML virtual machines.

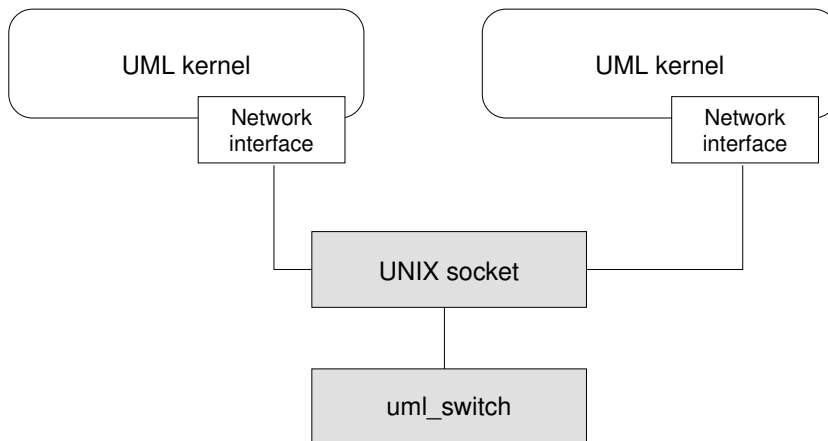


Figure 1.2: This diagram shows how Netkit virtual machines are networked

From the point of view of the network stack, UML provides implementations of the following ISO/OSI⁶ layers:

- The **physical layer** is implemented by a set of `uml_switch` processes running on the host. They are configured to behave as hubs, and packets are forwarded to interfaces of other virtual machines by using UNIX sockets.
- The **data link layer** supports the Ethernet protocol, but collisions cannot happen because the `uml_switch` avoids them.
- The **network layer** supports both IPv4 and IPv6 by means of kernel code and user space utilities.
- What happens on **upper layers** is up to the specific software being run inside virtual machines. For example, running a web server would introduce support to HTTP.

⁶ITU-T X.200: Reference Model of Open Systems Interconnection

Netkit Virtual machine settings can be passed to UML via a command line interface. While this is an effective way of specifying configuration parameters, it is often the case that users interested in just emulating networks are not willing to deal with complex kernel invocation commands. Furthermore, compiling an UML kernel is complex and takes a lot of time. Also preparing the disk image is not an easy task. For this reason, a software package called Netkit was developed at the Roma Tre University⁷ to provide a higher-level user interface to UML.

The goal of Netkit is to provide a ready-to-use virtual experience based on UML. Netkit enriches UML by an intuitive interface consisting of several scripts and a preconfigured disk image (filesystem) containing most state of the art networking tools.

Netkit is conceived for easy installation and usage and does not require administrative privileges for either one of these operations. Starting a virtual machine means starting a UML instance, which often requires dealing with somewhat complex command line arguments. For this reason, Netkit supports straightforward configuration and management of virtual machines. A virtual host can be started by the Netkit's command line utility `vstart` followed by a hostname and parameters which defines the network connectivity. Netkit scripts take also care of automatically starting `uml_switch` processes.

While UML is a lightweight and powerful virtualization environment, Netkit turns UML to a ready-to-use experience for emulating hosts and networks. Since 2017, Netkit is going to become container-based [Bonofiglio et al., 2018]. An improved implementation called Kathará⁸ uses Python and Docker which allow to have much better scalability.

1.2.2 Virtual Computer Security Lab (VCSL)

This chapter introduces the technical architecture of the Virtual Computer Security Lab (VCSL). It is based on and summarizes the work of Harald

⁷Università degli Studi Roma Tre, Via Ostiense 169 - 00154 Rome, Italy, <http://www.uniroma3.it>

⁸Kathará: Implementation of the notorious Netkit using Python and Docker, <http://www.kathara.org/>

Vranken, who is the originator of the VCSL [Vranken and Koppelman, 2009]. This VCSL is the basis for the research work in Part I and Part II of this thesis.

The VCSL is a stand-alone environment, composed of two nested software virtualization layers, that each student can install on his/her computer. The software components to build the VCSL are freeware or open source, and are distributed to students on a DVD. Recently, students can also download the VCSL from a website.

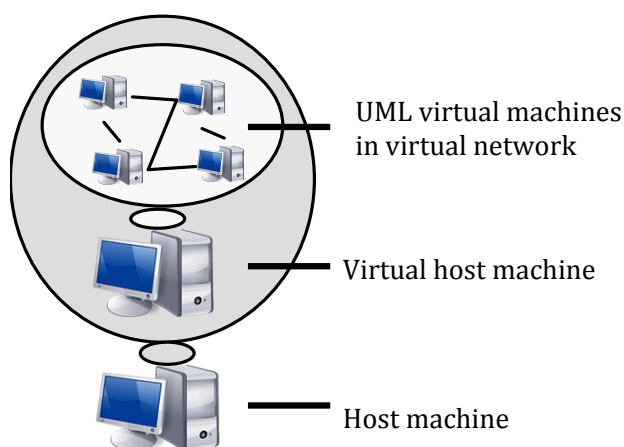


Figure 1.3: Architecture of VCSL

The VCSL is composed of one physical host and two virtualization layers, as shown in figure 1.3. The host machine is the student's computer, which runs an arbitrary operating system, i.e., the host operating system. The first virtualization layer creates the virtual host machine. It consists of virtualization software such as VMware Player⁹ (freeware) or Oracle VM VirtualBox¹⁰ (open source), which runs on the host machine just like an ordinary application. Versions of this software are available for a

⁹VMware Player, <http://www.vmware.com>, Online, accessed November 2017

¹⁰VirtualBox, <http://www.virtualbox.org>, Online, accessed November 2017

large range of platforms. VirtualBox for instance runs on host machines with either Windows, Linux or Mac OS X. This first virtualization layer therefore runs on nearly all student computers, regardless of the hardware and the host operating system. The virtual host machine runs the virtual host operating system. For the VCSL we selected Linux, since it is open source and can be distributed to students without licensing costs. In fact, we selected Knoppix, a bootable live Linux system containing a collection of GNU/Linux applications and the KDE graphical desktop environment.

The second virtualization layer is a Linux application, called Netkit [Pizzonia and Rimondini, 2008, Pizzonia and Rimondini, 2016], that runs inside the virtual host machine. This second virtualization layer allows to instantiate multiple virtual machines that all run Linux. Netkit applies virtualization based upon User Mode Linux (UML) and allows to setup and configure UML virtual machines with virtual network interfaces, and to connect these into virtual networks.

The hardware requirements for running the VCSL are very modest. A few UML virtual machines can already be run smoothly on a PC with a Pentium-4 processor and 256 MB memory. The VCSL has been used successfully in a security course by hundreds of students with only few minor problems.

In the thesis, we refer to the VCSL (Virtual Computer Security Lab) with the abbreviation VCL (Virtual Computer Lab).

1.3 Organization of this Thesis

This section introduces involved people and environments in the scope of this thesis. This section also describes the organization of the remaining of this thesis and highlights my contributions and my role within related projects.

1.3.1 Involved People and Environments

This thesis is the outcome of an international research cooperation between the Dutch Open Universiteit¹¹ and the German Cologne University of

¹¹Open Universiteit, Valkenburgerweg 177, 6419 AT Heerlen, Netherlands

Applied Sciences, Campus Gummersbach^{12,13}. In 2008, a research group was formed with the aim to improve the support for students working on practical networking and IT security exercises in a virtual lab. Major members of this research group were

- Dr. Ir. Harald Vranken, Associate Professor at the Department of Computer Science, which is a subdivision of the Faculty of Management, Science & Technology at the Open Universiteit,
- Prof. Dr. Stefan Karsch, leading Professor at the Laboratory for Communication Technology and Data Security¹⁴, affiliated to the Department of Computer Science, which is a subdivision of the Faculty of Computer Science and Engineering Science at the Cologne University of Applied Sciences and
- Graduate Computer Scientist (Diplom-Informatiker FH) Jens Haag, Academic Staff Member at the Laboratory for Communication Technology and Data Security, affiliated to the Department of Computer Science, which is a subdivision of the Faculty of Computer Science and Engineering Science at the Cologne University of Applied Sciences.

In 2012, also

- Prof. Dr. Marko van Eekelen, Professor Software Technology in the Faculty of Management Science and Technology at the Open University and also Associate Professor at the research section Digital Security of the Institute for Computing and Information Sciences within the Faculty of Science of the Radboud University Nijmegen¹⁵.

joined this research cooperation.

Some work was done, developed and evaluated with or by students at the Laboratory for Communication Technology and Data Security, where I worked from 2008 to 2014. Some findings were gained from and also integrated in the course “Communication Technology and Networks”, lectured by Prof. Dr. Hans Ludwig Stahl.

¹²Fachhochschule Köln / Cologne University of Applied Sciences, Campus Gummersbach, Steinmüllerallee 1, 51643 Gummersbach, Germany

¹³Please note, that the The Fachhochschule Köln / Cologne University of Applied Sciences was renamed to Technische Hochschule Köln / University of Applied Sciences in September 2015

¹⁴<http://www.ktds-koeln.de>

¹⁵Radboud University Nijmegen, Comeniuslaan 4, 6525 HP Nijmegen, Netherlands

1.3.2 Contribution and Organization of this Thesis

This thesis is divided into two parts. The first part *Design Issues of DVCL* reflects the work I was part of, which addresses concepts, design and implementation of the Distributed Virtual Computer Lab. This part is focused on technical issues. The second part *Educational Aspects of DVCL* reflects the work I was part of, which addresses the educational use of the DVCL.

All chapters within these two parts are based on journal articles [Haag et al., 2014a, Haag et al., 2017], reviewed publications at international conferences [Vranken et al., 2011, Haag et al., 2011, Haag et al., 2012, Haag et al., 2013, Haag et al., 2014b], bachelor theses and student project works. This origin is also indicated in a footnote at the beginning of each chapter. I am the author (first author) or co-author (second author) of each conference paper and I was predominantly also the presenter at the related conference. I acted as initiator, mentor, consultant and supporter for the bachelor theses and student projects. All student workings were successfully finished at the Cologne University of Applied Sciences and graded by at least one of the following scientists: Prof. Dr. Stefan Karsch, Prof. Dr. Hans Ludwig Stahl, Prof. Dr. Heiner Klocke and Associate Prof. Dr. Harald Vranken.

Chapter 2: *Distributed Virtual Computer Lab* addresses the conceptual and technical base of the Distributed Virtual Computer Lab, which extends the previous work of Harald Vranken's and Herman Koppelman's Virtual Computer Security Lab [Vranken and Koppelman, 2009]. This chapter is based on the first part of Tobias Horsman's bachelor thesis [Horsmann, 2011] and a conference paper [Vranken et al., 2011]. I was mentor, consultant and supporter for the bachelor thesis and I am the co-author of the conference paper. The paper was reviewed and accepted at the 3rd International Conference on Computer Supported Education (CSEDU 2011) and presented by Harald Vranken.

Chapter 3: *Adding a Central Authority* addresses the conceptual and technical base of the Central Authority for the Distributed Virtual Computer Lab. This chapter is based on the second part of Tobias Horsman's

bachelor thesis [Horsmann, 2011] and a conference paper [Haag et al., 2011]. I was mentor, consultant and supporter for the bachelor thesis and I am the author of the conference paper. The paper was reviewed and accepted at the Computer Science Education Research Conference (CSERC 2011) and presented by me.

Chapter 4: *Applicability Enhancements* addresses work that arises during the research and development process of the DVCL. This work was necessary and essential to push the prototypical implementation of the DVCL environment closer to a productive learning environment. **Section 4.1:** *Security* covers and resolves security issues and is based on Christian Doehring's and Sandra Kahrau's student project work [Döhring and Kahrau, 2011], which was supported and supervised by me. The outcomes are published in [Haag et al., 2017]. **Section 4.2:** *Graphical User Interface* introduces a Graphical User Interface to increase the usability of the DVCL environment. This chapter is based on Christian Doehring's student project work [Döhring, 2012], also supported and supervised by me. The outcomes are part of [Haag et al., 2017] too.

Chapter 5: *Course Evaluation* shows the results of an evaluation I did in a practical networking course at the Cologne University of Applied Sciences with about 200 participants, in order to get key factors about student's learning behaviour and success. This chapter is based on a paper [Haag et al., 2013], reviewed and accepted at the 2nd International Conference on E-Learning and E-Technologies in Education (ICEEE 2013) and presented by me. Furthermore, I am the author of an extended version of these research results, reviewed and published in the *Yükseköğretim Dergisi / Journal of Higher Education* [Haag et al., 2014a].

Chapter 6: *Electronic Exercise Assistant* introduces a new software program, which was designed and developed by me. I also did major parts of the coding. This program is able to guide and verify certain networking exercises in the DVCL. This chapter is based on two reviewed conference papers [Haag et al., 2012, Haag et al., 2014b] and a bachelor thesis [Alfers, 2013], which I supported and supervised at the Cologne University of Applied Sciences. I am the author of both papers, the first one was presented

by me at the 3rd Annual International Conference on Computer Science Education: Innovation and Technology (CSEIT 2012), the second one was presented by my then colleague Christian Witte at the 6th International Conference on Computer Supported Education (CSEDU 2014). Our contribution at the CSEDU 2014 conference won the best student paper award.

Chapter 7: *Educational Enhancements* introduces a virtual classroom concept for the DVCL environment and a related prototype. Similar to chapter 4, this student work was also targeted to push the prototypical implementation of the DVCL environment closer to a productive learning environment. This chapter is based on Christian Doehring's bachelor thesis [Döhring, 2013], where I was mentor, supporter and consultant. The results are also published in [Haag et al., 2017].

Part I

Design Issues of DVCL

Chapter 2

Distributed Virtual Computer Lab

The Virtual Computer Security Lab (VCSL) [Vranken and Koppelman, 2009] can be easily installed and run locally on each student's computer. This decentralized approach is suited to accommodate any number of students, provides students the freedom to run the lab whenever and wherever they want, and eliminates the need for a central lab at the university. A shortcoming however is that students have to work on their own. It is for instance impossible to offer exercises on distributed attacks involving large botnets, or hacking games in which students are challenged to attack each other's systems and securing their systems against attacks from fellow students. We therefore extended the VCSL towards a distributed virtual computer lab (DVCL). The DVCL allows connecting the VCSL's of multiple students into a large virtual network running over the internet. Traffic inside the virtual network is completely isolated from the outside world. Hence, communication between virtual hosts inside the DVCL and hosts outside the DVCL is impossible. Students can therefore safely carry out assignments related to security without any restrictions - even spreading malware could be allowed - without the risk of accidentally (or intentionally) attacking or infecting hosts on the internet.

In this chapter, we outline the architecture and implementation of

This chapter is based on the following publications: [Horsmann, 2011, Vranken et al., 2011]

the DVCL. Each VCSL consists of a number of UML virtual machines, connected by virtual networks, and also connected to the virtual host machine. For building a DVCL, a transparent connection between the virtual host machines of distinct VCSL's is required. We therefore equip each VCSL with an interface, such that a point-to-point connection can be created between two VCSL's.

2.1 Core DVCL Architecture

In order to connect two VCSL's transparently, we connect the VCSL's at OSI-layer 2¹⁶. We presume Ethernet-based networks, and hence the Ethernet protocol running at OSI-layer 2. Connecting two virtual networks at OSI-layer 2 requires that Ethernet frames in the first virtual network should be transmitted transparently to the second, remote virtual network and vice versa. By connecting VCSL's at OSI-layer 2, the connected virtual networks will behave like a single broadcast domain [Comer, 2008]. Hence, students working in the DVCL have the notion of being connected to other students over an Ethernet Local Area Network (LAN), although the actual connection is by a Wide Area Network (WAN) using the public internet involving the entire TCP/IP protocol-stack. This is in contrast with conventional network operation, where LAN-frames are converted into WAN-packets at the network perimeter by gateways, preserving the original payload of the upper OSI layers. Our approach also differs from VPN's, where tunnelling is generally done at OSI-layer 3¹⁷ (e.g., when applying IPsec) or above (e.g., when applying SSL).

Performing practical exercises on networking and IT-security introduces an additional requirement: network data that is transported inside a DVCL must not harm non-participating systems also connected the WAN. It should be absolutely assured that non-participating systems, such as the host systems on which the VCSL's run and any other hosts in the internet, are not affected by the transmitted data. Hence, traffic inside the DVCL should be completely isolated from the world outside the DVCL.

To connect virtual networks, we first examine the virtual networking architecture of Netkit.

¹⁶Open Systems Interconnection Model (OSI) Layer 2 is the data link layer

¹⁷Open Systems Interconnection Model (OSI) Layer 3 is the network layer

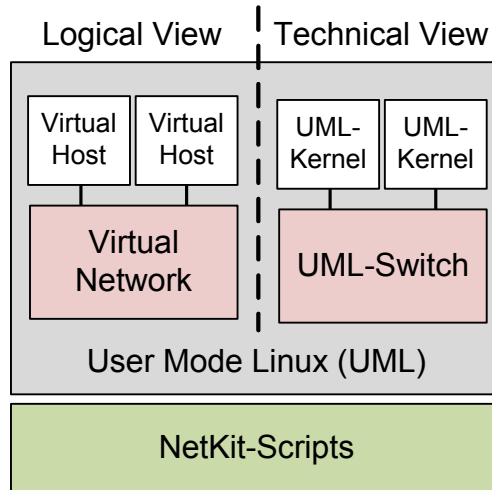


Figure 2.1: Architecture of Netkit

Figure 2.1 shows the architecture of Netkit and provides a logical and a technical view of its components. The logical view shows two virtual hosts which are connected in a virtual network. In the technical view, the virtual hosts are UML-kernels that communicate with each other using the service provided by an UML-Switch.

As the name implies, a UML-Switch connects two or more virtual hosts with switch-like behaviour. A network switch establishes a logical point-to-point connection between hosts in a network that are connected to the ports of the switch.

The UML-Switch can also be configured to behave like a hub [Dike, 2006]. Virtual hosts that are connected in a virtual network with hub-like behaviour can be considered to be connected in the same network segment [Schreiner, 2009]. In a network segment, a host receives all data sent by other hosts in the network segment, even if the host is not the designated receiver of the data. Netkit uses the UML-Switch with this hub-like behaviour, which offers users the opportunity to analyse data transmitted from any virtual host in the virtual network.

In the DVCL, we connect two remote virtual networks by extracting data at a local UML-Switch and sending it across a connection network to

a remote UML-Switch where the extracted data is then injected, and vice versa. In this way two virtual networks can be connected. Implementing this approach is done by two subtasks:

- Subtask A: extracting and injecting network data at a UML-Switch.
- Subtask B: sending and receiving the extracted network data across a network.

Subtask A: extracting and injecting

We examine the technical implementation of a virtual network as shown in figure 2.1 on the preceding page in which the UML-Switch is involved. For each virtual network, an instance of the UML-Switch is running, which in fact uses a UNIX-Socket to communicate with the virtual hosts. A UNIX-Socket is a system resource which serves as a communication endpoint and can be used for remote communication or local inter-process communication [Stevens, 2003].

The UML-Switch attaches itself to a UNIX-Socket, and listens for incoming data sent by the virtual hosts to the UNIX-Socket. The UML-Switch reads the network data from the UNIX-Socket and writes the data to all other connected virtual hosts, which creates the hub-like behaviour of the UML-Switch.

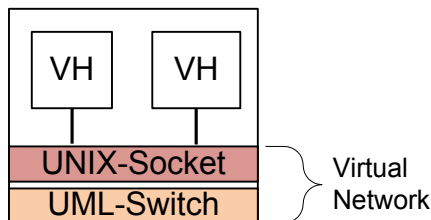


Figure 2.2: Construction of a virtual network

Figure 2.2 shows the architecture of a virtual network. For simplicity, only two virtual hosts (VH's) are shown, but multiple VH's can be connected to the virtual network. The VH's are connected to the UNIX-Socket. Communication over the virtual network is realized by the VH's sending data to the UNIX-Socket, and the UML-Switch forwarding this data to all other connected VH's in the same virtual network.

Extracting data from the UML-Switch and injecting data from a remote virtual network into the UML-Switch, relies on the hub-like behaviour of the UML-Switch. We developed a new software component, that is applied in the virtual host operating system and that connects itself to the UNIX-Socket of the virtual network. This component can be considered as a ghost host in the virtual network because it is completely transparent to the other components. With this ghost host, it is possible to extract all Ethernet frames from the UNIX-Socket, as well as to inject frames received from a remote virtual network. Compared to a normal virtual host, the ghost host is not jailed in the Netkit environment. The ghost host can therefore communicate with the outside world, which cannot be done by a normal virtual host. The UML-Switch with hub-like behaviour guarantees that network data received from a remote virtual network is distributed to all other locally connected virtual hosts.

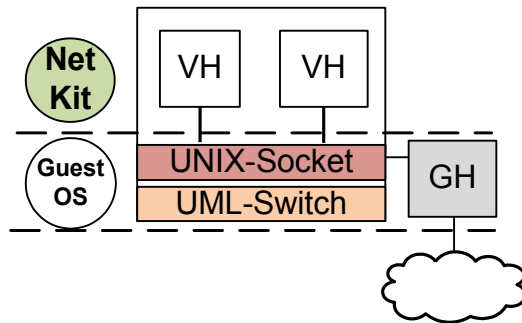


Figure 2.3: Virtual network with ghost host

Figure 2.3 shows the construction of the virtual network in figure 2.2 on the preceding page. The figure also shows the logical layers of the virtual network. The virtual hosts are located and jailed within the Netkit layer. The implementation of the virtual network is realized in the virtual host operating system, which may also be referred to as the guest OS. In the guest OS layer, a ghost host (GH) is located which realizes subtask A. The ghost host can extract all data that is sent over the virtual network and can forward the data to a destination outside the guest OS (cloud). Incoming data from outside (cloud) can be injected in the UNIX-Socket using the ghost host.

Hence, the ghost host provides an interface to the virtual network, where Ethernet frames can be extracted and injected without the need to modify any existing components of Netkit or UML.

Subtask B: sending and receiving

Subtask B consists of sending the extracted Ethernet frames from one ghost host to another ghost host in a remote system. Ethernet frames cannot be sent directly over the internet or a WAN [Comer, 2008], since those networks require a network protocol which runs on OSI-layer 3, presumably the Internet Protocol (IP). To resolve this issue, the Ethernet frames are sent over a remote bridge which is established between two ghost hosts.

A remote bridge can connect two distant networks by using a connection network [Schürmann, 2004]. A benefit of the remote bridge is that the connection network and the distant networks can use different protocols. Even if the protocols of the connection network and the distant networks are equal, the remote bridge ensures that the distant networks cannot intercommunicate with the connection network. The remote bridge consists of two remote bridge endpoints which connect two networks at OSI-layer 2. Such an endpoint encapsulates an Ethernet frame of the local environment in a transport protocol, and sends the data to the remote endpoint where the transport protocol is removed and the OSI-layer 2 data is injected in the remote environment.

We extended the ghost host component and added functionality of a remote bridge endpoint. The Ethernet frames that are extracted by the ghost host, are first encapsulated in the IP protocol (acting as transport protocol), and next sent to the remote bridge endpoint of a fixed distant destination. For incoming data, the IP protocol is removed by the ghost host and the Ethernet frames are sent into the local network, respectively the local UNIX-Socket.

Building the DVCL

With subtask A and B in place, we can connect two distant virtual networks by interfacing to a remote bridge between the virtual networks. Extracted data of one virtual network is encapsulated by the remote bridge endpoint

into a transport protocol, and sent to a distant remote bridge endpoint where the data is unpacked and injected into the local virtual network.

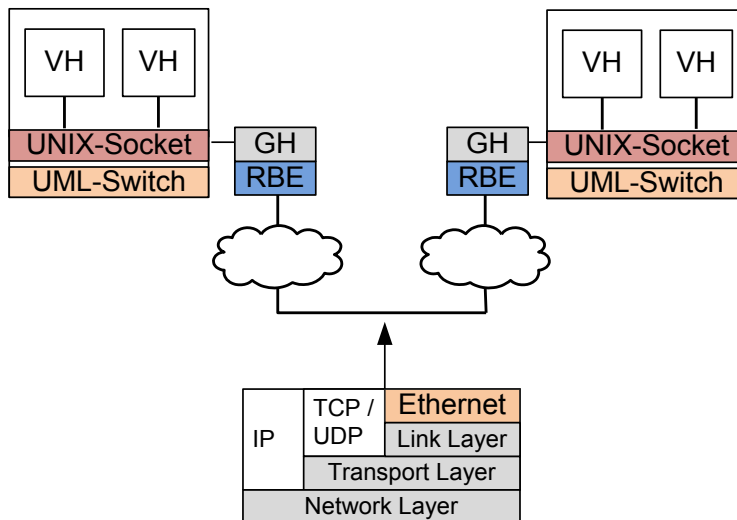


Figure 2.4: Connecting two virtual networks

Figure 2.4 shows an example of two virtual networks. Each virtual network has two virtual hosts (VH) and a ghost host (GH) offering a remote bridge endpoint (RBE), attached to the UNIX-Socket. The Ethernet frames that are extracted from a virtual network are sent across a transport network (cloud) encapsulated into a transport protocol. As transport protocol TCP/IP or UDP/IP is assumed.

2.2 DVCL Example Setups

Figure 2.5 on page 29 shows an example setup of a DVCL for two students (Student A and Student B). It is presumed that each student uses his/her own computer connected via a local or wide area network (LAN/WAN). In the example setup, Student A is at IP address 192.168.2.102 and can reach Student B at IP address 192.168.2.101, and vice versa. Each student locally runs a VCSL environment, but a student may also run a native Linux operating system, thereby eliminating the need for running

the first virtualization layer. In the example, each student sets up a Netkit environment consisting of two virtual hosts (VH) connected in a local virtual network. For example, Student A starts VH1 by issuing the commands in listing 2.1.

Listing 2.1: Setup virtual host

```
1 // Start VH1 connected to network netA
2 vstart VH1 --eth0=netA
3 // Setup network interface on VH1
4 ifconfig eth0 192.168.2.200 netmask 255.255.255.0 up
```

In a similar way, Student A starts VH2, and Student B starts VH3 and VH4 connected to netB. The local networks are connected as a distributed local subnet by using our developed utility, consisting of a remote bridge endpoint and a ghost host. Listing 2.2 shows the command, used by student A, to connect to the remote bridge endpoint of Student B, using this user application named “plug”.

Listing 2.2: Remote connection

```
1 // Start remote connection to Student B
2 plug --source-ip 192.168.2.102 --source-port 53838 --
   destination-ip 192.168.2.101 --destination-port 32000 --
   uml-switch-socket /path/to/vhub_USERNAME_netA.cntc
```

Student B will do the same, connecting to the remote bridge endpoint of Student A. From now on, netA and netB are connected with each other, and all four virtual hosts can reach each other.

In the following, we provide two scenarios that demonstrate the correct and secure operation of the DVCL shown in figure 2.5. In these scenarios, we use “ping”, a well-known tool for testing the connectivity between two computer systems by sending an ICMP echo requests¹⁸ and receiving an ICMP echo reply. We use Wireshark (www.wireshark.org), a tool for network protocol analysis, to visualize the recorded network data.

¹⁸RFC792: Internet Control Message Protocol, pages 14–15

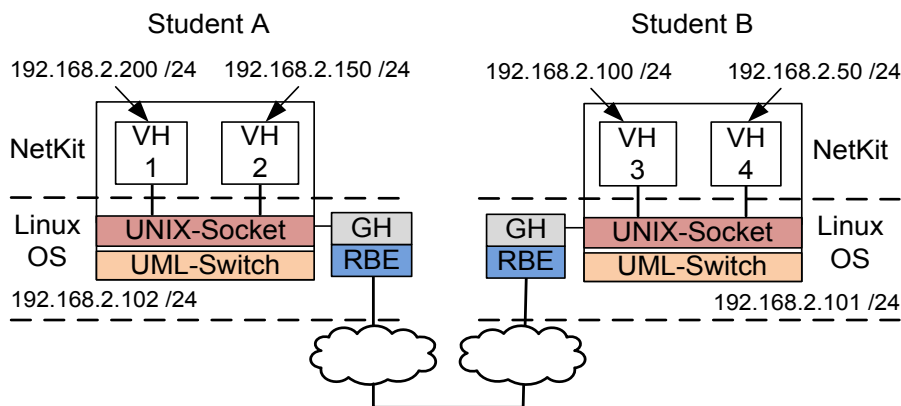


Figure 2.5: Example setup

Scenario 1

Scenario 1 demonstrates that virtual host VH1 of student A can communicate with virtual host VH4 of student B. Moreover, scenario 1 shows that the virtual hosts of student A and B act as if they were connected in the same network segment (hub-like behaviour). Student A runs “ping” on VH1, trying to connect to VH4. The expected result is that VH1 first sends an address resolution request, using ARP (Address Resolution Protocol), to obtain the Media Access Control (MAC) address of VH4. This procedure is common for IP/Ethernet-based networks to obtain the Ethernet address of a host when only its IP address is known. Once the MAC address of VH4 is obtained, VH1 sends an echo request which is answered by an echo reply. The network data which is sent between the Linux operating systems should be encapsulated in a transport protocol.

Figure 2.6 on the next page shows the network data captured at virtual network interface eth0 of VH1. As expected, an ARP request is sent to obtain the MAC address of virtual host VH4 (No. 1). Receiving the ARP reply indicates that both hosts are in the same local subnet (No. 2). Afterwards, the echo request is sent which is replied by VH4 (No. 3-4). The IP addresses of the virtual network interfaces of VH1 and VH4 are correctly shown as source and destination addresses.

Figure 2.7 on the following page shows the network data that is sent

No.	Time	Source	Destination	Protocol	Info
1	0.000000	76:44:5d:e6:c3:f8	Broadcast	ARP	Who has 192.168.2.50? Tell 192.168.2.200
2	0.051723	4a:4d:47:64:80:ff	76:44:5d:e6:c3:f8	ARP	192.168.2.50 is at 4a:4d:47:64:80:ff
3	0.051725	192.168.2.200	192.168.2.50	ICMP	Echo (ping) request
4	0.083610	192.168.2.50	192.168.2.200	ICMP	Echo (ping) reply

.....

▶ Frame 1 (42 bytes on wire, 42 bytes captured)

▶ Ethernet II, Src: 76:44:5d:e6:c3:f8 (76:44:5d:e6:c3:f8), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

▶ Address Resolution Protocol (request)

Figure 2.6: Wireshark capture inside the virtual lab (Scenario 1)

between VH1 and VH4, captured at the network interface of the underlying Linux operating system on student A’s computer. Due to the encapsulation of the network data in a transport protocol (shown as UDP), the IP addresses of the Linux operating systems are correctly shown as source and destination addresses. The payload of the first UDP packet has a size of 42 bytes which corresponds to the size of the ARP request as seen in figure 2.6. We conclude that the virtual hosts of the two students are capable of communicating with each other and that the network data is sent encapsulated in a transport protocol.

Scenario 2

Scenario 2 demonstrates that virtual host VH1 of student A cannot communicate with the Linux operating system of student B, although their IP addresses are located in a common subnet (192.168.2.0/24). The

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.2.102	192.168.2.101	UDP	Source port: 53838 Destination port: 32000
2	0.008243	192.168.2.101	192.168.2.102	UDP	Source port: 59734 Destination port: 32000
3	0.044800	192.168.2.102	192.168.2.101	UDP	Source port: 53838 Destination port: 32000
4	0.055193	192.168.2.101	192.168.2.102	UDP	Source port: 59734 Destination port: 32000

.....

▶ Frame 1 (84 bytes on wire, 84 bytes captured)

▶ Ethernet II, Src: AppleCom_98:77:33 (00:1b:63:98:77:33), Dst: Giga-Byt_d4:84:6f (00:16:e6:d4:84:6f)

▶ Internet Protocol, Src: 192.168.2.102 (192.168.2.102), Dst: 192.168.2.101 (192.168.2.101)

▶ User Datagram Protocol, Src Port: 53838 (53838), Dst Port: 32000 (32000)

▶ Data (42 bytes)

Figure 2.7: Wireshark capture outside the virtual lab (Scenario 1)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	72:e7:62:b4:5c:c9	Broadcast	ARP	Who has 192.168.2.101? Tell 192.168.2.200
2	1.007148	72:e7:62:b4:5c:c9	Broadcast	ARP	Who has 192.168.2.101? Tell 192.168.2.200
3	1.999256	72:e7:62:b4:5c:c9	Broadcast	ARP	Who has 192.168.2.101? Tell 192.168.2.200

▶ Frame 1 (42 bytes on wire, 42 bytes captured)					
▶ Ethernet II, Src: 72:e7:62:b4:5c:c9 (72:e7:62:b4:5c:c9), Dst: Broadcast (ff:ff:ff:ff:ff:ff)					
▶ Address Resolution Protocol (request)					

Figure 2.8: Wireshark capture inside the virtual lab (Scenario 2)

expected result for this scenario is that no communication between VH1 and the Linux operating system of student B is possible due to the strictly separated networks. VH1 sends ARP requests to obtain the MAC address of Student B's Linux operating system, but these ARP requests will never be received by the Linux operating system of Student B. Instead, they are encapsulated and sent to the remote bridge endpoint of Student B.

Figure 2.8 shows that VH1 tries to obtain the MAC address of the Linux operating system of Student B by sending ARP requests several times. These requests are not answered due to non-existing connectivity to the network of the Linux operating systems.

Figure 2.9 on the following page shows that the Linux operating system receives the ARP request encapsulated in the transport protocol. The payloads of the UDP packets have a size of 42 bytes which corresponds to the size of the ARP requests as seen in figure 2.8. Due to the encapsulation, the Linux operating system does not recognize the packets as ARP requests. It therefore does not send a reply, but sends the network data to the remote bridge endpoint. The virtual network of student B however does not contain a VH with IP address 192.168.2.101, and hence no echo reply will be sent. (This also shows that student B may add a VH with this IP address. Hence there is no restriction on the IP addresses that can be used for VH's inside the DVCL.)

We conclude that no connectivity is possible between the Linux operating systems and the virtual hosts, although they use the same subnet. This shows that the DVCL is securely isolated from the outside world.

Other use of DVCL

A recent application of the DVCL in a different scope can be found in [Jonkman, 2016]. J. H. A. Jonkman shows theoretically and practically that botnet¹⁹ simulations are possible in the DVCL. This makes it possible to perform repeatable and verifiable scientific empirical research on botnets using our DVCL.

2.3 Conclusion

We presented a DVCL in which remote students can perform network security exercises inside an encapsulated common networking environment. The DVCL is built by connecting distinct VCSL's transparently at OSI-layer 2 across an arbitrary TCP/IP-based WAN infrastructure like the internet. To implement this connection, we designed a software component called ghost host with an interface to access local virtual network traffic. The ghost host can extract and inject Ethernet frames. We used the concept of a remote bridge endpoint to transport all local OSI-layer 2 traffic between remote ghost hosts across a TCP/IP-based WAN. As a proof of concept, we demonstrated an example setup which shows that both major goals of our effort are reached: the remote virtual networks are connected transparently at OSI-layer 2 and no intentional or unintentional

¹⁹“Bots or software robots are unwanted software applications that are remotely managed by a botmaster, often unknowingly by the owner of the infected computer system. The botmaster can give these bots commands and thereby cause harm to users on the Internet. The botmaster accomplices this via the C&C Systems of the botnet.”

No..	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.2.102	192.168.2.101	UDP	Source port: 58255 Destination port: 32000
2	1.004712	192.168.2.102	192.168.2.101	UDP	Source port: 58255 Destination port: 32000
3	2.010133	192.168.2.102	192.168.2.101	UDP	Source port: 58255 Destination port: 32000


```

▶ Frame 1 (84 bytes on wire (84 bytes captured) on interface eth0)
▶ Ethernet II, Src: AppleCom_98:77:33 (00:1b:63:98:77:33), Dst: Giga-Byt_d4:84:6f (00:16:e6:d4:84:6f)
▶ Internet Protocol, Src: 192.168.2.102 (192.168.2.102), Dst: 192.168.2.101 (192.168.2.101)
▶ User Datagram Protocol, Src Port: 58255 (58255), Dst Port: 32000 (32000)
▶ Data (42 bytes)

```

Figure 2.9: Wireshark capture outside the virtual lab (Scenario 2)

damage can affect systems not participating in the DVCL.

Summarized our DVCL will allow remote students to attend practical courses in network security similar to courses performed in a real safeguarded networking laboratory on a technical level. As an overall result, this is a considerable step towards combining the advantages of distance education and on-site training.

Chapter 3

Adding a Central Authority

In the previous chapter we showed how a DVCL can be built by means of a transparent connection between two or more VCSL's [Vranken et al., 2011]. This resulted in a decentralized architecture with point-to-point connections between the VCSL's. We equip each VCSL with an interface consisting of a ghost host and a remote bridge endpoint, such that a transparent point-to-point connection can be established.

For connecting to a remote bridge endpoint, a student needs to know the IP address and port of the computer where the remote bridge endpoint is located. He has to perform configurations on the guest operating system level and inside the UML environment. With the point-to-point connection, also more than two VCSL's can be connected, e.g., student A can connect to student B and student B can connect to student C. As result, the three virtual networks are interconnected into a distributed virtual network and all virtual hosts in these virtual networks can communicate with each other. However, connecting three or more VCSL's can lead to an endless circular flow of network data and causes high network load. Avoiding circular flow requires careful planning by the students, which becomes more error prone as the number of interconnected virtual networks grows. Thus, managing and organizing the actual setup may consume more time than carrying out the actual assignment. To solve this, we suggest a centralized architecture, involving a central authority that connects and manages multiple VCSL's.

This chapter is based on the following publications: [Haag et al., 2011, Horsmann, 2011]

With an appropriate design, the central authority provides a scalable architecture to which preconfigured VCSL's can be connected easily.

3.1 Core CA Architecture

The DVCL with central authority (CA) consists of the VCSL's that run at the computers of remote students, and that are all connected to the central authority. Students interact with the central authority only when connecting their VCSL to the DVCL, and for managing the DVCL. Once the DVCL has been set up, the central authority is completely transparent. It acts as a point of distribution: network data that is send between virtual hosts in different VCSL's, is forwarded by the central authority in a transparent way. In fact, since all connected VCSL's form a single broadcast domain, the central authority broadcasts the network data send by a VCSL to all other connected VCSL's.

The central authority offers functionality to manage multiple DVCL's. Hence, different, disjunct groups of students can work in separate DVCL's, that are all created and managed using the central authority. The central authority refers to each DVCL as a session. Hence, a session corresponds to group of interconnected VCSL's. By adding sessions and session management, two functional layers can be differentiated in the central authority: a control layer for managing the sessions, and a session layer in which the actual DVCL's run. The control layer is used by the students to operate the DVCL, using the operations as shown in table 3.1.

Table 3.1: Student's interaction with central authority

No.	Description
1	Querying a running session
2	Creating a new session
3	Joining a session
4	Deleting a session

Based on the differentiation of two layers, conceptually also two kinds of communication channels are differentiated: the control channel for transporting control data, and the session channel for transporting session data that is sent to a running session at the session layer. Since the central authority is a central component where all network and management data passes, it is an ideal location to record data. Event data (for example: log-in, log-off of students) can be recorded at both the control layer and the session layer, while the session layer also offers recording of the sent network data for each session. An instructor or tutor can examine and evaluate this data, and provide feedback to the students or grade their work, or analyse statistics about usage of the DVCL.

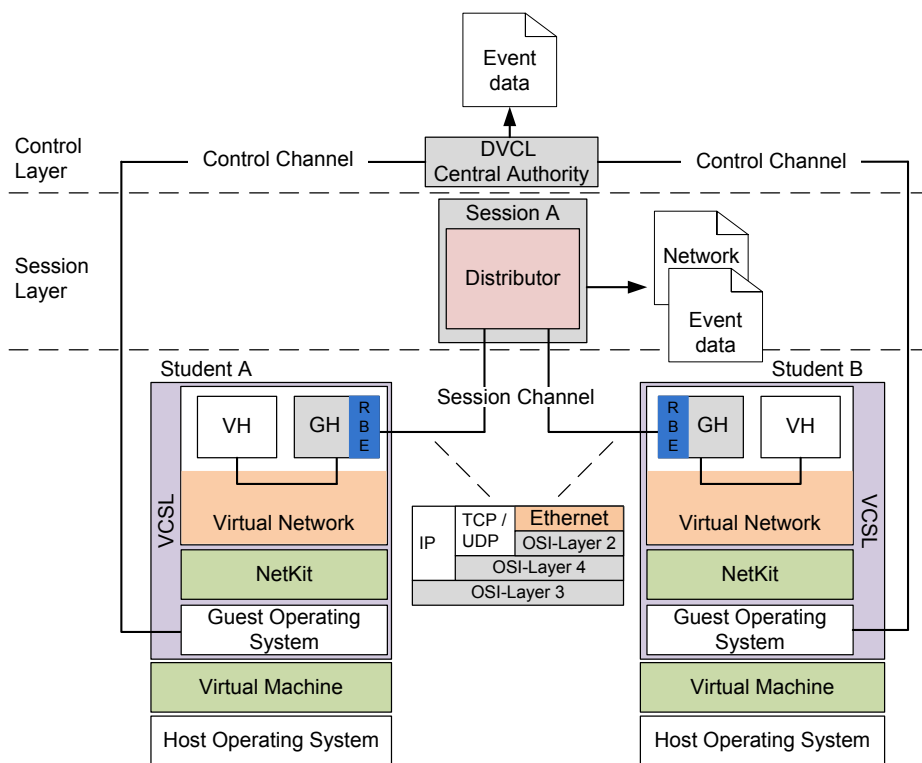


Figure 3.1: Connecting two virtual networks

Figure 3.1 shows a DVCL with CA running a single session. The figure

shows two students running a VCSL on their host operating system. In those VCSL's, each student runs a single virtual host (VH). Obviously, each student can run multiple VH's and connect them into a virtual network, multiple students can connect their VCSL, and different groups of students can run multiple sessions. This scalability is omitted in the figure for simplicity. The students use the control channel to operate the central authority and connect their virtual network to a session, using the operations shown in table 3.1 on page 36. Once the student decides to connect his virtual network to a certain session, a ghost host with remote bridge endpoint is created which connects the virtual network to the session. The ghost host sends the network data of the virtual network across the session channel to the session it is connected to. Through this connection, the virtual hosts in the student virtual networks are capable to communicate with each other, as if they were connected in the same local network. Figure 3.2 on the next page shows the transmission of an encapsulated OSI-Layer 2 frame across the session channel. The figure also shows that event data is recorded at both layers, and network data is recorded at the session layer.

3.2 CA Example Setup

We implemented a prototype of the DVCL with central authority using a client-server architecture. The central authority acts as a server and is hosted by the university. Each student runs a management client application which supports connecting his/her VCSL to the central authority. Figure 3.2 on the facing page shows an example setup of the DVCL with central authority connecting three VCSL's. For simplicity, also this figure only shows VCSL's with a single virtual host and a single session. The VCSL's underlying virtual machine and host operating system layer, as shown in figure 3.1 on the previous page, are omitted in figure 3.2. We also omit the first virtualization layer as indicated in figure 1.3 on page 13, assuming that the host operating system is the Linux operating system in which Netkit runs. Figure 3.2 shows the control channels and session channels as well as the IP addresses and port numbers which are assumed for this example.

We demonstrate in listing 3.1 on page 43 how each student first starts

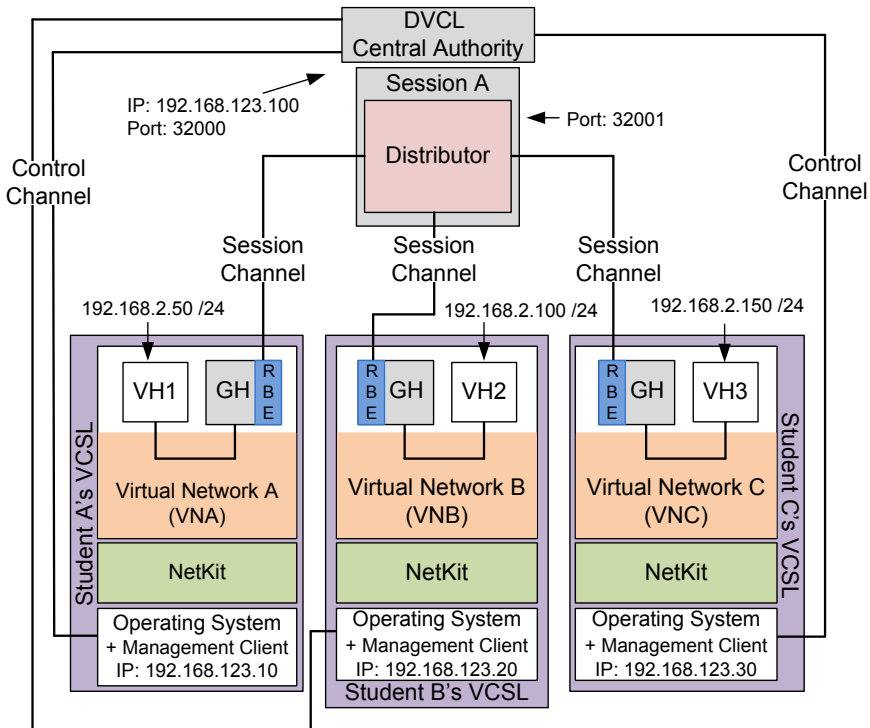


Figure 3.2: Example setup

his VCSL, and next connects to the DVCL. For both operations, the student enters command lines in a shell of the operating system in which Netkit runs.

Student A first runs a Netkit script to start virtual machine VH1. VH1 has a virtual network interface `eth0` that is connected to virtual network VNA. Once VH1 is started, the student configures the virtual network interface. Similarly, student B starts VH2. VH2 has a virtual network interface `eth0` that is connected to virtual network VNB. Once VH2 is started, student B configures the virtual network interface on VH2. And similarly, student C starts VH3. VH3 has a virtual network interface `eth0` that is connected to virtual network VNC. Once VH3 is started, student C configures the virtual network interface on VH3.

For connecting the created virtual networks according to figure 3.2, we assume that student A first connects to the central authority using the management client. Please note, that all IP addresses and ports are exemplary. The management client connects to port 32000 at IP address 192.168.123.100, which is at the server running the central authority. The server offers port 32000 to serve incoming connections. Next, the student can use the operations shown in table 3.1 on page 36. Presuming that no session is running yet, student A creates a new session. Each session corresponds to a port number on the machine running the central authority. In this example port 32001 is presumed to be used by `session_A`. For connecting virtual network VNA to `session_A`, the student uses the `join` operation and provides the port number of the session. By joining the session, the ghost host with remote bridge endpoint is created that exchanges the virtual network's data with `session_A`. Student A is now connected to the central authority and awaits other students to join.

Next, student B connects to the central authority in a similar way. Student B queries which sessions are running and the central authority returns the list of all running sessions. The last step is to connect virtual network VNB to the session created by student A.

Student C performs commands analogous to student B. The virtual networks of the students are now connected into a distributed virtual network with a common broadcast domain. Communication between the virtual hosts VH1, VH2 and VH3 is enabled now. The Ethernet frames that are exchanged between these virtual hosts are encapsulated/decapsulated

in the communication interface of each VCSL and distributed by the central authority.

3.3 Discussion on Scalability

A shortcoming however is the scalability. Depending on the student's task, more or less network traffic will occur. E.g. setting up and configuring a certain network scenario (e.g. NAT) will result in a small number of network packets, compared to a botnet simulation, where a huge amount of network traffic can be expected. In any case all network traffic has to pass or rather has to be distributed by the CA. This could lead into a resource issue with different side effects, e.g. in settings, where timing is crucial, the intermediate CA will cause a delay because the packets may have to pass additional hops. Also, the maximum network load at the CA should be taken into account to avoid an overload. This can affect the availability of the CA and maybe other systems within the provider's network. If a higher load of network traffic is expected, the use of the DVCL command line client (see chapter 2) is recommended to establish a direct (peer-to-peer) connection between the remote virtual networks. Another solution could be that the CA initiates direct connections between the DVCL clients for the Session Channel, e.g. by using UDP hole punching²⁰. This could also reduce network load at the CA.

However, a benefit of managing and distributing all network traffic of the VCSL clients is the "god mode". The network traffic of all started and connected virtual networks will pass the CA and can therefore be accessed and observed. This is necessary if the Electronic Exercise Assistant (see chapter 6) will be added.

3.4 Conclusion

The DVCL enables remote students to perform network security exercises inside an encapsulated common networking environment. The DVCL is built by connecting distinct VCSL's transparently at OSI-layer 2 across an

²⁰UDP hole punching is a method for establishing bidirectional UDP connections between Internet hosts in private networks using network address translation. https://en.wikipedia.org/wiki/UDP_hole_punching, Online, accessed November 2017

arbitrary TCP/IP-based WAN infrastructure like the internet. To implement this connection, we designed a communication interface consisting of a ghost host, to extract and inject Ethernet frames, and a remote bridge endpoint, to transport these frames between remote ghost hosts across a TCP/IP-based WAN. Each VCSL is connected through this communication interface to a session at the central authority, which manages the connections and distributes network data among the connected VCSL's within the session. The central authority can manage multiple sessions. We demonstrated a prototypical implementation of a DVCL with central authority using a client-server architecture. A big benefit of this CA is that it is not anymore required for students to deal with IP addresses of other learning partners. The CA takes care of all connections and additionally reduces the risk of a misconfiguration.

Listing 3.1: Example Setup

```
1 // Student A starts VH1
2 vstart VH1 --eth0=VNA
3 // Student A configures the network interface of VH1
4 ifconfig eth0 192.168.2.50 netmask 255.255.255.0 up

5 // Student B starts VH2
6 vstart VH2 --eth0=VNB
7 // Student B configures the network interface of VH2
8 ifconfig eth0 192.168.2.100 netmask 255.255.255.0 up

9 // Student C starts VH3
10 vstart VH3 --eth0=VNC
11 // Student C configures the network interface of VH3
12 ifconfig eth0 192.168.2.150 netmask 255.255.255.0 up

13 // Student A connects to the CA
14 connect VNA to 192.168.123.100:32000
15 // Student A creates a new session
16 create session_A
17 // Student A joins the created session
18 join session_A:32001

19 // Student B connects to the CA
20 connect VNB to 192.168.123.100:32000
21 // Student B queries which sessions are running
22 query sessions
23 // CA returns list of running sessions
24 running sessions:
25     session_A:32001
26 // Student B joins the session
27 join session_A:32001

28 // Student C connects to the CA and performs commands
    analogous to student B
```

Chapter 4

Applicability Enhancements

This chapter addresses work that arose during the research and development process of the DVCL environment. This work was necessary and essential to push our prototypical implementation closer to a productive learning environment.

The first issue addresses security: The DVCL and the DVCL with Central Authority (CA) prototypes are able to connect student's local labs, even if they are distant from each other. The system ensures, that no other hosts will be harmed by malicious network packets. The topic, that was not covered so far, is, that there is always a risk that third parties will try to unauthorized use, interfere or attack *our* system. This risk will increase by the importance of the system. If our system, meaning the DVCL client and the CA server, will no longer be used in a dedicated example group for evaluation but in a productive mode (go-live), new requirements will appear. If the system will be e.g. used to grade a student's work, it will be obvious that someone will try e.g. to influence the results. In section 4.1 we disclose, cover and resolve security issues in in the DVCL system in order to be able to use our system in a productive environment.

The second issue addresses usability: Up to now the DVCL client is designed as a tool to be used in a command line interface (CLI) environment. The CLI is a text-based input-output system provided by the operating system, where a user can interact with the system or a service by entering certain commands including parameters and receiving textual responses. The DVCL client requires providing different parameters, e.g. the IP and

the port of the DVCL's Central Authority (CA). The benefits include a flexible and easy way to add, remove and change parameters, the re-usability in other programs (batch-mode) and low development effort. A shortcoming however is, that students first have to learn utilizing the CLI including certain parameters in order to learn with the DVCL. One possibility would be to predefine values as far as possible, but this will not be feasible for certain parameters, e.g. the name of the local network. In section 4.2 we introduce a Graphical User Interface to increase the usability of the DVCL client for students.

4.1 Security

In this section, we disclose some security issues of the current DVCL prototype. While these issues occurred as minor issues during the development and testing phase, they will become major issues in a productive environment. Therefore, we decided to resolve the identified security issues using appropriate counter-measures.

In the remainder of this chapter, the terms authenticity, integrity and confidentiality are used according to [Eckert, 2013]:

- **Authenticity** means the genuineness and the credibility of an object or subject, which can be verified by using a unique identity and characteristic properties.
- **Integrity** means, that certain information cannot be manipulated unauthorized and undetected.
- **Confidentiality** means, that only authorized subjects are able to access certain information.

Use case scenario

The DVCL with CA system was designed that remote students can work together. While the students will use their own computer, the university has to provide the CA. In our use case scenario, we assume that a student uses the DVCL client outside the university (e.g. at home) utilizing an

The section 4.1: Security is based on the following publications: [Haag et al., 2017, Döhring and Kahrau, 2011]

internet connection to connect to the CA. A mobile environment using a cellular data plan is also possible.

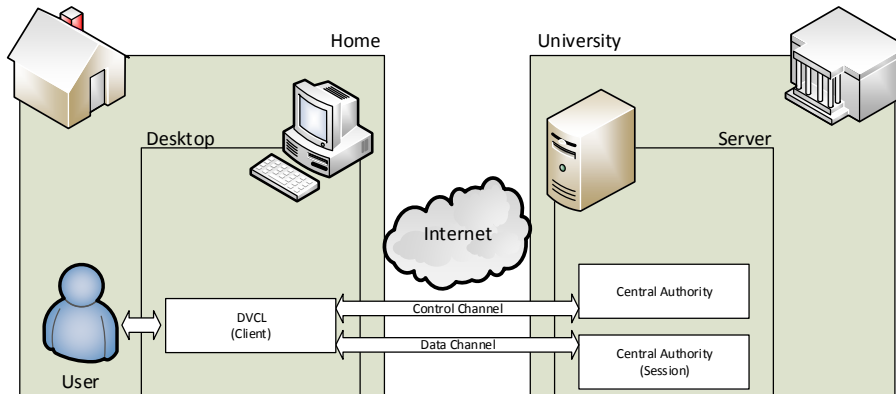


Figure 4.1: Use case diagram of the DVCL with CA

Figure 4.1 shows a corresponding use case diagram of the DVCL with CA. The left side represents the student's home, connected via internet (cloud in the middle) to the university's CA on the right side. A student, as a user of his desktop or laptop computer, utilizes the DVCL client to interact with a university's CA using a control channel connection. An additional data channel connection will be established for each remotely connected virtual network.

Security Issues

During the development, some security issues appeared. Details for each issue are listed in table 4.1 on the following page. This table first describes the security **issue**. E.g. the first issue addresses the fact, that everybody is able to use the CA if the IP address is known. This may be a desired behaviour for a general public learning environment, but the security **objective** for a university would be to limit the access, e.g. to their own students. **Affected** by this issue is the control channel which is open for everyone to connect to. An appropriate counter **measure** is to add a mechanism that users have to authenticate using a previously issued

username and password first. If the credentials are valid, the CA authorizes the user, otherwise the user will be rejected.

Table 4.1: Security Issues and Counter-Measures

No. 1	Authenticity of the user
Issue	Everybody can use the CA if the IP address is known / public. Furthermore, users or third parties can also connect to an already started session without connecting to the control channel first. As a result, it is possible to observe the network traffic of a certain session and also to inject packets.
Objective	Only certain authorized users, e.g. the participants of a network course, should be able to use the CA provided by the university. Furthermore, only authenticated users should be able to connect to a session.
Affected	Control channel on the CA's side, data channel on the CA's side.
Measure 1	The CA shall require a username and password first to access the control channel.
Measure 2	The CA shall issue an access token for authenticated users on the control channel, which will be required to use a data channel.
No. 2	Authenticity of the CA and a CA session
Issue	The source code of the DVCL with CA will be public, because everybody should be able to serve a CA.
Objective	The university's CA as well as their sessions should be authentic.
Affected	CA, CA session.
Measure 3	A verifiable certificate shall be added to the CA.
Measure 4	A verifiable certificate shall be also added to the CA sessions.

No. 3	Confidentiality of the control and session data
Issue	Data sent between the client and the CA resp. session can be read and also reused (e.g. the credentials) by third parties, if they are in a privileged position.
Objective	The data transmitted between client and CA resp. sessions should be confidential.
Affected	Control channel and data channel.
Measure 5	The data transmitted on the control channel should be encrypted.
Measure 6	The data transmitted on the data channel should also be encrypted.
No. 4	Integrity of the control and session data
Issue	Data sent between the client and the CA can be modified by third parties, if they are in a privileged position, e.g. within the same local network using a Man-in-the-middle attack.
Objective	Nobody should be able to modify commands or packets from or to a student's lab. This is essential if the DVCL with CA will be used for an exam.
Affected	Control channel and data channel
Measure 7	A verifiable signature for the transmitted data shall be added.

Security Measures

According to Table 4.1 on the preceding page, four security issues were identified. These issues can be resolved by adding (resp. implementing) the following seven measures to our DVCL client server architecture.

Measure 1: Credentials

There are already existing, common concepts to issue and verify user credentials. For a university, it could be wise to use existing directories, e.g. via Lightweight Directory Access Protocol (LDAP) or Network Information

Service (NIS), to authorize students and to keep the administration effort low. For our prototype, we decided to issue and use a combination of username and password, stored and verifiable using a text file on the CA. Once a student will connect to the CA, a valid credential will be required first. For security reasons, it is recommended for final productive systems to store the passwords hashed rather than in plain text. Equipped with a username and a password, the DVCL CA tries to find a matching combination using the database. Once a valid combination is found, the user is authenticated, otherwise rejected. We successfully added this measure to our DVCL with CA prototype.

Measure 2: Token

The control and the data channel are separated, independent channels and rely on different transport protocols (TCP and UDP). An authorized connection on the control channel does not involve the data channel. We decided to issue an access token on the control channel when a user was successfully authorized. This token will be required to authenticate on the data channel. Since the token will be issued and verified using an encrypted connection (see *Encryption*), it cannot be captured and reused by third parties.

Measure 3 and 4: Certificate

A common concept to be able to verify the authenticity of a certain host is to use a certificate based on a public key infrastructure (PKI). A PKI involves a private and a public key pair, based on strong mathematical algorithms. A message encrypted with a private key, can only be decrypted using the corresponding public key, and vice versa. For server authentication, the client uses the server's public key to encrypt a secret key. The server can get access to this secret key only if it can decrypt the data from the client with the correct private key. A common software package, that is able to deal with a PKI is Open Secure Socket Layer (OpenSSL) ²¹. Using OpenSSL, it is possible to create a private key and a certificate for the server. The key `server.key` has to be private and stored on the server

²¹OpenSSL: The Open Source toolkit for SSL/TLS, <http://www.openssl.org>, Online, accessed July 2014

while the certificate `server.crt` is public and will be used by the server to identify himself, and also by the client to verify the server's authenticity.

Using and verifying a certificate can be added with some lines of code to the DVCL environment. When the DVCL client connects to the server, he is able to verify the server's authenticity. In case of an unexpected or faked certificate, a message will be printed to the student and the connection will be aborted.

Measure 5 and 6: Encryption

A common method to use encryption in software components is to use an existing, public software library, for example the Open Secure Socket Layer (OpenSSL), which provides different symmetric and asymmetric cryptography algorithms and adjustable key sizes. The implementation of such a library into an application is rather simple, compared to a self-made library. We decided to use TCP-based Transport Layer Security (TLS) [Dierks and Allen, 1999, Dierks and Rescorla, 2006, Dierks and Rescorla, 2008] for the control channel and UDP-based Datagram Transport Layer Security (DTLS) [Rescorla and Modadugu, 2006, Rescorla and Modadugu, 2012] for the data channel.

Measure 7: Signature

The use of OpenSSL for encryption and decryption does also ensure data integrity by calculating and verifying a message digest. This digest of the message will be calculated and appended to the encrypted data before it is sent to the network. When the message arrives at the destination node, OpenSSL recalculates the digest based on the data and compares that digest to the digest appended to the message. If the values do not match, the data has been corrupted and will not be processed. This is a build-in process of the TLS²² specification.

²²RFC4346, page 3: "The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications. [...] The connection is reliable. Message transport includes a message integrity check using a keyed Message Authentication Code (MAC)"

4.2 Graphical User Interface

A way to improve the usability of the DVCL can be the introduction of a graphical user interface (GUI) to control the DVCL client. Since the students are usually more familiar with using a GUI instead of using a CLI, a lower training period can be expected. This can result in an increased acceptance level, enables more time for learning and finally can lead to an improved learning outcome. In order to reduce the training period, the GUI should support typical DVCL use-cases. Also, common misconfigurations should be prevented.

Command Line Example

To work in Netkit with networks, which are open for remote connections, two steps are necessary: First, the creation of at least one virtual host, which is connected to at least one virtual network and secondly the establishment of a connection between at least one virtual network and the DVCL Central Authority. A typical example of these steps is presented in listing 4.1 on the facing page, starting with the creation of a virtual host, which will be connected to a virtual network named *net1*. This network can be identified and referenced via the file *vhub_jens_net1.cnct*, which will be created and resides by default in a subdirectory called *.netkit/hubs*, located in the user's home directory. This identifier and additional required parameters such as IP address and port of the CA, as well as the path to the corresponding public certificate, are required to start the DVCL client. If the certificate is valid, the user has to authenticate using credentials. An invalid certificate or unknown credentials will terminate the connection. Using menu item 2 within the dialogue mode provided by the CA, a new named session can be created. Finally, the virtual network's local file handle will be connected to this session by choosing menu item 3. The local virtual network *net1* is now open for remote connections. If another student, for example a distant learning partner, also connects a local network to the same session, both networks will behave like a single broadcast domain.

The chapter 4.2: Graphical User Interface is based on the following publication: [Haag et al., 2017], and also the project output: [Döhning, 2012]

Listing 4.1: Using the DVCL via command line

```
1 # Start the client in Netkit
2 vstart client --eth0=net1

3 # Prepare the virtual network for remote connection
4 DVCLClient -i 192.168.1.25 -p 14534 -f /home/users/jens/.
   netkit/hubs/vhub_jens_net1.cnct -c server.crt

5 # Dialogue with the DVCL CA
6 Username: Jens
7 Password: ****

8 ....Menu....
9 1.) Get list of open sessions
10 2.) Create new session
11 3.) Connect to session (Port-Number needed, see 1.) or 2.))
12 4.) Delete session by port number
13 5.) End.

14 Choice: 2
15 Please provide a short description for your session: net1
16 Your created session runs with port number: 14535

17 Choice: 3
18 Connection to session on Port 14535 established.
19 █
```

Common Use Cases

Table 4.2 on the next page lists typical use cases, which can occur while utilizing Netkit and the DVCL client. The table starts with different cases on working with Netkit, followed by major cases occurring while administering the CA using the client, and closes with cases originated by the interaction of Netkit and the client, respectively a remote connection. All cases except two should be supported in a GUI. To create a host without a network connection does not make sense in a learning environment which focuses on networking practices. Connecting a local network to more than one remote sessions is technical possible but increases the risk of a circle, which can lead to a failure of all involved systems.

Table 4.2: Use Cases

No.	Case	
1	Create a host without a network connection	<input type="checkbox"/>
2	Create a host with a network connection	<input checked="" type="checkbox"/>
3	Create a host with more than one network connection	<input checked="" type="checkbox"/>
4	Login to the CA with username and password	<input checked="" type="checkbox"/>
5	List available remote sessions	<input checked="" type="checkbox"/>
6	Create a new remote session	<input checked="" type="checkbox"/>
7	Delete a remote session	<input checked="" type="checkbox"/>
8	Create a local network without a remote connection	<input checked="" type="checkbox"/>
9	Connect a local network to a remote session	<input checked="" type="checkbox"/>
10	Connect a local network to more than one remote session	<input type="checkbox"/>
11	Connect more than one local network to one remote session	<input checked="" type="checkbox"/>
12	Disconnect a local network from a remotely connected session	<input checked="" type="checkbox"/>

Explanation:

- ⊗ This case should be supported via GUI.
- This case should not be supported via GUI.

In addition to these common cases, this GUI should have the ability to predefine certain required parameters, e.g. the IP address of the CA. Nevertheless, these parameters should be still editable. Moreover, the CLI client should not be replaced by the GUI but should be used optionally, for example in custom scripts.

A GUI for the DVCL Client

This section introduces a developed example GUI for the DVCL client, which supports the use cases listed in table 4.2. These GUI can be started as usual by clicking on a program icon. This can also be an automatic process within the DVCL environment using the autostart feature of the underlying operating system. The following screenshots within this chapter

were captured using an Ubuntu Linux distribution. The look, fonts and the colours were set by the Ubuntu theme and will look different on other desktop environments.

At first, a logon screen will appear as shown in figure 4.2. This screen is responsible to support case no. 4. The address of the CA and also the path to the certificate are predefined but still editable. Since the certificate is stored as a text file, the related path can be entered directly or the button “Browse” can be used to open a file requester. Also, a username and a password have to be entered in order to continue. The button “Reset” will set back all entries to their default values. The button “Login” establishes a connection to the given CA and sends username and password for verification. The certificate will also be validated. If one of these processes fails, the connection will be aborted and the GUI will report an error.

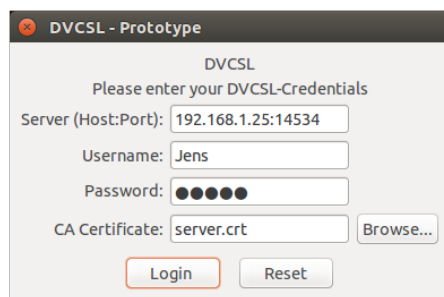


Figure 4.2: GUI login

If the credentials and also the certificate are successfully validated, the main screen will appear as illustrated in 4.3 on the next page. This screen consists of three essential areas: The **local area** on the left, the **remote area** on the right and the **status area** at the bottom.

The **local area** can be used to administrate local Netkit hosts and networks and thus is responsible for case no. 2 and 3. Using the button “Create virtual Host/Network” will present a dialog shown in figure 4.4 on page 57. This dialogue can be used to create a new virtual host connected to at least one virtual network. A hostname and at least one named network are required, fulfilling case no. 1. The button “+” can be used to add and name additional networks, for example to create a router.

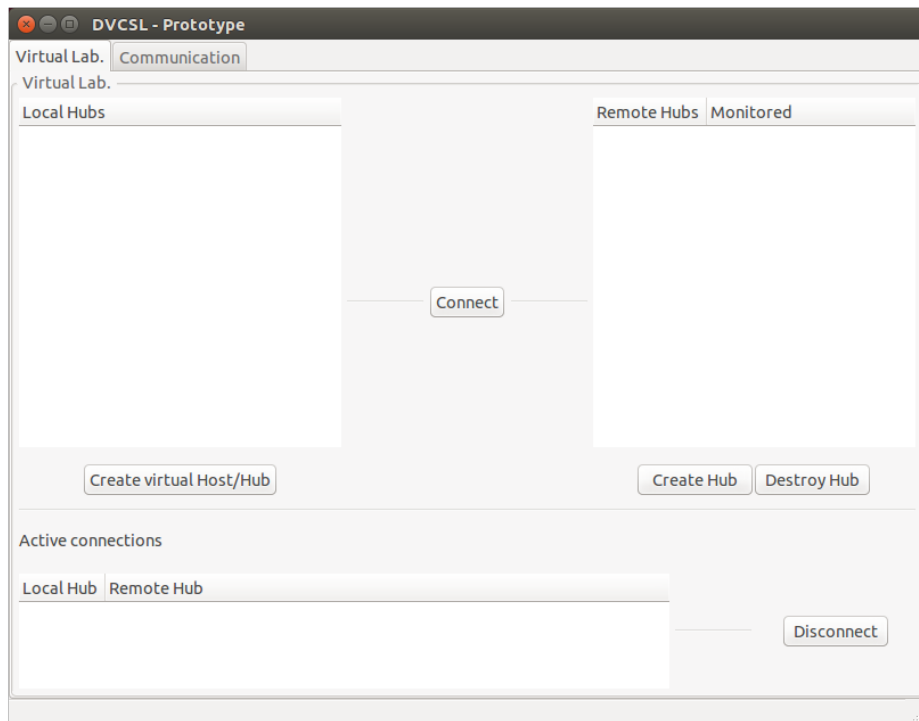


Figure 4.3: GUI main view

The button “Cancel” will close this dialogue, the button “OK” creates a Netkit host according to the settings of the dialogue. This host will run in a new window, this is identical to a start via CLI (e.g. `vstart client --eth0=net1`). The new network(s) will appear in a list at the left area. This list also captures virtual networks, which are not started using the GUI but the CLI. We utilize a function of the linux kernel called *inotify* to observe a certain directory and to detect, if something has changed. If a Netkit network was started, a file handle will be placed in a certain preconfigured directory. This file handle will be used by the virtual hosts to connect to a certain virtual network.

An example screen can be found in figure 4.6 on page 58.

The **remote area** can be used to administrate sessions of the CA. In the GUI, the term *session* was mapped to the closer matching network

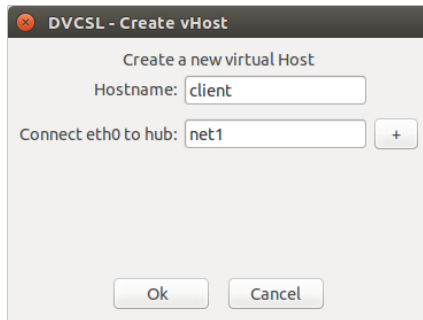


Figure 4.4: GUI to create a new Netkit host

term *Hub*. Already existing sessions will be listed directly after the login, according to case no. 5. A new session can be created according to case no. 6 using the button “Create Hub”. This leads to a new dialogue, which is presented in figure 4.5. The session first requires a name. Additionally, two features are planned but still not implemented: The creator of the session as the “owner” should be able to set a password. Other participants, e.g. the learning group members, have to know and also have to enter this password in order to connect to this session. Also, the invitation of an electronic exercise assistant (see chapter 6: Electronic Exercise Assistant), e.g. to guide certain exercises, is planned. The button “Cancel” will close this dialogue without any results, “OK” will create a new remote session on the CA. This session will appear immediately in the remote area. An example screen can also be found in figure 4.6.

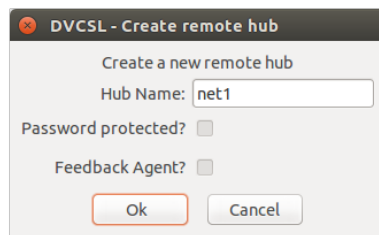


Figure 4.5: GUI to create a new remote hub

Local networks can stay local, according to case no. 8. A local network

can also be connected to a remote session (case no. 9), by using the button “Connect”. This requires to select a previously created network in the local area and an existing session in the remote area first. An established connection between a local network and a remote session will be displayed in the **status area**. This connection can be terminated by selecting the related entry and using the button “Disconnect” (case no. 12). It is possible to connect more than one local network to the same remote session (case no. 11). An additional connection attempt from an already connected local network will be aborted and a message will pop up (case no. 10).

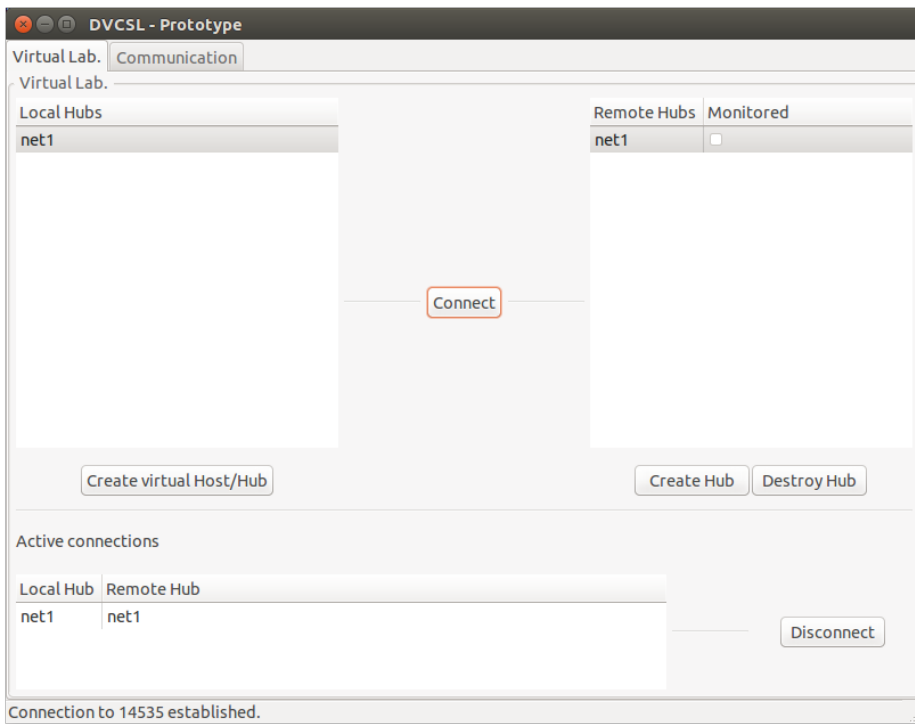


Figure 4.6: GUI showing a connected network

4.3 Conclusion

In section 4.1 security issues in our DVCL environment were presented and each of them can lead to a threat. This is why we added counter-measures to cover each security issue. Summarized, these measures are targeted

- to get authentic users,
- to get confidential and unmodified data between the DVCL client and the CA server,
- to provide a verifiable authentic CA and
- to get also authentic users within a DVCL session.

These new gained properties of our DVCL environment express a significant step towards a productive learning environment. Nevertheless, the process of identifying security issues followed by adding counter-measures is a continuous process where the system including the applied measures needs to be reviewed again. Furthermore, we used a generic scenario which does not cover special situations, for example incorporating certain examination rules. These situations may require a stronger security setting and thus again a security review.

In section 4.2 we showed that the developed GUI is an example on how essential interactions with the DVCL can be graphically supported and simplified. Since command line parameters are not required, the training period for students can be shortened and they can spend more time working on the exercises. Moreover, we expect a higher acceptance level and more students willing to use our DVCL environment for learning.

We also created a possibility that students can intercommunicate within the GUI in a so-called lobby. An example can be viewed in figure 4.7 on the next page. Using the lobby, students can write text messages which can be read by other participants in realtime. The primary goal is that connected DVCL users can arrange and coordinate themselves on their own without the need of third party tools, e.g. phone, Skype, Whatsapp. A further development of this lobby in the form of a virtual classroom can be found in chapter 7.

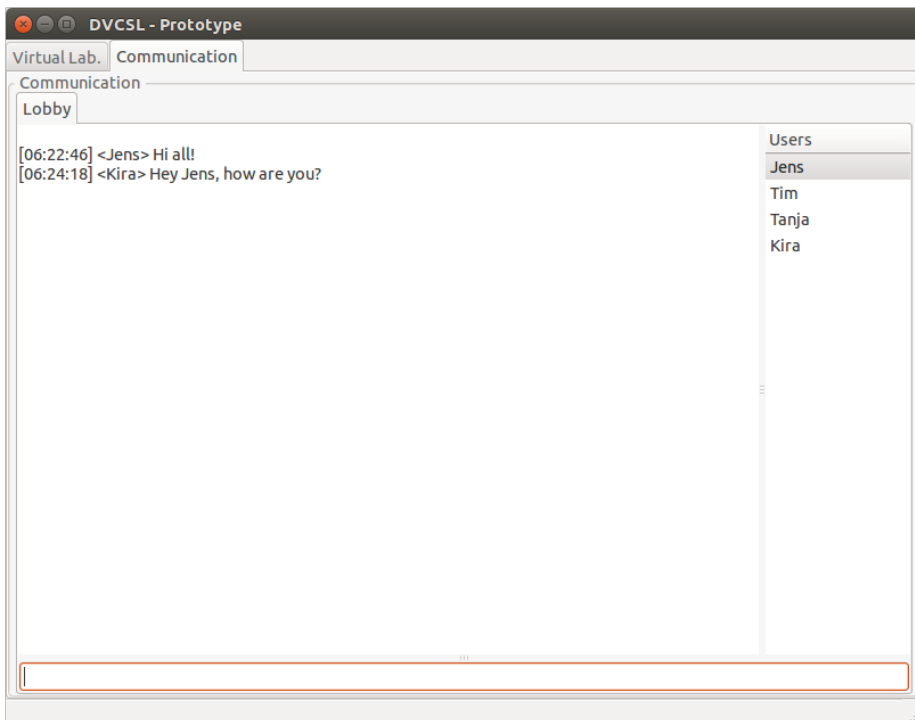


Figure 4.7: Lobby Chat

Part II

**Educational Aspects of
DVCL**

Chapter 5

Course Evaluation

Computer science curricula for students at universities nowadays include courses on networking and information technology (IT) security. Teaching theory on networking and IT security is usually done by means of textbooks and classes (either face-to-face classes or virtual classes, which are popular at universities for distance education). To anchor and deepen the acquired theoretical knowledge, a commonly used teaching method is to hand out practical assignments. While solving the assignments can be voluntary, a mandatory practical course can be used to guide the students at the university while working on the assignments and to finally verify if the students have successfully applied their theoretical knowledge.

In a steadily changing world with new technologies, growing mobile connectivity and everytime-everywhere character it seems natural that these changes may impact the students' needs and requirements regarding their learning environment. With respect to students working on assignments in a physical lab at a traditional university, we expect a growing demand for computer supported learning environments. The obvious question is whether our aforementioned concepts and prototypes fit into and can improve upon a classic on-campus learning environment. A way for educational staff to answer this question is to interrogate the students about their opinions (acceptance research).

In the winter semester 2012, we evaluated a practical course at the

This chapter is based on the following publication: [Haag et al., 2013, Haag et al., 2014a]

Cologne University of Applied Sciences, where students have to work out and solve assignments. They also have to defend their solutions. A special property of this course is that the students are given a high degree of flexibility during their assignment preparation time. We wanted to discover the learning behaviour of students who are free to choose their learning environment in order to successfully complete the course. In addition, we wanted to determine the success of the course. The results will be used as a motivation for future work with respect to an optimized alignment of our technical implementation and practical course concepts in place.

5.1 Learning Situation and Environment

All 3rd semester students of the Bachelor programs at the department of computer science at the Cologne University of Applied Sciences in Germany have to take part in the course “Communication technology and networks (*German: Kommunikationstechnik und Netze*)”, where they learn about concepts and standards of computer networks, hosts and intercommunications. The course consists of two hours of weekly lectures, accompanied and supplemented by one additional hour, where students work on their assignments in a practical course (*in German: Praktikum*). Students are required to pass the practical course as a prerequisite to take the exam. According to the curriculum, the practical course’s outcome is either pass or fail while the exam is being graded. It is planned but not mandatory to participate in the lecture and practical course within the same semester.

The practical course is organized as follows: Students have to register to take part. In a kick off meeting they will get to know the course advisors, the assignments and the computer laboratory. The course advisors are members of the academic staff at the university, have expert knowledge about the course content and are able to support and guide the students. The assignments are related to certain theoretical concepts of the lecture and facilitate that students have to apply their previously acquired knowledge, e.g. by setting up and configuring real world networking scenarios or by analysing network traffic.

After the kick off meeting the students have to work out and solve the assignments within a specified time (currently 3 weeks). Students were told

that they are free to choose their learning environment; the assignments were prepared not to require any special setting. Students can work e.g. alone, in a learning group, at home, at the lab or in any combination. No matter what specific learning environment they end up choosing, they work on the same assignments. The students that choose to work in groups are free in forming groups of any size and organize the group and work as they please. Students were also told to be able to get support in guided learning hours, which are regularly offered by the course advisors. The students are given this flexibility, so that they can choose the learning environment that best suits their respective learning.

The aim of the practical course is to make sure that every student has learned the concepts related to the assignment, has an understanding of the solution and is able to reproduce and defend the solution. To successfully complete the course each student has to demonstrate and defend the solved assignment in a final bilateral expert talk with a course advisor. The course advisor knows the solution and possible ways of solving. He is able to judge whether a student has successfully acquired and applied theoretical concepts of the lecture.

5.2 Networking Assignment Example

Most of today's intra- and internetworking were set up and configured in a decentralized manner by several different IT network administrators. This decentralized approach works, because that interconnection of computer hosts is based on commonly accepted and applied standards. In our practical course, the students were given real world assignments in order to practice the application of the aforementioned networking standards and the configuration of modern networks. One such assignment taken from the examined course is:

“Set up and configure a scenario with at least four hosts (e.g. client, router1, router2, server) to demonstrate routing behaviour. The client and the server should be located in different networks. The client should be able to intercommunicate with the server by using the intermediate router1 and router2.”

The minimal requirement for this setup is shown in Figure 5.1, consisting of at least four hosts. The client and the server have one network interface; the routers are equipped with two network interfaces.

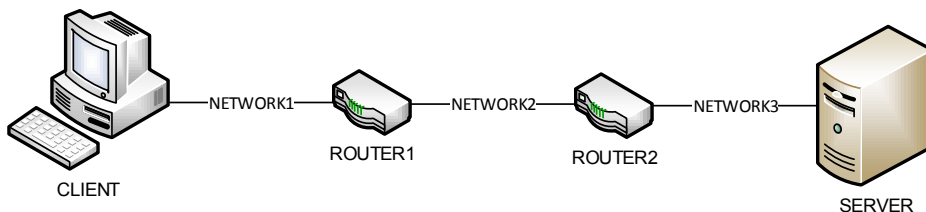


Figure 5.1: Example network setup

A valid and straightforward solution for this networking assignment example solved in the virtual environment Netkit is stated in listing 5.1.

In order to accomplish the given task, students will have to start four virtual hosts and interconnect them accordingly within three virtual networks. They will then have to assign appropriate addresses to these networks and hosts and ultimately configure the routing on each host. Once the network is configured properly, students can demonstrate the validity of their solution by sending network packets between client and server and by using a suitable tool (e.g. `tcpdump`²³) to analyse the packet flow. Students have to assure that the packets match their expectations based on the aforementioned standards.

After working through this exercise, the students have learned about network classes, the OSI layer model, the concept of routing, basic network configuration tools in Unix-based systems, routing tables and their manipulation and tools to analyse network packets and network behaviour.

5.3 Evaluation

In the winter semester 2012, 249 students signed up for the practical course “Communication technology and networks”. 178 of them (71%) participated in our evaluation process. The students were divided into smaller groups

²³`tcpdump`: a powerful command-line packet analyser. <http://www.tcpdump.org>, Online, accessed December 2017

Listing 5.1: Example Solution

```
1 // Create the virtual environment
2 vstart client --eth0=network1
3 vstart router1 --eth0=network1 --eth1=network2
4 vstart router2 --eth0=network2 --eth2=network3
5 vstart server --eth0=network3

6 // Configure the client
7 ifconfig eth0 150.0.0.1 up
8 route add default gw 150.0.0.2

9 // Configure router1
10 ifconfig eth0 150.0.0.2 up
11 ifconfig eth1 160.0.0.1 up
12 route add -net 170.0.0.0 netmask 255.255.0.0 gateway
    160.0.0.2

13 // Configure router2
14 ifconfig eth0 160.0.0.2 up
15 ifconfig eth1 170.0.0.1 up
16 route add -net 150.0.0.0 netmask 255.255.0.0 gateway
    160.0.0.1

17 // Configure the server
18 ifconfig eth0 170.0.0.2 up
19 route add default gw 170.0.0.1
```

with reserved timeslots to get an evenly distributed utilization of the laboratory. These groups were created for organizational purpose only and were not related to any didactic concept. Three course advisors were working at the laboratory which provides 15 computer workstations for the students.

We prepared and distributed a free and anonymous questionnaire at the end of the winter semester, after the students completed their last expert talk. The aim of this summative evaluation was to evaluate the learning behaviour of the students when they work on the assignments, and the success of the practical networking course. Our major motivation was to determine whether the learning situation and environment in our

practical course meet the needs of the students and to get findings about key factors with respect to possible improvements to the course. For that purpose, questions were designed to interrogate about different parameters, which should give us an answer to the following questions:

Q1: Which learning environment did the students choose with respect to learning location, learning method and guided learning, and which environment would they prefer with respect to form of education?

Q2: What is the objective and subjective success of the course?

For *Q1* we designed 6 questions to interrogate the students. The appropriate evaluation can be found in the following sections A to D. To assess the course success (*Q2*) we designed 3 questions for the students in order to interrogate their subjective course success. In addition, we utilize the course's list of passed and failed students in order to get an objective success rate. The appropriate evaluation can be found in section E.

The questions were designed to interrogate students in the actual course. The results may be biased, e.g. by student's previous experiences on e-learning or group work. Nevertheless, with a sample size of over 200 students we expect a convincing result.

A. Learning Location

The students were able to work on the assignments without the need for own equipment. The university provided all resources needed (e.g. computer lab with host and network infrastructure, staff, schedule) so that students could work according to their fixed lesson plan. In addition to this, the assignments were prepared to be solved without the need to be at the lab physically by using virtualization technology. The aim was to give students as much flexibility as possible preparing the assignments. We wanted to know how this flexibility was accepted by the students. We therefore asked them: "Did you work primarily at home, in the laboratory, or in a combination of the two?"

Figure 5.2 on the facing page shows that with 51% a little more than half of the students worked in the laboratory almost exclusively, 22%

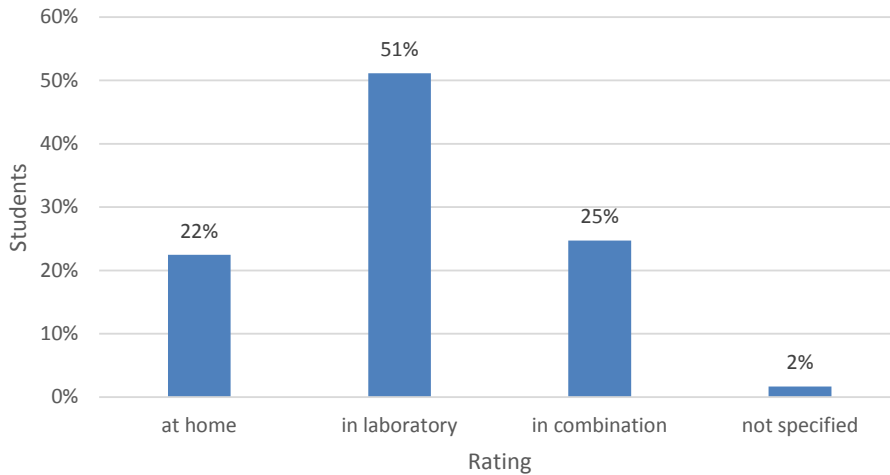


Figure 5.2: Learning Location

worked only at home and 25% used a combined approach. This shows that 76% of students rely on utilizing the university's resources by working in the laboratory, while 47% of our students accept the offer to work at home independently from the lab, at least partially.

B. Learning Method

The students were free to choose whether to learn and work on their own or in a group setting. Given that they had to defend their solution individually, we wanted to know which learning method they chose for preparing the assignments. Our question was "Did you work primarily alone, in a group, or a combination of the two?"

Figure 5.3 on the next page shows that 16% learned on their own, 66% learned in a group setting, and 17% used a combination of both approaches. This adds up to 83% of our students for whom a group setting is an important part of their learning process, despite being tested individually.

C. Guided Learning

The students had the opportunity to get support when working on their assignments by means of getting help and guidance from the course advisors.

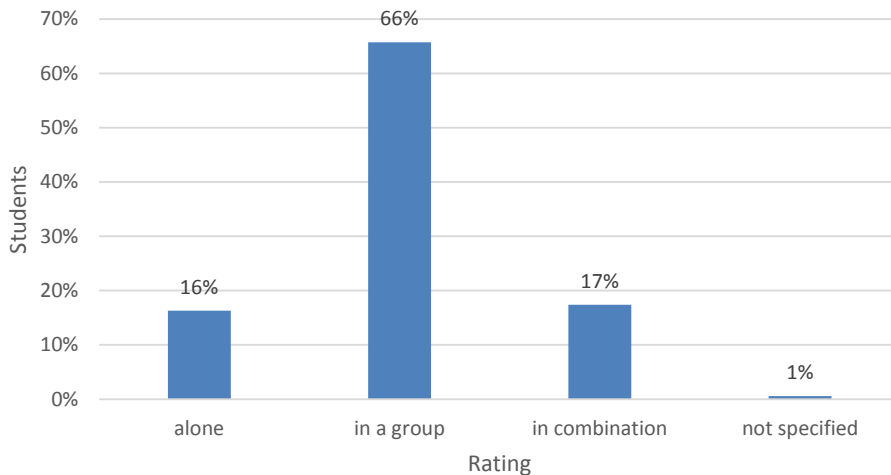


Figure 5.3: Learning Method

Guided learning hours were scheduled for this, which were open to all of the students. We did so to enable all our students that needed help to ask for and to receive it, and they were made voluntary, so that not every student needed to commute to the lab, i.e. the university, despite not needing any guidance. We wanted to know how much of a demand for these hours existed among the students, and how these hours were perceived by them. First, we asked them: “How often have you utilized the guided learning hours?”

Figure 5.4 on the facing page shows, that there was an even distribution of students who felt they needed a lot of help, students that needed very little help or no help at all, and students that fell just in between. That shows that these guided learning hours are very valuable to a lot of students, and that their existence seems vital to the course design. It also shows that a voluntary nature is very reasonable when it comes to guided learning hours. Next, we wanted to know how students assess the availability of the course advisors in order to verify students are indeed able to get support if they need to. In our evaluation, we asked the students: “How do you rate the availability of the course advisors?”

While most of the students (85%) were being able to get support if they needed to, nobody was left alone (see figure 5.5 on the next page).

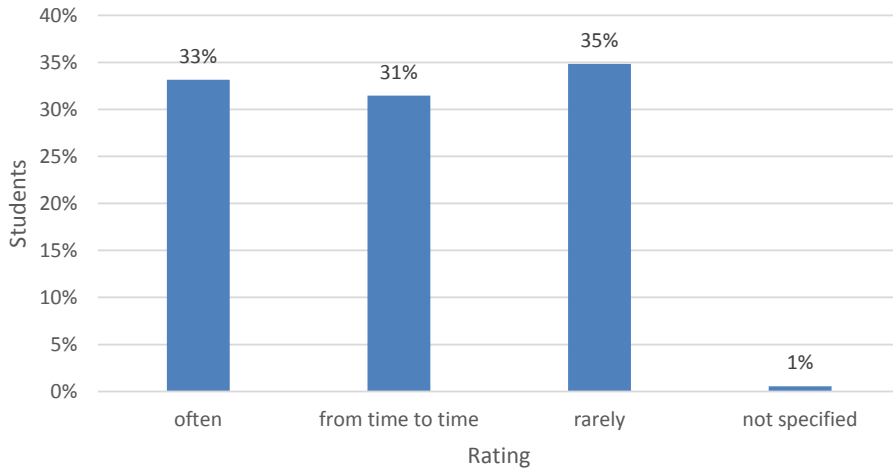


Figure 5.4: Utilization of the guided learning hours

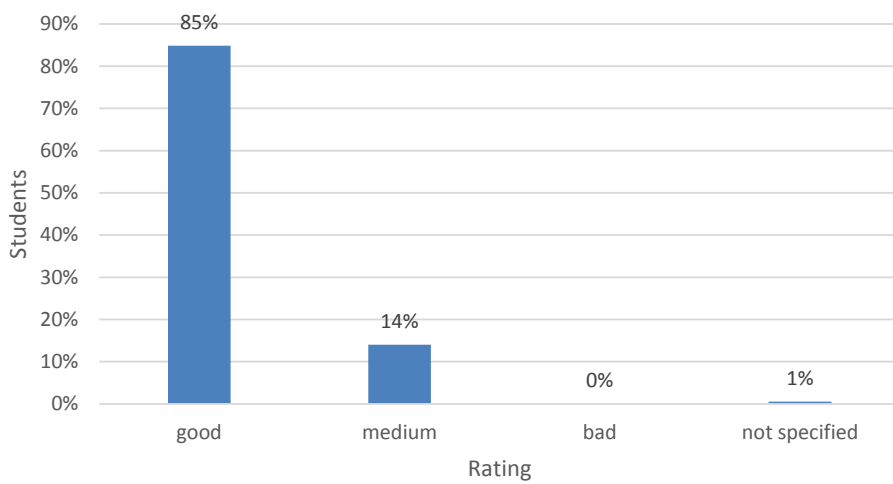


Figure 5.5: Availability of the course advisors

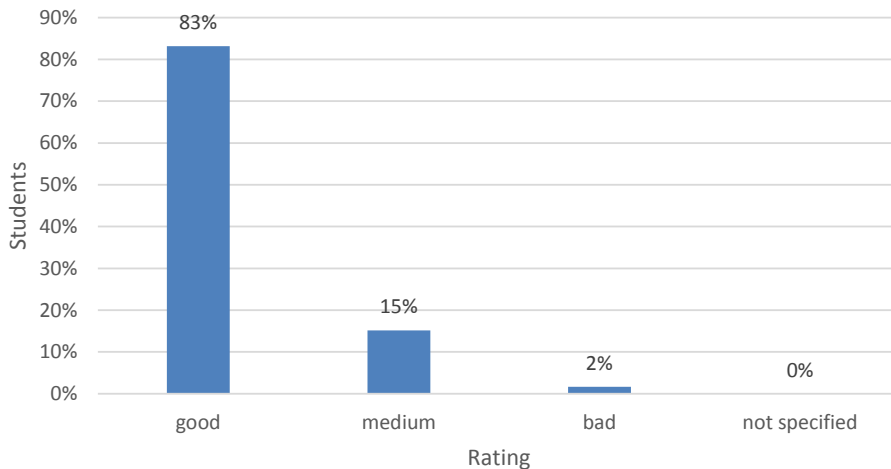


Figure 5.6: Quality of support

Next, we asked the students: “How do you rate the quality of support during the guided learning hours?”

Figure 5.6 shows that the vast majority felt that the support’s quality was very good, and only 2% thought poorly of it.

D. Preferred Style of Education

With respect to the blended learning approach, where conventional classroom settings and e-learning will be combined, we wanted to know what form of education the students would prefer. We asked: “In which of the following scenarios would you expect your learning success to be highest: working online with an e-learning system, working in the lab in a face-to-face setting, or in a combination of the two?”

According to figure 5.7 on page 74, 8% of the students would prefer to learn exclusively in an e-learning environment, 43% prefer to learn exclusively in a conventional face-to-face classroom setting and 41% of the students think it would be beneficial to combine these two approaches. While this shows that the majority (84%) of the students think of a classical face-to-face environment as essential to their learning process, 49% would welcome the introduction of an additional e-learning environment.

E. Success Rate

The objective success rate of the course was 77%. Out of 249 registered students, 191 students passed, i.e. they worked out the assignments, and demonstrated and defended their solutions successfully. Most of the 23% unsuccessful students registered but did not participate at all; some seemed to have other shortcomings; a few tried but could not defend their solution properly. Because of the anonymous nature of our evaluation and because the practical course is not graded but has a result of either passed or failed, we cannot deduce whether or how the success rates differ among different groups of students. We also wanted to know what students thought about their subjective learning success. We first asked them: “How would you rate your own knowledge acquisition with respect to the practical course?”

According to Figure 5.8 on the following page, a majority of 78% rated their personal knowledge acquisition as high. Next, we wanted to know what skills students had before the course. We asked them: “How would you assess your level of familiarity with the course’s contents prior to the course?”

The evaluation in Figure 5.9 on page 75 shows that 21% of the students assess their own familiarity with the course content as high, 25% as medium and nearly half of the students assess their pre-course knowledge as low. In a third question, we asked the students: “How do you rate the difficulty level of this practical course?”

Figure 5.10 on page 75 shows that a majority of the students (70%) assessed the difficulty level of the practical course as medium. Only 9% found it easy and only 13% struggled and found the assignments hard to solve.

5.4 Conclusion

With a participation rate of over 70% of 249 students in our evaluation we acquired a representative sample of the learning behaviour in a practical networking course. With respect to our second research question Q2 we found out that we obviously observed an objectively successful course, as participants had a success rate of over 75% in the winter semester 2012. The flexibility granted to the students in preparing their assignments was generally well accepted and the students utilized all possibilities to varying

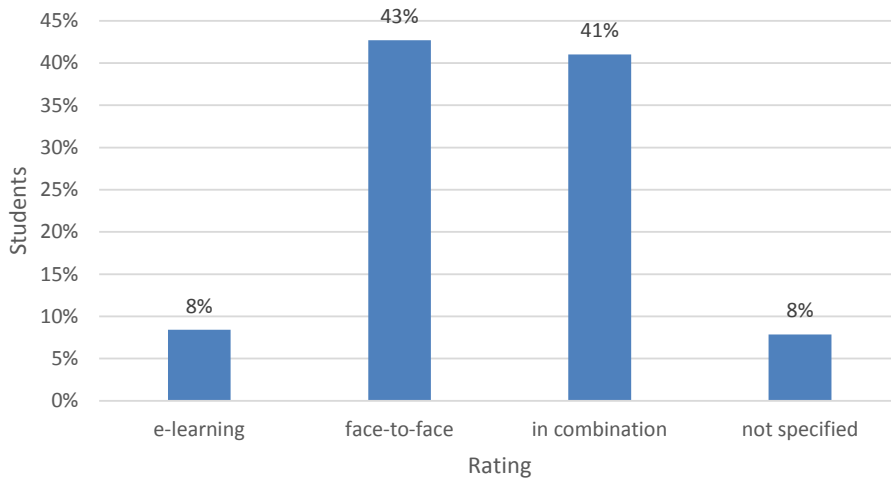


Figure 5.7: Style of education

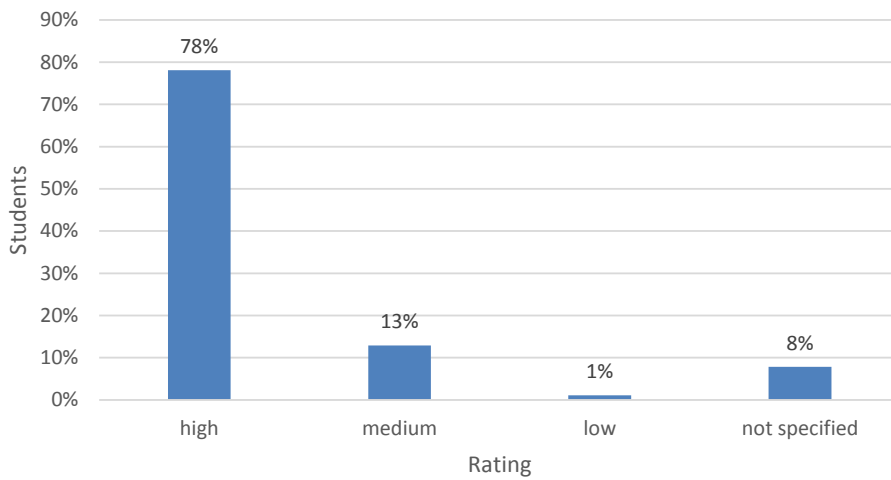


Figure 5.8: Knowledge acquisition

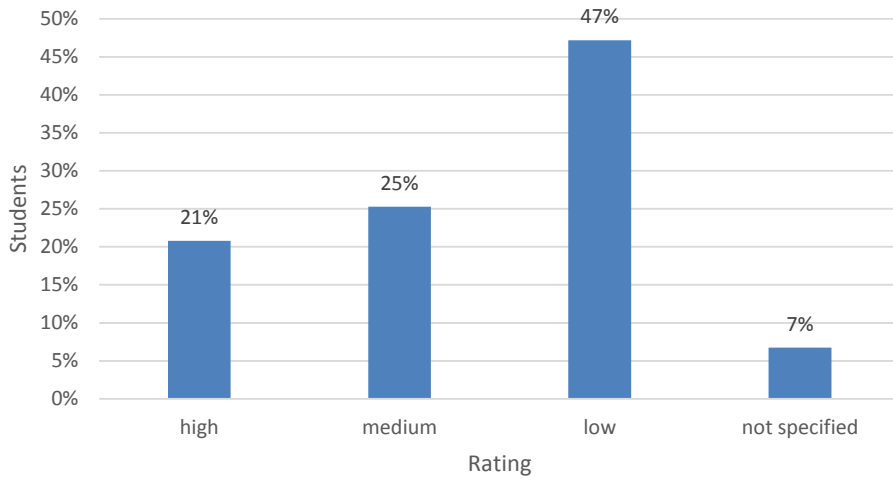


Figure 5.9: Preknowledge

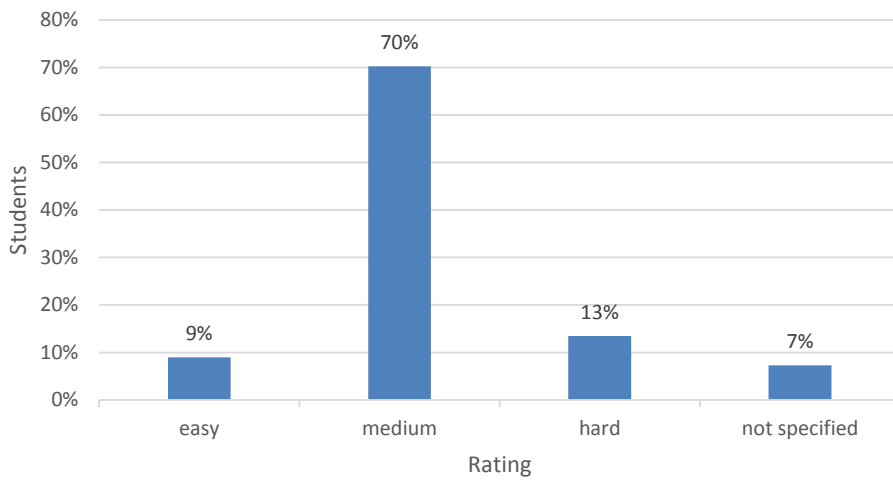


Figure 5.10: Difficulty level of the course

degrees. Without being recommended certain learning environments or a certain learning behaviour, the students seemingly chose what they deemed best for them personally, resulting in a high success rate of the course. We were also able to find results regarding our research question Q1. While the preferences for learning environments and behaviour were fairly distributed, a predominant majority of the students thought of working in groups as well as receiving guidance and feedback as crucial to their learning success. Students are also interested in new and modernized learning environments. It is important that these new environments do not replace more traditional ones, but rather add on to them.

One way of modernizing the practical course would be the introduction of an e-learning system, which would be explicitly welcomed by 49% of the students. In addition to that, nearly half of the students said that they would like to work independently from the lab at least partially, which they would be enabled to do by the introduction of such a system.

Given the students' preference for group working and guided learning, one should take these two key factors into account when introducing an e-learning system. This means that, in order to gain the students' acceptance, an e-learning system should enable collaboration as well as guidance, and shouldn't be limited to simply providing an environment for solving assignments online.

Chapter 6

Electronic Exercise Assistant

Recent evaluation shows that students of a traditional on-campus networking course deem it crucial for their learning success to be able to get support from a course advisor (see chapter 5). While an on-campus university will be able to provide course advisors, which can support students in so-called guided learning hours, this support is no longer feasible if students work e.g. at home in the evening hours using a virtual lab.

A common way to resolve this issue is an intelligent tutoring systems (ITS). An intelligent tutoring system is a computer system that aims to provide instruction or feedback to learners, usually without requiring intervention from a human advisor [Corbett et al., 1997, Psotka et al., 1988]. Literature reports many scopes where tutoring systems were developed or applied, e.g. in the scope of teaching mathematics [Heeren et al., 2008, Gerdes et al., 2010, Melis et al., 2004, Canfield, 2001], databases [Yang, 2011, Kenny and Pahl, 2005, Suraweera and Mitrovic, 2004], programming languages [Queirós and Leal, 2012] like JAVA [Sykes and Franek, 2003, Vesin et al., 2013] or Haskell [Jeuring et al., 2014], IT security [Hu et al., 2003, Mahdi et al., 2016, Hu et al., 2004], and physics [VanLehn et al., 2005, Albacete and VanLehn, 2000].

In this chapter, we introduce a tutoring system called Electronic Exer-

This chapter is based on the following publications: [Haag et al., 2012, Alferts, 2013, Haag et al., 2014b]

cise Assistant (EEA) for our DVCL environment. The aim of this EEA is to offer support and guidance for students while working out an exercise, even if a human course advisor is not available.

This section is divided into six sections. In the first section, we start by showing a typical network example exercise and an appropriate possible solution using our VCSL, followed by operating principles of a human course advisor with respect to this exercise in the second section. In the third section, we introduce our concept of an Electronic Exercise Assistant (EEA) and we demonstrate the technical feasibility: Our EEA is able to detect a student's lab setup based on a given exercise and observed network traffic. In the following fourth section, we show the educational feasibility. We introduce our concept to model exercises in order to be able to verify and also to guide different exercises. The fifth section introduces the architecture of our Electronic Exercise Assistant followed and concluded by a complete guidance example in section six.

6.1 A Typical Exercise Example

A very common task for future IT network administrators is to setup, configure and probably secure a network. Therefore, an example assignment for students participating in a networking course could be:

“Setup and configure a scenario with at least three hosts (client, router, server). Client and server should be located within different subnets. The client should be able to intercommunicate with the server by using the intermediate router. The routing should be based on static routing tables.”

The minimal requirement for this setup is shown in Figure 6.1, consisting of at least three hosts. The client and the server have one network interface; the router is equipped with two network interfaces: one interface connects to a network with the server, the other one with the client.

In this example exercise, students will have to set up hosts and interconnect them accordingly within two different networks. They will then have to assign appropriate IP addresses to these hosts and ultimately configure the routing by altering the routing tables on the hosts. Once the setup is

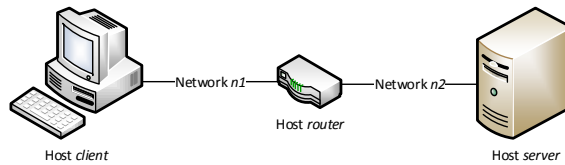


Figure 6.1: Example Network Setup

configured properly, students can demonstrate the validity of their solution, e.g. by sending network packets between client and server.

While setting up and configuring this scenario, the students can gather several basic and advanced practical experiences, like designing, preparing and setting up an example network scenario according to the assignment, configuring the hosts and network interfaces, configuring the routing and discovering different behaviour caused by different configurations.

Solve the Exercise

To solve this exercise, students need to know at least the basic commands to start and to administrate the virtual lab (Netkit). In addition, they have to apply their networking knowledge to configure the hosts according to the assignment. Using the VCSL, a valid and straightforward configuration to solve the example assignment may look like stated in listing 6.1 on the next page.

6.2 Performance of a Human Course Advisor

While working on this assignment, the human course advisor can guide the students in several ways by asking, checking or discussing questions like:

- How should the final setup look like?
- How many hosts are already there?
- Do we have at least two networks?
- Do we have appropriate IPs?
- Can the client and the server intercommunicate with the router?

Listing 6.1: Valid solution using Netkit

```
1 // Create the hosts and networks in Netkit
2 vstart client --eth0=n1
3 vstart router --eth0=n1 --eth1=n2
4 vstart server --eth0=n2

5 // Assign IP address on the client
6 ifconfig eth0 10.0.0.1 up

7 // Assign IP address on the router
8 ifconfig eth0 10.0.0.2 up
9 ifconfig eth1 11.0.0.2 up

10 // Assign IP address on the server
11 ifconfig eth0 11.0.0.1 up

12 // Set default gateway on the client
13 route add default gw 10.0.0.2

14 // Set default gateway on the server
15 route add default gw 11.0.0.2

16 // Connection test on client to the server
17 ping 11.0.0.1
```

- Is the gateway correctly set?
- Can the client communicate with the server and vice versa?

Finally, the course advisor will check the setup to verify that the students' work fulfils the requirements of the assignment. Possible checks could be:

- Do we have at least three hosts and two networks?
- Do we have a router between two hosts?
- Does the routing work?

The course advisor knows different, well-known ways to detect routing functionality within a network. In addition, he needs access to the students'

setup. If the network architecture is known, a simple `ping`²⁴ command between two hosts located in different subnets shows if they can reach each other, ergo the routing works or not. If the command `traceroute`²⁵ reports one or more hosts between the source and destination host, the routing should also work. The tools `ping` and `traceroute` are powerful tools often used by network administrators in real world scenarios to analyse a network. They use special techniques on the network layer to determine the network behaviour and will provide their results to the administrator. A shortcoming however is that these tools are connected neither to an assignment nor to a desired setup or network behaviour in any way. Choosing suitable tools and checking a student's setup is usually based on the course advisor's expert knowledge.

6.3 Technical Feasibility

One aspect to be considered when designing an Electronic Exercise Assistant (EEA) is the technical feasibility. One initial question is:

Is it possible to check the students' VCL setup by an EEA with respect to an existing exercise?

In the following we show that our EEA prototype is able to check a VCL setup within the scope of the previously introduced network example exercise in chapter 6.1. Strictly speaking, the EEA can detect if routing occurs in a given Netkit setup.

²⁴RFC1208, A Glossary of Networking Terms, page 13: "ping: Packet internet groper. A program used to test reachability of destinations by sending them an ICMP echo request and waiting for a reply."

²⁵RFC1393, Traceroute Using an IP Option, page 2: "The existing traceroute operates by sending out a packet with a Time To Live (TTL) of 1. The first hop then sends back an ICMP error message indicating that the packet could not be forwarded because the TTL expired. The packet is then resent with a TTL of 2, and the second hop returns the TTL expired. This process continues until the destination is reached. The purpose behind this is to record the source of each ICMP TTL exceeded message to provide a trace of the path the packet took to reach the destination."

Concept

When students work out the networking example exercise, network packets will occur. A big benefit of the VCL is that all started virtual networks are located on the local computer and can therefore be accessed and observed. While the course advisor uses tools at a specific point of time (e.g. when the student thinks that he finished the exercise), the EEA will be able to constantly observe the network packet flow. In the next step, we try to model the knowledge, which is used by the course advisor to check the student's setup, to be also processed by the ECA. An example to determine routing behaviour within a network is:

“Routing occurs if an OSI layer 3 IP transmission of a network packet between two hosts is based on more than one OSI layer 2 transmissions”.

Major parts of the course advisor's knowledge could be represented as rules of this kind. Finally, the EEA should be able to apply this knowledge to the observed network packets to determine whether desired network behaviour occurs (e.g. routed network packets).

The communication concept of the EEA is shown in Figure 6.2. We created a new software program which consists of two major processes:

The first process automatically captures all network packets from all local virtual networks of a student's VCL and stores them into a database. This capturing will be done by using ghost hosts. A ghost host is an additional virtual host connected to a virtual network but is completely transparent to other hosts and therefore does not interfere with the student's setup. The virtual network with hub-like behaviour guarantees that network data is distributed to all other locally connected virtual hosts, even to the ghost host. A ghost host will be attached to every started virtual network. Storing the network packets will be done by using a SQLite²⁶ database. SQLite is a relational database and can be managed by using standard SQL commands. A big benefit is that a SQLite database does not need an additional database server but can be

²⁶SQLite: "SQLite is a software library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine." <http://www.sqlite.org>, Online, accessed April 2015.

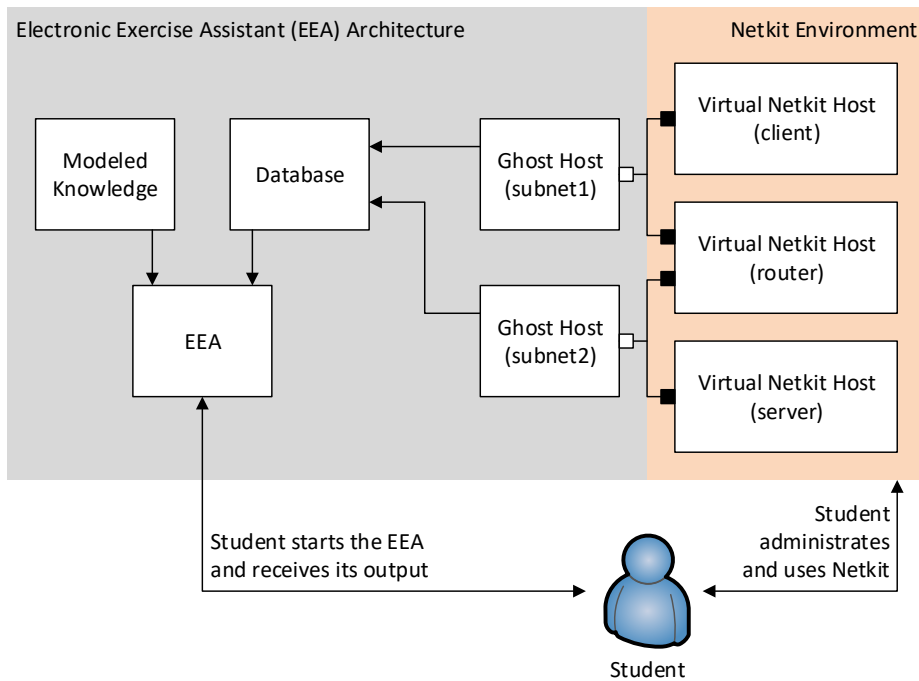


Figure 6.2: Architecture of the EEA interacting with a student and Netkit

embedded completely into the EEA. In addition, SQLite is open source software and available for free. An entry of the EEA’s database contains the raw network packet, a network identifier which represents the virtual network origin and several extracted packet header fields like MAC and IP addresses.

The second process runs parallel to the first process and is able to continuously apply the knowledge. In future concepts knowledge should be represented by ontologies or grammars on an additional implementation layer. In our exemplary implementation, we decided to use a canonical approach by directly transforming rules to SQL queries on the database containing recorded network packets. In particular, we modelled the routing statement by using only one SQL query (see listing 6.2 on the following page) to prove this statement.

The result of this SQL query is either “YES” or “NO”, depending on the

Listing 6.2: SQL query to detect routing

```
1 -- Subquery (3)
2 SELECT
3 CASE WHEN COUNT(HOPS) = 0 THEN 'NO' ELSE 'YES' END
4 AS ROUTING_DETECTED
5 FROM
6 (
7 -- Subquery (2)
8 SELECT IPv4_SOURCE, IPv4_DESTINATION,
9 COUNT(SOURCE) as HOPS
10 FROM
11 (
12 -- Subquery (1)
13 SELECT IPv4_SOURCE, SOURCE, IPv4_DESTINATION
14 FROM ETHERNET
15 WHERE IPv4_SOURCE NOTNULL
16 AND IPv4_DESTINATION NOTNULL
17 GROUP BY SOURCE, IPv4_SOURCE,
18 DESTINATION, IPv4_DESTINATION
19 -- Closing Subquery (2)
20 )
21 GROUP BY IPv4_SOURCE, IPv4_DESTINATION
22 -- Closing Subquery (3)
23 )
24 WHERE HOPS > 1;
```

stored packets matching the routing statement. The first query (1) gathers all OSI layer 2 and 3 connections between hosts excluding broadcasts and non-IP traffic like ARP. Based on the result, the second query (2) counts the MAC tuples for each IP connection. Based on this result, the third query (3) will report “YES” if the second query found an IP connection based on more than one MAC tuples and “NO” otherwise. The return values “YES” or “NO” are for demonstration purposes only. They will be processed further by the EEA and could lead to the output “Assignment successfully finished” or to a message that something went wrong.

Example

Detecting routing functionality is only one example. Based on the traced network packets in the database the EEA will be able to detect several properties and behaviours within a Netkit setup like:

- Presence and configuration of hosts and networks
- Occurrences and properties of application-based data communication, e.g. sent E-Mail, FTP transfer
- Network behaviour, e.g. routing, NAT

Similar to the human course advisor, the EEA should be able to simulate the behaviour of a real course advisor on a technical level. Prepared with more knowledge, optionally aligned in a particular order, the EEA will not be limited to check the final setup, but will be able to guide students while they are working on an assignment, to provide them with hints and help when something goes wrong, and to verify and perhaps grade the students' work.

Finally, we show an example guidance of the Electronic Course Advisor prototype for the previously introduced network example assignment. The listings show the output of the EEA while a student solves the assignment. The output is ordered top-down; new messages from the EEA arrive at the bottom and can supersede previous messages.

The student will first start the EEA, which can even be an automated task, and receives the output of listing 6.3. The VCL is initially clean which means that no host or network is started yet.

Listing 6.3: Example screen 1

```
The Electronic Course Advisor is now online.  
[TODO] I can see 0 network(s). This exercise requires at least 2.
```

The first event that could be detected by the EEA is a newly created network. This implies, that at least one host is started as well. Nevertheless, the host will start with an unconfigured network interface and therefore

cannot be seen by the EEA yet. This is why the check for existing networks is done first. The student will now start the client, connected to a network, by typing the command in line 2 of listing 6.1 on page 80.

Listing 6.4: Example screen 2

```
The Electronic Course Advisor is now online.  
[TODO] I can see 0 network(s). This exercise requires at least 2.  
[TODO] I can see 1 network(s). This exercise requires at least 2.  
[TODO] I can see 0 hosts(s). This exercise requires at least 3.  
■
```

The started network will be noticed by the EEA as shown in listing 6.4 and triggers the check for hosts. After this, the student will start the router and the server according to line 3 and 4 of listing 6.1.

Listing 6.5: Example screen 3

```
The Electronic Course Advisor is now online.  
[TODO] I can see 0 network(s). This exercise requires at least 2.  
[TODO] I can see 1 network(s). This exercise requires at least 2.  
[TODO] I can see 0 hosts(s). This exercise requires at least 3.  
[OK] I can see 2 network(s). This is fine for this exercise.  
■
```

The appearance of two networks can be recognized by the EEA as shown in listing 6.5. While the network interfaces of the hosts are still unconfigured, the presence of the hosts cannot be detected by the EEA yet. The student will now configure and start the network interface of the client according to line 6 of listing 6.1 on page 80. This could be recognized by the EEA as shown in listing 6.6 on the facing page due to some network discovery packets initially sent by the TCP/IP protocol stack.

Proceeding the student will configure and start the network interfaces of the router and the server as well. The appropriate commands can be found in line 8, 9 and 11 of listing 6.1 on page 80. The EEA already noticed in listing 6.6 on the facing page, that the minimal count of different networks (2) was fulfilled. Now the minimal count of hosts (3) will be fulfilled as well (see listing 6.7 on the next page). Adding the default

Listing 6.6: Example screen 4

```
The Electronic Course Advisor is now online.
[TODO] I can see 0 network(s). This exercise requires at least 2.
[TODO] I can see 1 network(s). This exercise requires at least 2.
[TODO] I can see 0 hosts(s). This exercise requires at least 3.
[OK] I can see 2 network(s). This is fine for this exercise.
[TODO] I can see 1 hosts(s). This exercise requires at least 3.
```

gateway as listed in line 13 and 15 of listing 6.1 on page 80 did not cause network packets and cannot be detected by the EA.

Listing 6.7: Example screen 5

```
The Electronic Course Advisor is now online.
[TODO] I can see 0 network(s). This exercise requires at least 2.
[TODO] I can see 1 network(s). This exercise requires at least 2.
[TODO] I can see 0 hosts(s). This exercise requires at least 3.
[OK] I can see 2 network(s). This is fine for this exercise.
[TODO] I can see 1 hosts(s). This exercise requires at least 3.
[TODO] I can see 2 hosts(s). This exercise requires at least 3.
[OK] I can see 3 host(s). This is fine for the current exercise.
[TODO] Show me that your routing setup is correct.
```

To finish the exercise the EEA asks the student at the end of listing 6.7 to show that the current setup is suitable to demonstrate routing. The student can for example use the command `ping 11.0.0.1` on the client, trying to reach the server. The EEA uses the previously introduced SQL query to check for routing behaviour.

If the `ping` command succeeds, the EEA will notice the routing behaviour and can finally congratulate the student in listing 6.8 on the next page because of a successfully solved exercise.

The output generated by the EEA is for demonstration purpose only and should be seen as an example for guidance. For this example, the EEA was prepared with knowledge to detect when students did something right and to give feedback on what is missing with respect to the assignment. Another requirement is that the EEA should be able to deal with situations

Listing 6.8: Example screen 6

```
The Electronic Course Advisor is now online.
[TODO] I can see 0 network(s). This exercise requires at least 2.
[TODO] I can see 1 network(s). This exercise requires at least 2.
[TODO] I can see 0 hosts(s). This exercise requires at least 3.
[OK] I can see 2 network(s). This is fine for this exercise.
[TODO] I can see 1 hosts(s). This exercise requires at least 3.
[TODO] I can see 2 hosts(s). This exercise requires at least 3.
[OK] I can see 3 host(s). This is fine for the current exercise.
[TODO] Show me that your routing setup is correct.
[OK] Routing detected.
Congratulation, your routing setup seems to work. Well done!
```

where students make mistakes. The EEA then e.g. should give hints on what is wrong and how this could be corrected. Determining when the student will get what kind of information needs further research.

6.4 Educational Feasibility

For a proof of concept implementation our SQL based approach is very applicable. However, an open task is to find a proper and efficient way to model the advisor's knowledge in a more human centred way than SQL statements.

In this section, we introduce an exercise assistant for networking courses which is able to support students while they work on networking exercises. Equipped with a formal model of an exercise, the exercise assistant can be run on a student's computer whenever and wherever support is needed. The effort to author such an exercise has to be done once while instances of the exercise assistant equipped with this exercise will then be able to support any number of students.

Exercise Modelling

In the following we show how the exercises can be transferred into a formal representation, in order to be processed by a computer program. First, we will show the partition of our example exercise into activities that will

then be organized in a graph structure. This graph will then be extended with conditions that will make the activities verifiable. We also show a way to add feedback attributes to the graph in order to model a certain feedback strategy. Finally, we introduce probing, a mechanism to improve the verifiability of activities.

Activities

Typically, exercises will start with an empty lab. Students have to perform activities that result in a working network environment, configured according to the requirements of the given exercise. While listing 6.1 on page 80 shows the commands needed to solve the exercise in Netkit, the minimal conceptual activities needed for solving this exercise are listed in table 6.1 on page 104.

While A10 is the final activity, the order of the activities A1 through A9 shows only one possible sequence. The order can vary because some activities are independent from each other (e.g. A1 and A2), while some other activities have interdependencies (e.g. A1 is a precondition for A3).

These activities and their interdependencies can be modelled as an acyclic, directed graph with exactly one sink (node N with $outdegree(N)=0$) and at least one source (node N with $indegree(N)=0$). Activities are represented by nodes. A precondition is modelled as a directed edge from the predecessor to the successor, seamlessly indicating the order of the activities. The final activity will be represented by a sink. Activities without a precondition will be represented by sources.

A valid graph for our example exercise is shown in figure 6.3 on the following page. This graph is based on the activities stated in table 6.1 on page 104. The interdependencies and thus possible sequences of activities show a valid example. These can of course vary, depending on the exercise and the author's intent, too.

Conditions

In order to process the graph, the activities have to be verifiable. That means that a condition is needed to detect or to decide, whether an activity is deemed passed, i.e. whether the student has successfully solved a part of the exercise.

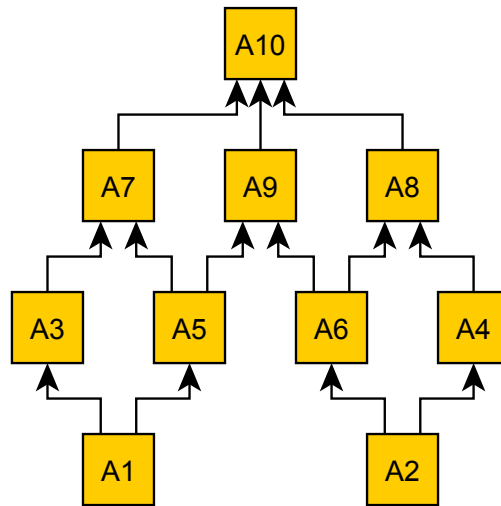


Figure 6.3: Example graph

In chapter 6.3 and [Haag et al., 2012] we showed, that network packets, obtained from the student’s Netkit lab, can be used to detect and verify network properties and behaviour of an Ethernet based network. By modelling network specific expert knowledge as predicates and verifying these predicates using the captured network packets, it is possible to detect e.g. the presence of certain hosts and also routing behaviour. While the prototype demonstrated the technical feasibility of that approach by using SQL queries to model predicates, we improved on it by using description logics [Baader et al., 2003].

For the terminological box (TBox) we created a network ontology for Ethernet based networks, representing the network layers 2 and above [Tanenbaum, 1985], including but not limited to the header and payload fields of the most commonly used protocols, e.g. Ethernet (RFC1042), ARP (RFC826), IP (RFC791), TCP (RFC793) and UDP (RFC768). In addition, we added a unique identifier for each packet and the network origin. An excerpt of our ontology for Ethernet networks is shown in Figure 6.4 on the facing page.

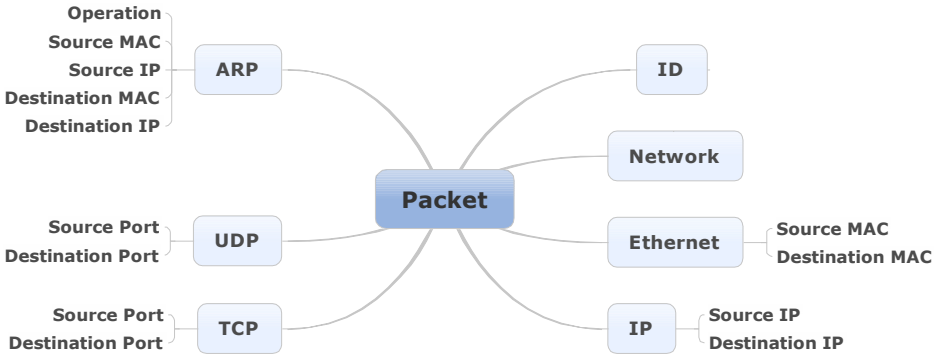


Figure 6.4: Ontology excerpt for Ethernet networks

Using this ontology, it is possible to model expert knowledge as predicates using a logic programming language, e.g. Prolog [Colmerauer and Roussel, 1993]. For example, the expert knowledge to describe the network behaviour “routing” is:

“Routing occurs if an OSI layer 3 IP transmission of a network packet between two hosts is based on more than one OSI layer 2 transmissions.”

The technical background is shown in figure 6.5 on the next page. The client wants to communicate with the server using the IP protocol, but the server is located in a different network segment. Direct intercommunication between client and server is not possible because the underlying Ethernet protocol does not support communication over network borders. The client has to use a known router located in the same network as itself, and thus reachable by Ethernet. The client now sends an IP packet addressed to the IP address of the server, but the underlying Ethernet packet will be addressed to the router. When the router does receive such a packet, it will forward it to the server. While the two packets that the client and the router send do not differ on the IP layer (both are sent from the client, and addressed to the server), both differ on the Ethernet layer, with different source and destination MAC addresses.

Based on the Ethernet network ontology, this behaviour can be expressed as the Prolog predicate in listing 6.9 on page 93. This predicate

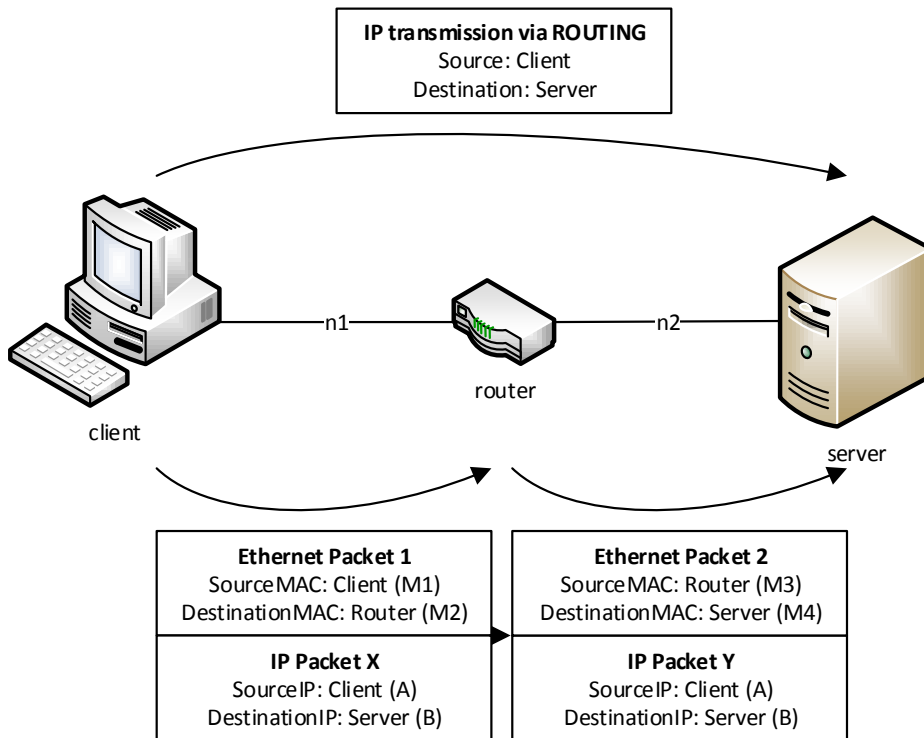


Figure 6.5: Routing packet flow example

Listing 6.9: Prolog predicate for routing

```
1 routing :-  
2   ip_packet(X,A,B),  
3   ip_packet(Y,A,B),  
4   ethernet_packet(X,M1,M2),  
5   ethernet_packet(Y,M3,M4),  
6   M1 \= M3, M2 \= M4.
```

can be read as “*routing occurs, when there are two IP layer packets X and Y, both sent from IP address A to IP address B, for which the source and destination addresses differ on the Ethernet layer.*”

Predicates can be used as conditions to detect activities. E.g. the predicate “routing” can be used to verify the activity A10. We extended the graph, so that every activity can be associated with a condition to verify that activity.

Routing is only one example. We successfully created predicates describing e.g. the presence of hosts and networks, the network behaviour NAT or routing and also higher-level usage. E.g. ARP spoofing behaviour can be detected if two hosts within the same subnet having different MAC addresses pretend to own the same IP address using the ARP protocol. However, this behaviour can also be caused by a misconfiguration of the hosts. For that reason, this condition requires preconditions to verify a valid and error-free setup.

We also found a trade-off between the shape of an assignment and the capabilities to design predicates. If the assignment is more tightly controlled (e.g. predefined network names and IP addresses), more precise predicates can be designed to detect activities. If the assignment is more generic, the predicates also have to be designed in a more generalized manner.

Feedback

There are various types of feedback strategies which can be used to support students working on the exercise, e.g. suggestions, complete guiding or an exam mode. The specific shape will be either customized to match

the author's aims or customized to the learning style of the learner or a combination. Usually recent progress the student has made in the exercise graph should trigger interaction with the student according to the feedback strategy.

Therefore, we extended the graph with feedback attributes. The graph as a whole can be associated with an attribute containing the exercise description; all activities can be associated with different attributes for feedback control, i.e. text messages that give hints about what the next activity might involve (pre-messages), or text messages that give feedback about detected activities (post-messages). An example for activity A1 from our example exercise look like listing 6.10.

While our message mechanism provides the technical means for the implementation of various feedback strategies, the evaluation and choice of an appropriate strategy resides with the exercise author.

Probing

While the verification of activities based on passively observed network packets works for many activities, there still are limitations. One such limitation occurs when an activity needs to be verified that does not have immediate results in the form of network packets.

An example for that would be A9 from our example exercise: the routing functionality has to be activated on the router. Students can do that by setting the appropriate kernel flag²⁷ on the router if this flag is not enabled by default. This however will not result in the occurrence of observable network packets, until packets are sent to the router for being routed. A possible solution would be to ask the student to send appropriate network packets himself. We followed a different approach. For

²⁷Enable IP Forwarding: `echo 1 > /proc/sys/net/ipv4/ip_forward`

Listing 6.10: Example feedback attributes

```
1 pre_message = "You will need at least one host connected to
   network 'n1'."
2 post_message = "Network 'n1' detected."
```

detecting certain activities, we inject special predefined network packets into the Netkit environment to provoke a certain predictable behaviour. This behaviour can also be expressed as a predicate. In the routing example, we inject an Ethernet packet addressed to the router into the client network that is addressed to a host in the server network (which does not have to exist) on the IP level. If routing is enabled in the router, the router will try to reach that host in the server network using ARP requests. These packets can be used to verify that routing is indeed enabled on the router.

Such a “probing” packet can be assembled by strictly following the network stack, starting with an Ethernet frame. The destination MAC address must be the router’s NIC connected to network n1. In Netkit, the MAC address of a network interface is bound to the name of the client, resulting in a predictable MAC address for the router’s first NIC eth0: 0aab64910980. The source MAC address can be virtual, e.g. eeba7b99bca5, followed by an IPv4 ethertype identifier (0x0800). The encapsulated IP packet starts with the version identifier (0x4), followed by mandatory header fields, e.g. length and checksum. The source IP address can be virtual but should be located within the IP range of network n1. The destination IP address can also be virtual but must be part of subnet n2. The IP packet encapsulates an ICMP echo request just to get a complete and valid network packet. This customized packet layout can be represented by a hexadecimal character array, e.g. 0aab6491 0980eeba 7b99bca5 08004500 001c1234 4000ff01 549c0a00 00010b00 00100800 f7fd0001 0001. Tools, e.g. PackEth²⁸, can help authors to design and validate such packets.

We extended the graph, so that every activity can be associated with a custom network “probing” packet to be sent once before verifying its condition. While that actively alters the environment, it enables the verification of additional activities.

6.5 The Electronic Exercise Assistant

In order to support a student while working on an exercise, we developed an exercise assistant, which can be used in the VCSL. As shown in figure 6.6,

²⁸PackEth: Packet generator tool for ethernet. <http://packeth.sourceforge.net>, Online, accessed April 2015.

the exercise assistant is composed of three components: reasoning engine, feedback engine, and an interface to the student's working environment called Netkit interface.

The reasoning engine itself is composed of a reasoner and a knowledge base, which contains a TBox ("terminology box") and an ABox ("assertion box"). The TBox contains knowledge about the domain, i.e. our ontology, in the form of predefined predicates that can be extended by the author with exercise specific extensions, while the ABox contains the concrete instantiations.

The data in the ABox is obtained through an interface to the "real world", in our case the Netkit interface. The Netkit interface consists of one or more Ghost Hosts [Vranken et al., 2011] that record network packets from their respective Netkit network, extract the information in them and store that information in the ABox. The Ghost Hosts can also be used to inject special network packets into the environment.

The feedback engine is the part where the activity graph will be processed. Our exercise assistant is able to read an exercise graph stored in the GraphML²⁹ [Brandes et al., 2002] format. Once read, the activities are continuously processed according to their interdependencies, starting at the source nodes which represent activities without preconditions. Processing the activities in this case means verifying their conditions and giving the student feedback according to the feedback attributes of that activity. Once the activity is completed it will be removed from the graph and thus as a precondition for its successors. The feedback engine can also use the Netkit interface, respectively the Ghost Hosts, to insert custom network packets into the environment in order to provoke certain network behaviour to verify an activity's condition using the reasoning engine.

The Exercise Assistant is a software program written in the programming language *C* using SWI-Prolog³⁰ [Wielemaker, 2009] as the reasoning engine.

²⁹GraphML: "GraphML is a comprehensive and easy-to-use file format for graphs.", <http://graphml.graphdrawing.org>, Online, accessed April 2015

³⁰SWI-Prolog: "SWI-Prolog offers a comprehensive free Prolog environment.", <http://www.swi-prolog.org>, Online, accessed April 2015.

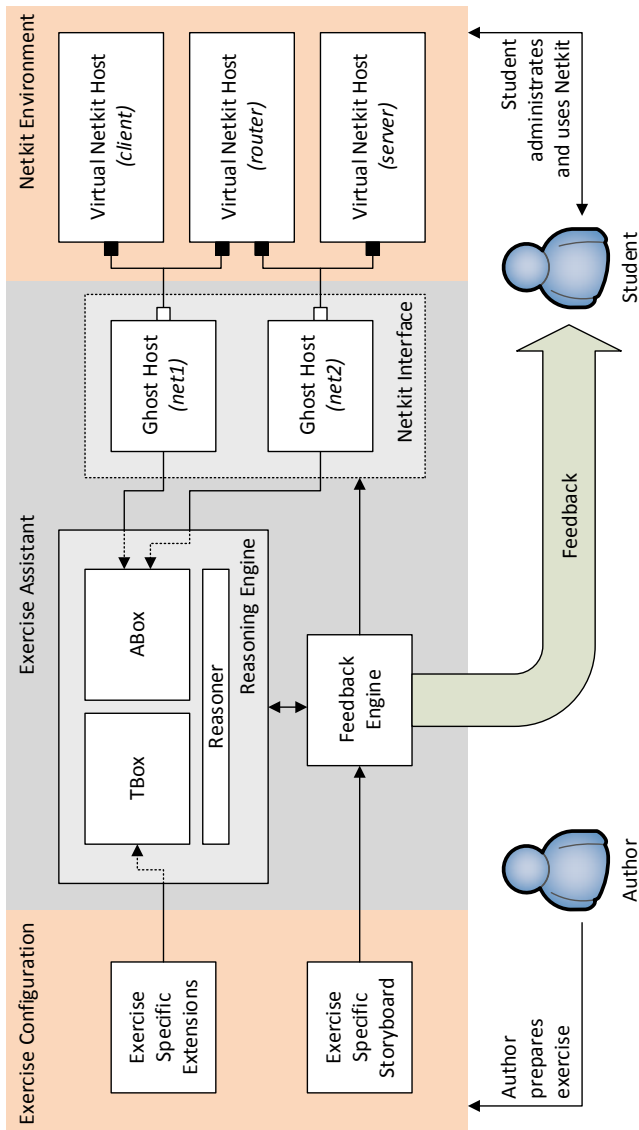


Figure 6.6: Exercise assistant architecture draft

description. Starting with the activities without precondition (A1 and A2), the exercise assistant will prompt the student using the respective `pre_messages` (see output in figure 6.8).

```

Exercise Assistant Shell
Example Exercise 1: IP Routing

Setup and configure at least three hosts (client, router, server).
Client and server should be located in different networks. The
client should be able to intercommunicate with the server by using
the intermediate router. Verify your routing environment by using
sending routed network packets between client and server.

Please name the network(s) and the host(s) according to the
diagram below.

Diagram:
+-----+           +-----+           +-----+
|client| <-----n1-----> |router| <-----n2-----> |server|
+-----+           +-----+           +-----+

IP addresses:

    client                router                server
    10.0.0.1 <-----n1-----> 10.0.0.2
                                |
                                11.0.0.2 <-----n2-----> 11.0.0.1

Good luck!

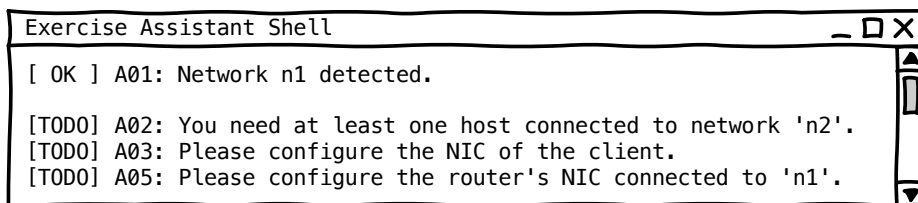
[TODO] A01: You will need at least one host connected to 'n1'.
[TODO] A02: You will need at least one host connected to 'n2'.

```

Figure 6.8: EA Guiding Example 1

The student can start solving the exercise according to figure 6.1. After the first command `vstart client --eth0=n1` is entered using the linux shell, the exercise assistant is able to confirm this valid activity (see output in figure 6.9 on the following page).

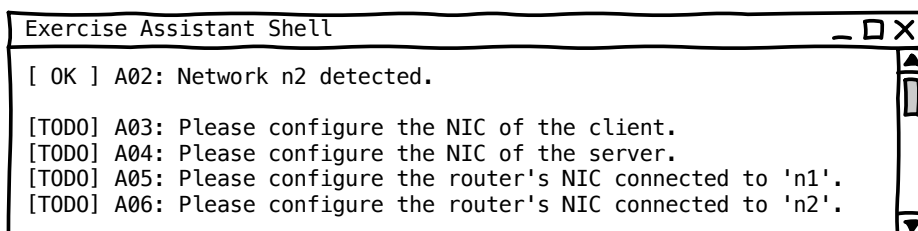
While A1 is being marked as verified, using the respective `post_message` of A1, the remaining independent activities without preconditions will be displayed again, superseding the preceding messages. According to the exercise graph, the student is now able to choose A2, A3 or A5 as the next



```
Exercise Assistant Shell
[ OK ] A01: Network n1 detected.
[TODO] A02: You need at least one host connected to network 'n2'.
[TODO] A03: Please configure the NIC of the client.
[TODO] A05: Please configure the router's NIC connected to 'n1'.
```

Figure 6.9: EA Guiding Example 2

activity. Starting the router connected to network n1 and n2 results in a verified presence of n2 (see output in figure 6.10).



```
Exercise Assistant Shell
[ OK ] A02: Network n2 detected.
[TODO] A03: Please configure the NIC of the client.
[TODO] A04: Please configure the NIC of the server.
[TODO] A05: Please configure the router's NIC connected to 'n1'.
[TODO] A06: Please configure the router's NIC connected to 'n2'.
```

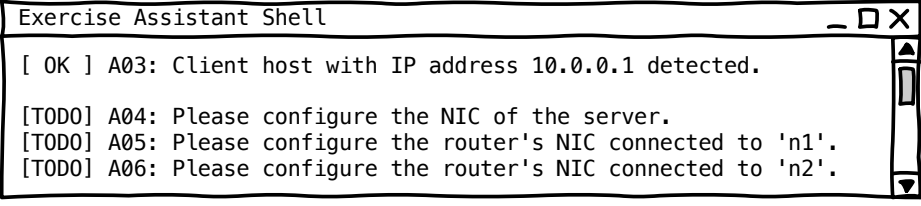
Figure 6.10: EA Guiding Example 3

While the presence of the two networks is verified now, the exercise assistant is not able to detect whether the student has started the server, unless its network interface card gets assigned an IP address. Therefore the `pre_messages` are authored to prompt the student properly. Choosing to assign the client's IP address as next activity, using the command `ifconfig eth0 10.0.0.1 up` in the client shell, will result in a verified activity A3 (see output in figure 6.11 on the facing page).

Still missing IP addresses of router's and server's NICs, the student can proceed to configure the router's NICs (see output in figure 6.12 on page 101 and succeeding output in figure 6.13 on page 101).

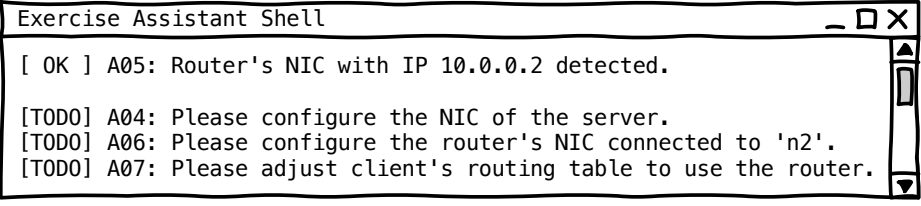
Having verified that the two NICs of the router are present, the exercise assistant is able to verify A9 using a probe packet. For the simple reason that routing is enabled per default for hosts in the Netkit environment, the condition of A9 can be verified immediately (see output in figure 6.14).

After assigning an IP address to the remaining NIC of the server, the



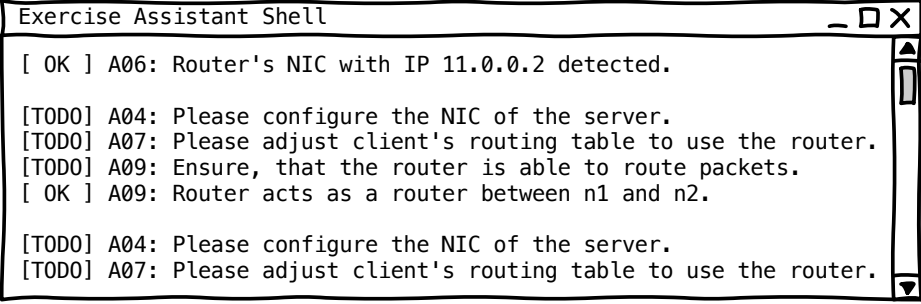
```
Exercise Assistant Shell
[ OK ] A03: Client host with IP address 10.0.0.1 detected.
[TODO] A04: Please configure the NIC of the server.
[TODO] A05: Please configure the router's NIC connected to 'n1'.
[TODO] A06: Please configure the router's NIC connected to 'n2'.
```

Figure 6.11: EA Guiding Example 4



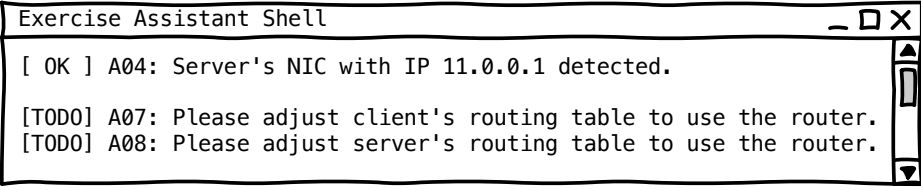
```
Exercise Assistant Shell
[ OK ] A05: Router's NIC with IP 10.0.0.2 detected.
[TODO] A04: Please configure the NIC of the server.
[TODO] A06: Please configure the router's NIC connected to 'n2'.
[TODO] A07: Please adjust client's routing table to use the router.
```

Figure 6.12: EA Guiding Example 5



```
Exercise Assistant Shell
[ OK ] A06: Router's NIC with IP 11.0.0.2 detected.
[TODO] A04: Please configure the NIC of the server.
[TODO] A07: Please adjust client's routing table to use the router.
[TODO] A09: Ensure, that the router is able to route packets.
[ OK ] A09: Router acts as a router between n1 and n2.
[TODO] A04: Please configure the NIC of the server.
[TODO] A07: Please adjust client's routing table to use the router.
```

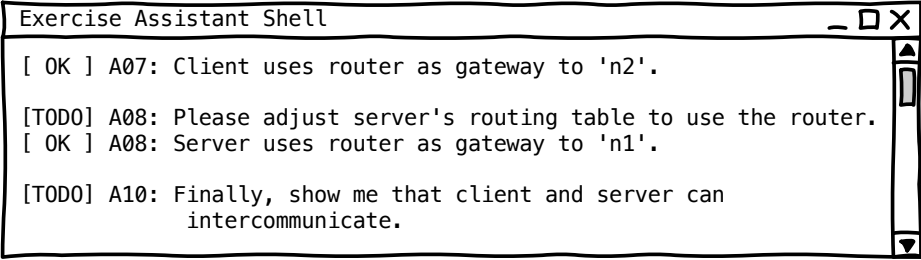
Figure 6.13: EA Guiding Example 6



```
Exercise Assistant Shell
[ OK ] A04: Server's NIC with IP 11.0.0.1 detected.
[TODO] A07: Please adjust client's routing table to use the router.
[TODO] A08: Please adjust server's routing table to use the router.
```

Figure 6.14: EA Guiding Example 7

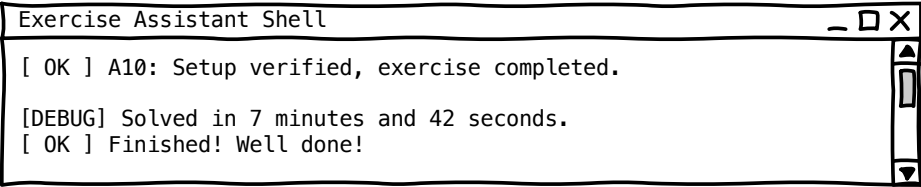
student has to alter the routing table on the client and on the server. The exercise assistant is also able to verify these activities by using probing packets (see output in figure 6.15).



```
Exercise Assistant Shell
[ OK ] A07: Client uses router as gateway to 'n2'.
[TODO] A08: Please adjust server's routing table to use the router.
[ OK ] A08: Server uses router as gateway to 'n1'.
[TODO] A10: Finally, show me that client and server can
intercommunicate.
```

Figure 6.15: EA Guiding Example 8

Finally, the student is asked to demonstrate the routing functionality by sending packets between the client and the server using the intermediate router. One valid solution is to use the command `ping`.



```
Exercise Assistant Shell
[ OK ] A10: Setup verified, exercise completed.
[DEBUG] Solved in 7 minutes and 42 seconds.
[ OK ] Finished! Well done!
```

Figure 6.16: EA Guiding Example 9

Once the final activity is verified, the exercise assistant congratulates the student and then quits (see output in figure 6.16).

6.7 Conclusion

We presented a concept for an Electronic Exercise Assistant in a virtual computer security lab for courses in networking and computer security. By transforming a real-world assignment from a physical lab to an EEA setting in a virtual lab we exemplarily describe, how the correctness of a student solution can be proven by the EEA as well how an EEA can offer guidance

and support to the students. For the proof of the technical feasibility we use a canonical approach by modelling the real-world advisor's knowledge as SQL queries to a database containing all network packets generated during a virtual lab session. As an intermediate step, we give rules on the entirety of all network packets processed.

For the proof of the educational feasibility we also presented an approach to formally model exercises in a manner processable by the exercise assistant. For that purpose, the exercise author can define possible activities and sequences using a graph structure. Description logic is used to define conditions for the verification of these activities. The exercise author is also able to define a feedback strategy by adding feedback attributes to the graph. This includes pre- and post-messages as immediate feedback and does not cover delayed feedback [Mory, 2003, Erev et al., 2006, Hattie and Timperley, 2007] yet.

While the current version of the EEA can give feedback based on observed network packets, a future version may incorporate more values, e.g. typed commands of a student to administrate NetKit, to issue feedback in a more accurate way.

Especially for courses with many participants, our first experience shows that teaching staff can benefit from utilizing the exercise assistant. While the teaching method of tutors personally and individually supporting students is certainly one of the most effective for knowledge transfer, it is not feasible for courses of sufficient size [Odekirk-Hash and Zachary, 2001]. In such scenarios, the exercise assistant can e.g. be used to offer all students a basic guided tutoring support not only wherever and whenever they want, but also at the speed that best suits their own learning style and their own abilities.

Table 6.1: Activities needed to solve the example exercise

Activity	ID
The client network has to be created.	A1
The server network has to be created.	A2
The client has to be connected to the client network and an appropriate IP address has to be assigned.	A3
The server has to be connected to the server network and an appropriate IP address has to be assigned.	A4
One NIC of the router has to be connected to the client network and an IP address from the client network has to be assigned.	A5
One NIC of the router has to be connected to the server network and an IP address from the server network has to be assigned.	A6
The client has to be configured to use the router's NIC in the client network as default gateway.	A7
The server has to be configured to use the router's NIC in the server network as default gateway.	A8
Routing has to be enabled on the router.	A9
Client and server must intercommunicate via the intermediate router using the IP protocol.	A10

Chapter 7

Educational Enhancements

The Virtual Computer Lab (VCL) environment enables students to work on practical networking exercises whenever and wherever they want. While they have to work on their own using the VCL, the Distributed Virtual Computer Lab (DVCL) extension enables students to work and learn together, even if they are distant. Many students rely on group work, thus the DVCL is a reasonable learning environment.

Our experiences with practical on-campus networking courses show that the laboratory, where the courses take place, is more than a room with computer and networking facilities. Rather it is a social place, where students e.g. meet, form learning groups, talk and discuss. Also, the course advisers are involved, e.g. they offer support, manage that learning groups have a certain size and can finally verify an exercise. A shortcoming however is, that the current DVCL environment does not support such interactions. If we model a distributed virtual lab on an on-campus lab, requirements also arise from a social viewpoint.

A student project in 2012 was started to deal with this issue. The aim of this project was to add support for social interactions to the DVCL environment. First, an existing practical on-campus networking course was observed in order to get key factors about student's social behaviour in the lab. Using these key factors, a concept was developed to support certain identified major social interactions in our DVCL environment. Finally, a

This chapter is based on the following publication: [Döhring, 2012, Döhring, 2013, Haag et al., 2017]

prototype was developed to demonstrate the new DVCL virtual classroom.

7.1 Classroom Settings

On-Campus Classroom

We observed a basic practical on-campus networking course with nearly 200 participants and we gained the following major insights:

- The exercises will be delivered in hard copy or electronical (via PDF³¹ document) by the course advisers to the students.
- Some students are working out the exercises on their own and not in a group. Sometimes they talk with other students in the lab.
- Some students are forming a learning group to jointly solve an exercise. They exchange information within the group and between two students. Sometimes one part of the group disturbs the rest of the group. Sometimes they interact with other students or groups.
- Most of the time the students arrange their groups on their own resulting in groups with different size.
- In many cases the intervening of a course advisor is needed if the learning groups should have a fixed size of participants.
- Sometimes a learning group is stalled because the participants have to wait for additional students if the learning group requires to have a fixed size. Another case is that the learning group will start working on an exercise having an insufficient group size.
- Students (with higher skills) will be contacted and asked by other students to get support. Sometimes they will be asked to join a group.
- In rare cases students will leave or join a group subsequently.
- Students, resp. learning groups are able to get support by a course advisor.
- According to the course design, a final solution of an exercise can be verified, revised, rated and graded by a course advisor.

³¹PDF: Portable Document Format, <http://www.adobe.com>, Online, accessed December 2017

Virtual Classroom Concept

Based on our observation, we identified three major groups of activities, which should be supported in the DVCL in addition to the pre-existing environment (cf. part I on page 21 pp.): Communication Activities, Organizational Activities and Educational Activities. Each group and also the interaction between these groups will be introduced in the remainder of this section.

Communicational Activities

An important issue is the communication, especially when students utilize the DVCL from different locations. Based on our observations, we decided to support the following three communication models.

1. **Lobby Chat** The lobby chat is modelled on the classroom (the on-campus lab). Each student, which logs on to the DVCL, enters the lobby chat automatically. He can read messages from all other users. He can write a message which will be visible by all other users instantly.
2. **Group Chat** The group chat is a separate, independent area, which will be created if a group is formed. A sent message will only arrive at other group members. The group chat is meant for communication within a group.
3. **Private Chat** The private chat is meant for communication between two students. A sent message will be readable only by the conversation partner.

Organizational Activities

If no learning style is given, our experiences shows that the students will choose different styles, which are suitable to work best for them. The majority will choose group work in combination with single person working (see figure 5.3 on page 70). According to that, the DVCL should support single person working, that students are able to utilize the lab without being reliant on others. The DVCL should also support group work. For group work, we decided to support the following three models:

1. **Automatic Group Forming** Students can subscribe to an exercise. The DVCL will automatically arrange and assign subscribed students to

groups of sufficient size. An advantage of this model is that this process is fast. A group can be created once enough students subscribed to an exercise. Furthermore, the students are not involved in the group forming process and no social interactions are needed. The groups are arranged without personal preferences. A disadvantage is that personal preferences or pre-existing learning groups will be disregarded; even single person working will be impossible.

2. **Manual Group Forming** Students will create the groups on their own. An advantage is that personal preferences can be considered. Also, single person working will be possible, since no group has to be formed. A disadvantage is that the group forming process will be slower compared to an automatic process. This is because students must get active to create a group, and also agreements are required. Another issue is that more than one group with insufficient participants willing to work on the same exercise can occur.
3. **Combination** The advantages of the automatic and the manual group forming model will be combined. Students can work on their own, can create a group on their own or can subscribe to an exercise accepting the automatic group forming process.

In certain cases, there will be not enough DVCL participants to form a group with sufficient size. Then, a learning group will be unable to start the exercise. To resolve this issue, the group should be able to start the exercise by overwriting the predefined value for the group size. However, this will require the approval of all group members.

Educational Activities

The DVCL as a technical platform is able to provide an environment where students can work on networking exercises. We identified three issues which have to be taken into consideration.

1. **Exercises** The exercises should be available and selectable within the DVCL environment. This will turn the distributed lab as a technical platform even closer to a learning platform. An exercise author would be able to publish exercises directly and he would be able to add attributes, e.g. a predefined group size, which can be processed and managed by the DVCL environment. Levels of difficulty, which will be offered one after another to a student, are also imaginable.

2. **Support** Providing support to students, which are not working in an on-campus classroom but in the DVCL, is a complex topic, which requires more research. One approach could be, that a human course advisor will join the DVCL at different times to support students while solving an exercise. The course advisor will be able to answer questions but he will be unable to verify a certain configuration of a student's solution because there is currently no remote access to a student's lab. Using remote desktop access could be a solution for this. An interesting approach can be adapting the electronic exercise assistant (cf. chapter 6 on page 77) to support a distributed lab environment.
3. **Resources** Resources need to be rearranged when the group size will change and the exercise has already begun. E.g. when a student will leave a group, his part must be accomplished by another group member. Since leaving the group means also losing brought in virtual networks and hosts within the DVCL, this case is within this project out of scope and requires further investigation.

Functional Interaction

The three major groups *communicational*, *organizational* and *educational* activities are illustrated in figure 7.1 on the following page. Each group is represented as a coloured circle within the outer circle, which represents the classroom.

A student will start by entering the classroom. He is now able to pick an activity, e.g. he asks other students in the classroom if everybody is fine. This means using the classroom chat as part of the communicational activities. As an example of an educational activity, he can decide to select an exercise. An example for an organizational activity is to form a learning group by inviting other students.

The selection is not limited to one activity. Furthermore, the activities can be connected but it is not mandatory. E.g. students can find other students in an informal talk, using the classroom chat, and jointly decide to form a learning group. This is a combination of a communicational and an organizational activity, represented as the intersection between these activities in figure 7.1. In another example, a student will choose an exercise (educational activity) and starts talking with another student

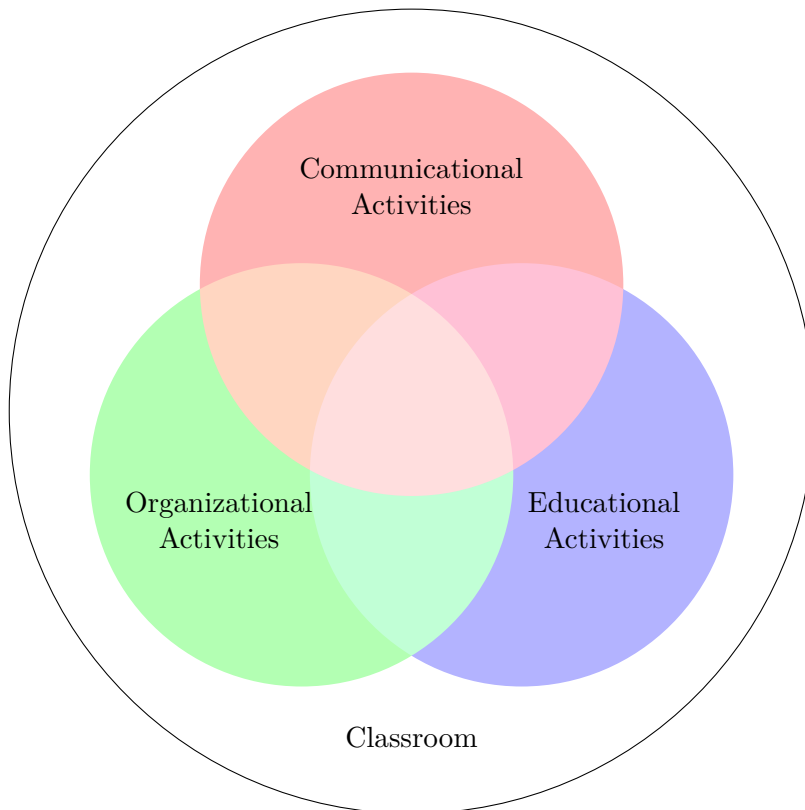


Figure 7.1: Activities in classroom

(communication activity) at the same time.

An example for the intersection of all activity groups is: Some Students will meet in the classroom chat (communication activity). They decide to form a learning group (organizational activity) and jointly solve a certain exercise (educational activity). Another example is: A student selects a certain exercise (educational activity), the automatic group forming process will assign him to an existing group with missing members (organizational activity) and the student starts talking by using the group chat (communication activity).

7.2 Virtual Classroom Prototype

The virtual classroom prototype is based, utilizes and extends our existing DVCL (cf. chapter 2 on page 21) with Central Authority (cf. chapter 3 on page 35), including the security enhancements (cf. chapter 4.1 on page 46) and the GUI (cf. chapter 4.2 on page 52). The prototype is written using the programming language *C* and relies on the GTK³² toolkit.

A student has first to login to use the DVCL. The related screenshot can be found in figure 7.2. Once logged in, the main screen with the lobby, resp. the classroom chat will be presented as illustrated in figure 7.3 on the following page. The lobby consists of three areas: An area where messages will appear to the left, an area with a list of users currently logged in to the right and an area with a text input box to create and send own messages at the bottom.

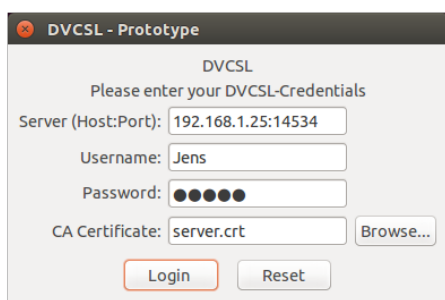


Figure 7.2: GUI login

The student will have more than one activities to proceed. He can start a private chat with another student (communication activity), or he can invite another student to create a group, or he will be invited by another student to join a group (both are organizational activities), or he can proceed to the exercise selection area (educational activity). Starting a private chat will open a new tab similar to the lobby chat, except that messages will only be readable by the selected dialogue partner. A private chat is independently of other states. For a manual group forming, the student can select another student to invite him into a group (Figure 7.4a),

³²GTK: "GTK+, or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces." <http://www.gtk.org>, Online, accessed April 2015

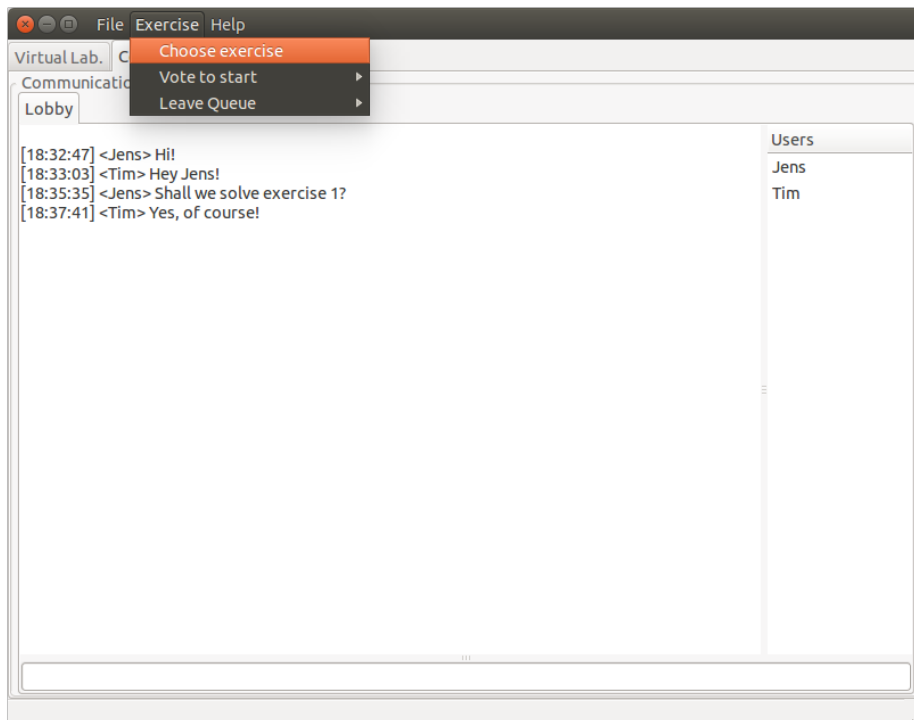


Figure 7.3: DVCL classroom main screen

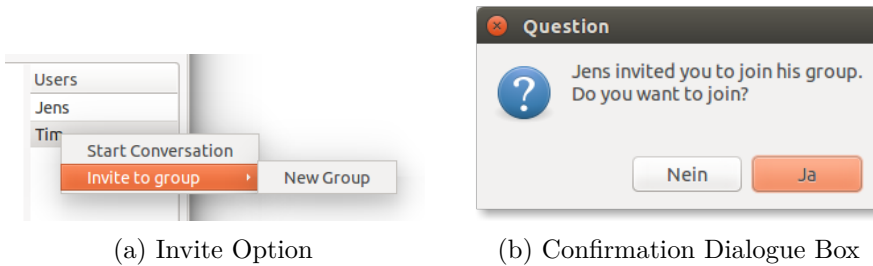


Figure 7.4: Manual Group Forming

or he will be invited to join a group. A pending invitation has to be accepted or rejected in a dialogue prompt (Figure 7.4b). If a group with at least two students was formed, a new tab will appear for the group chat. This tab is similar to the lobby chat, except that messages will only be readable by the group members.

Despite a group already exists or not, the student can access the exercise selection area using the menu on the top (Figure 7.3). By selecting *Exercise > Choose exercise*, a new window will appear where the exercises are listed and can be selected. The exercise consists of a title, a description and attributes, e.g. the desired group size. In figure 7.5, one exercise titled *Basic Networking A1* is included with a group size of 3. After selecting the exercise, the student has to decide to work alone or to search for a group. If the student is already a member of a group, which does not have an exercise selected yet, a third option will be available to assign the selected exercise to this pre-existing group.

Work alone means that the exercise will start immediately without waiting for group members. By selecting *Search a group*, the DVCL environment will look for an already existing group, where at least one group member is still missing and the group decided to solve the same exercise. If such a group is found, the student will be assigned to this group. If no suitable group can be found, the student will be assigned to a waiting queue (see figure 7.6). In case that the queue already holds a suitable student (who has selected the same exercise), a new group will be created for them. Finally, the group size will be checked. If the group is already complete, which means, that the group has the predefined group size, the exercise will be started. If the group is still incomplete, the group

members can vote to start (see figure 7.7) the exercise anyway, but this action will require the approval of all other group members (see figure 7.8).

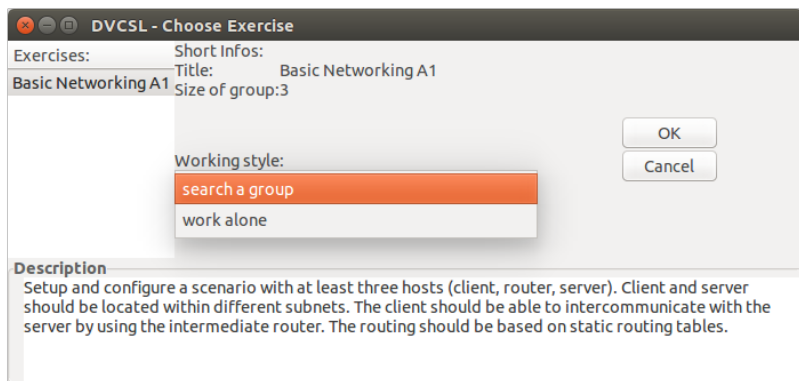


Figure 7.5: Exercise Selection

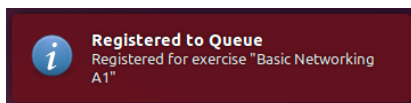


Figure 7.6: Student is assigned to waiting queue

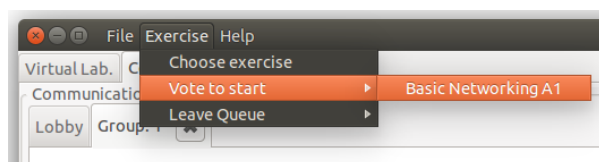


Figure 7.7: Overwrite a predefined group size

7.3 Conclusion

Our concept and the prototype shows a way to turn the distributed lab as a technical platform even closer to a distributed learning platform with benefits for all involved parties. The course advisor will get a single place

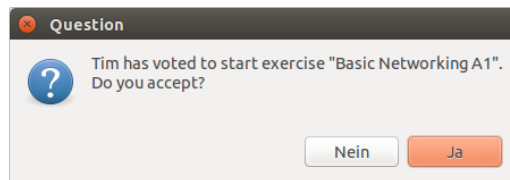


Figure 7.8: Confirmation Dialogue

where exercises can be published. Moreover, rules can be predefined, e.g. the group size, which will be managed, preserved and verified by the DVCL environment, even if a human course advisor is not present. For students, the DVCL is no longer a technical platform, where virtual networks can be remotely connected but a virtual place, where they can meet, communicate, arrange learning groups, exchange experiences and work on exercises.

Summary

Teaching networking and IT security in higher education requires a safe playground for students, where they can safely carry out hands-on exercises. This safe playground is known as a computer lab. Universities have to design and to provide such a lab with respect to certain criteria, e.g. technical opportunities, educational requirements and demands of the learners. Since there is no one-size-fits-all lab, the labs will be designed to fit into a certain context and thus have own strengths and weaknesses.

In this thesis, we investigate and work with two established labs, which were designed for hands-on experiences in networking and it security courses. These labs are predominantly different but have an essential overlap in educational requirements.

One lab is developed by the Open University. It is dedicated for distance learning. It is based on virtualization and every student is able to start this lab on his own computer. Students can work out exercises whenever and wherever they want. A shortcoming however is that students have to work alone, (distant) group work is not possible due to the isolated lab architecture. This lab is the technical base for our research.

The other lab is developed by the Cologne University. It is a physical lab, dedicated for on-campus courses and thus it is not portable. But students can meet in the lab, work in groups and are able to get support from a course advisor, who is also able to verify exercises. A shortcoming however is that students must be present (they have to travel to the university) and they are dependent on the opening hours of the laboratory and the availability of the course advisers.

In two research parts, we show how such two different lab approaches can be combined and what can be achieved.

The first research part is about **design issues**.

Initially, we enable group work in our lab for distance education, since group work is an essential part in on-campus classes. Also remote students should be able to work together. Since the lab is designed as an isolated system, the challenge is to connect two of them on the network level but without creating a potential bridge between the isolated and the outside world. We achieved this by adding a communication interface to the lab architecture. This communication interface consists of a ghost host to extract and inject network packets, and a remote bridge endpoint, to transport these packets between remote ghost hosts across an intermediate connection, e.g. the internet. The developed prototype is called Distributed Virtual Computer Lab (DVCL) and enables to connect two or more distant labs while preserving the isolated character.

The DVCL is then extended and improved by a central authority (CA). While the point-to-point connection of the communication interface can connect two remote networks in a handy way, more connections require careful planning by the students. We show that a CA simplifies the usage of our DVCL for the students (and also for academic staff) and in addition to it avoids administrative configuration errors while connecting remote labs, e.g. a circular flow which leads to an unusable lab.

The first part is completed by two applicability enhancements. The first enhancement covers and resolves security issues in order to push our prototypical implementation of the DVCL and the CA closer to a productive learning environment. The second enhancement introduces a Graphical User Interface to increase the usability of the DVCL.

The second research part is about **educational aspects**.

In the first part, we assume that working independent from a physical on-campus lab as well as group work is essential for our students. Our evaluation of more than 200 students participating in an on-campus networking course shows, that nearly half of the students actually say, that they would like to work independently from the university at least partially and they would welcome the introduction of an e-learning system. In addition, a predominant majority think of working in groups as well as receiving guidance and feedback as crucial to their learning success. This result justifies and confirms our research and also reveals an additional requirement.

The challenge is to provide feedback and guidance to a student, who is working on an exercise and a human course advisor is not available. This is e.g. when students use the DVCL at home in the evening hours. We show, that captured network traffic of a lab can give some indication of what a student has already configured according to a certain exercise. We use this insight to develop an Electronic Exercise Assistant. This software program is able to recognize the progress of an exercise and can provide appropriate feedback and support, based on preloaded rules and conditions. This significantly improves the learning situation for students working remotely in lab. Besides this automatic support, the exercise assistant can verify intermediate and complete solutions of an exercise.

The second part is completed by an educational enhancement. Our evaluation and also own observations show, that a lab is more than a room with computer and network facilities. Rather it is a social place where students e.g. meet, form learning groups, talk and discuss. We use these insights and enhance the DVCL to support social interactions. Based on our on-campus lab as source, we model a set of communicational, organizational as well as educational activities and implement them in our DVCL. The result shows, that our DVCL prototype is no longer a technical platform but a virtual place, where students can meet, communicate, arrange learning groups, exchange experiences and work on exercises.

This thesis shows that aspects of our two different lab environments can be combined. Our resulting Distributed Virtual Computer Lab incorporates strengths of each source lab. It is a gain for distance teaching as well as for on-campus classes. Remote students are now able to utilize the lab being a virtual classroom, where they can learn in groups, assisted by an electronic advisor and without the need for a face-to-face meeting. On-campus classes can offer students a new learning environment, where they can learn in a classroom character without the need to travel to the university.

Bibliography

- [Agarwal et al., 2001] Agarwal, K. K., Critcher, A., Foley, D., Sanati, R., and Sigle, J. (2001). Setting up a classroom lab. *J. Comput. Sci. Coll.*, 16(3):281–286. Cited on page 3.
- [Albacete and VanLehn, 2000] Albacete, P. and VanLehn, K. (2000). The conceptual helper: An intelligent tutoring system for teaching fundamental physics concepts. In *Intelligent tutoring systems*, pages 564–573. Springer. Cited on page 77.
- [Alfers, 2013] Alfers, T. (2013). Formale Beschreibung komplexer Praktikumsaufgaben aus dem Bereich Netze und IT-Sicherheit – Weiterentwicklung eines Systems zur automatisierten Verifikation von Lösungen. Bachelor’s thesis (Studienarbeit), Cologne University of Applied Sciences, Campus Gummersbach, Gummersbach, Germany. Cited on pages 17 and 77.
- [Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA. Cited on page 90.
- [Bardas and Ou, 2013] Bardas, A. G. and Ou, X. (2013). Setting up and using a cyber security lab for education purposes. *J. Comput. Sci. Coll.*, 28(5):191–197. Cited on page 3.
- [Berman et al., 2014] Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I. (2014). GENI: A federated testbed for innovative network experiments. *Computer Networks*, 61(0):5–23. Cited on page 4.

- [Bishop and Heberlein, 1996] Bishop, M. and Heberlein, L. (1996). An isolated network for research. In *Proceedings of the Nineteenth National Information Systems Security Conference*, pages 349—360. Cited on page 3.
- [Bonk et al., 2005] Bonk, C. J., Graham, C. R., Cross, J., and Moore, M. G. (2005). *The Handbook of Blended Learning: Global Perspectives, Local Designs*. Pfeiffer & Company. Cited on page 7.
- [Bonofiglio et al., 2018] Bonofiglio, G., Iovinella, V., Lospoto, G., and Battista, G. D. (2018). Kathará: A container-based framework for implementing network function virtualization and software defined networks. *To appear at IEEE/IFIP Network Operations and Management Symposium, 23-27 April 2018, Taipei, Taiwan*. Cited on page 12.
- [Border, 2007] Border, C. (2007). The development and deployment of a multi-user, remote access virtualization system for networking, security, and system administration classes. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '07*, pages 576–580, New York, NY, USA. ACM. Cited on page 4.
- [Brandes et al., 2002] Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., and Marshall, M. (2002). Graphml progress report structural layer proposal. In Mutzel, P., Jünger, M., and Leipert, S., editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 501–512. Springer Berlin Heidelberg. Cited on page 96.
- [Brian Hay, 2006] Brian Hay, K. L. N. (2006). Evolution of the assert computer security lab. In *Proceedings of the 10th Colloquium for Information Systems Security Education*. Cited on page 3.
- [Bullers et al., 2006] Bullers, Jr., W. I., Burd, S., and Seazzu, A. F. (2006). Virtual machines - an idea whose time has returned: Application to network, security, and database courses. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '06*, pages 102–106, New York, NY, USA. ACM. Cited on page 3.
- [Canfield, 2001] Canfield, W. (2001). Aleks: A web-based intelligent tutoring system. *Mathematics and Computer Education*, 35(2):152. Cited on page 77.

- [Catuogno and De Santis, 2008] Catuogno, L. and De Santis, A. (2008). An internet role-game for the laboratory of network security course. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '08*, pages 240–244, New York, NY, USA. ACM. Cited on page 1.
- [Colmerauer and Roussel, 1993] Colmerauer, A. and Roussel, P. (1993). The birth of prolog. In *The Second ACM SIGPLAN Conference on History of Programming Languages, HOPL-II*, pages 37–52, New York, NY, USA. ACM. Cited on page 91.
- [Comer, 2008] Comer, D. E. (2008). *Computer Networks and Internets*. Prentice Hall, 5rd edition. Cited on pages 22 and 26.
- [Corbett et al., 1997] Corbett, A. T., Koedinger, K. R., and Anderson, J. R. (1997). Intelligent tutoring systems. *Handbook of human-computer interaction*, 5:849–874. Cited on page 77.
- [Crowley, 2004] Crowley, E. (2004). Experiential learning and security lab design. In *Proceedings of the 5th Conference on Information Technology Education, CITC5 '04*, pages 169–176, New York, NY, USA. ACM. Cited on page 1.
- [Damiani, 2006] Damiani, E. (2006). The open source virtual lab: a case study. In *Proceedings of the Workshop on Free and Open Source Learning Environments and Tools, FOSLET '06*. Cited on page 3.
- [Dierks and Allen, 1999] Dierks, T. and Allen, C. (1999). The tls protocol version 1.0. <http://www.ietf.org/rfc/rfc2246.txt>. [Online; accessed 22-July-2014]. Cited on page 51.
- [Dierks and Rescorla, 2006] Dierks, T. and Rescorla, E. (2006). The transport layer security (tls) protocol version 1.1. <http://www.ietf.org/rfc/rfc4346.txt>. [Online; accessed 22-July-2014]. Cited on page 51.
- [Dierks and Rescorla, 2008] Dierks, T. and Rescorla, E. (2008). The transport layer security (tls) protocol version 1.2. <http://www.ietf.org/rfc/rfc5246.txt>. [Online; accessed 22-July-2014]. Cited on page 51.

- [Dike, 2000] Dike, J. (2000). A user-mode port of the linux kernel. In *Proceedings of the 4th Annual Linux Showcase & Conference - Volume 4*, ALS'00, pages 7–7, Berkeley, CA, USA. USENIX Association. Cited on page 8.
- [Dike, 2006] Dike, J. (2006). *User Mode Linux*. Prentice Hall. Cited on pages 8 and 23.
- [Drigas et al., 2005] Drigas, A. S., Vrettaros, J., Koukianakis, L. G., and Glentzes, J. G. (2005). A virtual lab and e-learning system for renewable energy sources. In *Proceedings of the 1st WSEAS/IASME Conference on Educational Technologies*, EDUTE '05, pages 149–153. Cited on page 4.
- [Döhring, 2012] Döhring, C. (2012). Konzeption und Implementierung eines grafischen Userinterfaces für das DVCSL. Project work (Praxis-Projekt) in the scope of the Computer Science Bachelor's Program at the Cologne University of Applied Sciences, Campus Gummersbach. Cited on pages 17, 52, and 105.
- [Döhring, 2013] Döhring, C. (2013). Konzeption und Teilimplementierung einer Plattform für kollaboratives Arbeiten in einem verteilten virtuellen Computersicherheitslabor. Bachelor's thesis (Studienarbeit), Cologne University of Applied Sciences, Campus Gummersbach, Gummersbach, Germany. Cited on pages 18 and 105.
- [Döhring and Kahrau, 2011] Döhring, C. and Kahrau, S. (2011). Einführung einer Zugangsbeschränkung eines verteilten virtuellen Computersicherheitslabors (kurz DVCSL). Project work (AI-Projekt) in the scope of the Computer Science Bachelor's Program at the Cologne University of Applied Sciences, Campus Gummersbach. Cited on pages 17 and 46.
- [Eckert, 2013] Eckert, C. (2013). *IT Sicherheit*, chapter 1.2 Security Objectives. Oldenbourg Verlag. Cited on page 46.
- [Ellabidy and Russo, 2014] Ellabidy, M. and Russo, J. P. (2014). Using the cloud to replace traditional physical networking laboratories. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 729–729, New York, NY, USA. ACM. Cited on page 4.

- [Erev et al., 2006] Erev, A., Luria, A., and Erev, I. (2006). On the effect of immediate feedback. Cited on page 103.
- [Gerdes et al., 2010] Gerdes, A., Heeren, B., and Jeuring, J. (2010). Properties of exercise strategies. In *Proceedings International Workshop on Strategies in Rewriting, Proving, and Programming, IWS 2010, Edinburgh, UK, 9th July 2010.*, pages 21–34. Cited on page 77.
- [Haag et al., 2011] Haag, J., Horsmann, T., Karsch, S., and Vranken, H. (2011). A distributed virtual computer security lab with central authority. In *Proceedings of the Computer Science Education Research Conference, CSERC '11*, pages 89–95. Open Universiteit, Heerlen. Cited on pages 16, 17, and 35.
- [Haag et al., 2012] Haag, J., Karsch, S., Vranken, H., and Van Eekelen, M. (2012). A virtual computer security lab as learning environment for networking and security courses. In *Proceedings of the 3rd Annual International Conference on Computer Science Education: Innovation and Technology, CSEIT '12*, pages 61–68. Global Science & Technology Forum. Cited on pages 16, 17, 77, and 90.
- [Haag et al., 2017] Haag, J., Vranken, H., and Van Eekelen, M. (2017). A virtual class room for cybersecurity education. *Manuscript submitted for publication*. Cited on pages 16, 17, 18, 46, 52, and 105.
- [Haag et al., 2013] Haag, J., Witte, C., Karsch, S., Vranken, H., and Van Eekelen, M. (2013). Evaluation of students' learning behaviour and success in a practical computer networking course. In *Proceedings of the 2nd International Conference on E-Learning and E-Technologies in Education, ICEEE '13*, pages 201–206. IEEE Xplore Digital Library. Cited on pages 16, 17, and 63.
- [Haag et al., 2014a] Haag, J., Witte, C., Karsch, S., Vranken, H., and Van Eekelen, M. (2014a). Evaluation of students' learning behaviour and success as a prerequisite for modernizing practical on-campus networking courses in higher education. *Yükseköğretim Dergisi / Journal of Higher Education*, 4(2):83–90. Cited on pages 16, 17, and 63.
- [Haag et al., 2014b] Haag, J., Witte, C., Karsch, S., Vranken, H., and Van Eekelen, M. (2014b). An exercise assistant for practical networking

- courses. In *Proceedings of the 6th International Conference on Computer Supported Education*, CSEDU '14, pages 97–104. Cited on pages 16, 17, and 77.
- [Hattie and Timperley, 2007] Hattie, J. and Timperley, H. (2007). The power of feedback. *Review of Educational Research*, 77(1):81–112. Cited on page 103.
- [Heeren et al., 2008] Heeren, B., Jeuring, J., van Leeuwen, A., and Gerdes, A. (2008). *Specifying Strategies for Exercises*, pages 430–445. Springer Berlin Heidelberg, Berlin, Heidelberg. Cited on page 77.
- [Horsmann, 2011] Horsmann, T. (2011). Design and implementation of a communication concept for an encapsulated distributed virtual computer security lab. Bachelor's thesis (Studienarbeit), Cologne University of Applied Sciences, Campus Gummersbach, Gummersbach, Germany. Cited on pages 16, 17, 21, and 35.
- [Hu et al., 2005] Hu, J., Cordel, D., and Meinel, C. (2005). Virtual machine management for tele-lab "it-security" server. In *Proceedings of the 10th IEEE Symposium on Computers and Communications*, ISCC '05, pages 448–453. IEEE Computer Society. Cited on page 4.
- [Hu et al., 2004] Hu, J., Meinel, C., and Schmitt, M. (2004). Tele-lab it security: An architecture for interactive lessons for security education. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '04, pages 412–416, New York, NY, USA. ACM. Cited on page 77.
- [Hu et al., 2003] Hu, J., Schmitt, M., Willems, C., and Meinel, C. (2003). A tutoring system for it security. In *Security education and critical infrastructures*, pages 51–60. Springer. Cited on page 77.
- [Jakab et al., 2009] Jakab, F., Janitor, J., and Nagy, M. (2009). Virtual lab in a distributed international environment - svc edinet. In *Fifth International Conference on Networking and Services*, ICNS '09, pages 576–580. Cited on page 3.
- [Jeuring et al., 2014] Jeuring, J., van Binsbergen, L. T., Gerdes, A., and Heeren, B. (2014). Model solutions and properties for diagnosing student

- programs in ask-elle. In *Proceedings of the Computer Science Education Research Conference*, CSERC '14, pages 31–40, New York, NY, USA. ACM. Cited on page 77.
- [Jonkman, 2016] Jonkman, J. H. A. (2016). Botnet Simulation in a Distributed Virtual Computer Security Lab. Master's thesis (Studienarbeit), Open Universiteit, Heerlen, The Netherlands. Cited on page 32.
- [Keller and Naues, 2006] Keller, J. and Naues, R. (2006). Design of a virtual computer security lab. In *Proceedings of the IASTED International Conference on Communication, Network, and Information Security*, CNIS '06, pages 211–215. Acta Press. Cited on pages 3 and 4.
- [Kenny and Pahl, 2005] Kenny, C. and Pahl, C. (2005). *Automated tutoring for a database skills training environment*, volume 37. ACM. Cited on page 77.
- [Krishna et al., 2005] Krishna, K., Sun, W., Rana, P., Li, T., and Sekar, R. (2005). V-netlab: A cost-effective platform to support course projects in computer security. In *Proceedings of 9th Colloquium for Information Systems Security Education*. The Printing House, Inc. Cited on page 4.
- [Lahoud and Tang, 2006] Lahoud, PhD (ABD), H. A. and Tang, PhD, X. (2006). Information security labs in ids/ips for distance education. In *Proceedings of the 7th Conference on Information Technology Education*, SIGITE '06, pages 47–52, New York, NY, USA. ACM. Cited on pages 4 and 6.
- [Li, 2009a] Li, P. (2009a). Exploring virtual environments in a decentralized lab. *SIGITERes. IT*, 6(1):4–10. Cited on page 3.
- [Li, 2009b] Li, P. (2009b). Exploring virtual environments in a decentralized lab. *SIGITERes. IT*, 6(1):4–10. Cited on page 4.
- [Li, 2010] Li, P. (2010). Centralized and decentralized lab approaches based on different virtualization models. *J. Comput. Sci. Coll.*, 26(2):263–269. Cited on page 4.
- [Lo et al., 2014] Lo, D. C.-T., Qian, K., Chen, W., Shahriar, H., and Clincy, V. (2014). Authentic learning in network and security with

- portable labs. *2014 IEEE Frontiers in Education Conference (FIE)*, 00:1–5. Cited on page 3.
- [Mahdi et al., 2016] Mahdi, A., Alhabbash, M., and Abu Naser, S. (2016). An intelligent tutoring system for teaching advanced topics in information security. 2:1–9. Cited on page 77.
- [Melis et al., 2004] Melis, E., Siekmann, J., et al. (2004). Activemath: An intelligent tutoring system for mathematics. In *ICAISC*, pages 91–101. Springer. Cited on page 77.
- [Mhd Wael Bazzaza, 2015] Mhd Wael Bazzaza, K. S. (2015). Using the cloud to teach computer networks. In *Proceedings of the 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, UCC '15, pages 310–314. Cited on page 4.
- [Mory, 2003] Mory, E. H. (2003). Feedback research revisited. In *Handbook of research for educational communications and technology*. Cited on page 103.
- [Odekirk-Hash and Zachary, 2001] Odekirk-Hash, E. and Zachary, J. L. (2001). Automated feedback on programs means students need less help from teachers. *SIGCSE Bull.*, 33(1):55–59. Cited on page 103.
- [O’Leary, 2006] O’Leary, M. (2006). A laboratory based capstone course in computer security for undergraduates. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, pages 2–6, New York, NY, USA. ACM. Cited on page 3.
- [Peltsverger and Zhang, 2014] Peltsverger, S. and Zhang, C. (2014). Bottleneck analysis with netkit: Teaching information security with hands-on labs. In *Proceedings of the 15th Annual Conference on Information Technology Education*, SIGITE '14, pages 45–50, New York, NY, USA. ACM. Cited on page 1.
- [Peterson et al., 2003] Peterson, L., Anderson, T., Culler, D., and Roscoe, T. (2003). A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64. Cited on page 4.

- [Pizzonia and Rimondini, 2008] Pizzonia, M. and Rimondini, M. (2008). Netkit: Easy emulation of complex networks on inexpensive hardware. In *Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, TridentCom '08, pages 7:1–7:10, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). Cited on page 14.
- [Pizzonia and Rimondini, 2016] Pizzonia, M. and Rimondini, M. (2016). Netkit: Network emulation for education. *Software: Practice and Experience*, 46(2):133–165. First appeared online in 2014. Cited on page 14.
- [Pсотka et al., 1988] Pсотka, J., Massey, L. D., and Mutter, S. A. (1988). *Intelligent tutoring systems: Lessons learned*. Psychology Press. Cited on page 77.
- [Queirós and Leal, 2012] Queirós, R. A. P. and Leal, J. P. (2012). Petcha: A programming exercises teaching assistant. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 192–197, New York, NY, USA. ACM. Cited on page 77.
- [Rescorla and Modadugu, 2006] Rescorla, E. and Modadugu, N. (2006). Datagram transport layer security. <http://www.ietf.org/rfc/rfc4347.txt>. [Online; accessed 22-July-2014]. Cited on page 51.
- [Rescorla and Modadugu, 2012] Rescorla, E. and Modadugu, N. (2012). Datagram transport layer security version 1.2. <http://www.ietf.org/rfc/rfc6347.txt>. [Online; accessed 22-July-2014]. Cited on page 51.
- [Rimondini, 2007] Rimondini, M. (2007). Emulation of computer networks with netkit. Technical Report RT-DIA-113-2007, Department of Computer Science and Automation, Roma Tre University. Cited on page 8.
- [Salah, 2014] Salah, K. (2014). Harnessing the cloud for teaching cybersecurity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 529–534, New York, NY, USA. ACM. Cited on pages 4 and 6.

- [Sarkar, 2006] Sarkar, N. I. (2006). Teaching computer networking fundamentals using practical laboratory exercises. *IEEE Transactions on education*, 49(2):285–291. Cited on page 1.
- [Schreiner, 2009] Schreiner, R. (2009). *Computernetzwerke: Von den Grundlagen zur Funktion und Anwendung*. Carl Hanser Verlag GmbH & Co. KG. Cited on page 23.
- [Schürmann, 2004] Schürmann, B. (2004). *Grundlagen der Rechnerkommunikation*. Vieweg+Teubner Verlag. Cited on page 26.
- [Seeling, 2008] Seeling, P. (2008). Labs@home. *SIGCSE Bull.*, 40(4):75–77. Cited on pages 4 and 6.
- [Stevens, 2003] Stevens, W. R. (2003). *Unix Network Programming, Volume 1: The Sockets Networking API*. Addison-Wesley Professional, 2nd edition. Cited on page 24.
- [Suraweera and Mitrovic, 2004] Suraweera, P. and Mitrovic, A. (2004). An intelligent tutoring system for entity relationship modelling. *International Journal of Artificial Intelligence in Education*, 14(3, 4):375–417. Cited on page 77.
- [Sykes and Franek, 2003] Sykes, E. R. and Franek, F. (2003). A prototype for an intelligent tutoring system for students learning to program in java (tm). In *Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education*, pages 78–83. Cited on page 77.
- [Tanenbaum, 1985] Tanenbaum, A. S. (1985). *Computer Networks*. Prentice Hall PTR, Upper Saddle River, NJ, USA. Cited on page 90.
- [Taylor et al., 1996] Taylor, K. D., Honchell, J. W., and DeWitt, W. E. (1996). Distance learning in courses with a laboratory. In *Proceedings of the 26th Annual Frontiers in Education - Volume 01, FIE '96*, pages 44–46, Washington, DC, USA. IEEE Computer Society. Cited on page 3.
- [Vanlehn et al., 2005] Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M.

- (2005). The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3):147–204. Cited on page 77.
- [Vesin et al., 2013] Vesin, B., Ivanović, M., Klašnja-Milićević, A., and Budimac, Z. (2013). Ontology-based architecture with recommendation strategy in java tutoring system. *Computer Science and Information Systems*, 10(1):237–261. Cited on page 77.
- [Vranken et al., 2011] Vranken, H., Haag, J., Horsmann, T., and Karsch, S. (2011). A distributed virtual computer security lab. In *Proceedings of the 3rd International Conference on Computer Supported Education, CSEDU '11*, pages 110–119. SciTePress. Cited on pages 16, 21, 35, and 96.
- [Vranken and Koppelman, 2009] Vranken, H. and Koppelman, H. (2009). A virtual computer security lab for distance education. In *Proceedings of the 5th IASTED European Conference on Internet and Multimedia Systems and Applications, EuroIMSA '09*, pages 21–27. Acta Press. Cited on pages 4, 5, 13, 16, and 21.
- [Wielemaker, 2009] Wielemaker, J. (2009). *Logic programming for knowledge-intensive interactive applications*. PhD thesis, University of Amsterdam. Cited on page 96.
- [Xu et al., 2012] Xu, L., Huang, D., and Tsai, W.-T. (2012). 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, ITiCSE '12*, pages 256–261, New York, NY, USA. ACM. Cited on page 1.
- [Yang, 2011] Yang, F.-J. (2011). A virtual tutor for relational schema normalization. *ACM Inroads*, 2(3):38–42. Cited on page 77.
- [Yang and Reddington, 2014] Yang, J. and Reddington, T. (2014). Enhance learning through developing network security hands-on lab for online students. In *Proceedings of the 2014 Information Security Curriculum Development Conference, InfoSec '14*, pages 11:1–11:1, New York, NY, USA. ACM. Cited on page 1.

- [Yang et al., 2004] Yang, T. A., Yue, K.-B., Liaw, M., Collins, G., Venkatraman, J. T., Achar, S., Sadasivam, K., and Chen, P. (2004). Design of a distributed computer security lab. *Journal of Computing Sciences in Colleges*, 20(1):332–346. Cited on page 3.
- [Yoo and Hovis, 2004] Yoo, S. and Hovis, S. (2004). Remote access inter-networking laboratory. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '04, pages 311–314, New York, NY, USA. ACM. Cited on page 3.
- [Yuan, 2017] Yuan, D. (2017). Developing a hands-on cybersecurity laboratory with virtualization. *J. Comput. Sci. Coll.*, 32(5):118–124. Cited on page 1.
- [Yuan et al., 2011] Yuan, D., Cody, L., and Zhong, J. (2011). Developing ip telephony laboratory and curriculum with private cloud computing. In *Proceedings of the 2011 Conference on Information Technology Education*, SIGITE '11, pages 107–112, New York, NY, USA. ACM. Cited on page 1.

Curriculum Vitae

- 1976 Born in Düsseldorf, Germany
- 1987 - 1993 **Gesamtschule der Stadt Kierspe (GSKi)**
Secondary School
- 1993 - 1996 **Gesamtschule der Stadt Kierspe (GSKi)**
Extended Secondary School
- 1997 - 2008 **Cologne University of Applied Sciences**
Higher Education
- 1998 **he-microsystems**
Student Associate
- 1999 - 2001 **Babcock GmbH**
Student Associate
- 2001 - 2003 **Pickard+Heffner GmbH**
Student Associate
- 2006 - 2008 **Cologne University of Applied Sciences**
Student Associate
- 2008 - 2014 **Cologne University of Applied Sciences**
Academic Staff Member
- since 2012 **Open Universiteit in the Netherlands**
External Doctoral Candidate
- since 2015 **Federal University of Applied
Administrative Sciences**
Lecturer in Information Technology