Technische Universität München

Department of Informatics

Interdisciplinary Project in Electrical Engineering

Virtual iLab Isle

Moritz Sichert

Technische Universität München

Department of Informatics

Interdisciplinary Project in Electrical Engineering

Virtual iLab Isle

Virtuelle iLab-Insel

Author	Moritz Sichert
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dr. Marc-Oliver Pahl
Date	October 27, 2017

Informatik VIII
Chair for Network Architectures and Services

Abstract

The iLab courses are practical lab courses that give students a very deep understanding of how the internet works. In that course students work on a so-called iLab Isle which is a work space that has all hardware that is needed to work on the practical exercises. In the future a MOOC called iLab^{x} that is very similar to the iLab will also be offered by the TUM. Because students can't be expected to have access to such special hardware that is available on the iLab Isles at home, we create a system – the vLab OS – that virtualizes all of the hardware of an iLab Isle. This system is very similar to the iLab OS that runs on the computers on the iLab Isle so that students used to working on an iLab Isle will have no difficulty using the vLab OS.

Zusammenfassung

Die iLab-Praktika bieten Studenten die Möglichkeit, ein tiefes Verständnis über die Technologien im Internet aufzubauen. Im Rahmen des Praktikums arbeiten die Studenten an einer sogenannten iLab-Insel. Die iLab-Insel ist ein Arbeitsplatz, an dem alle Hardware, die für die praktischen Aufgaben benötigt wird, verfügbar ist. In der Zukunft wird die TUM auch den MOOC iLab^{x} anbieten, der dem iLab sehr ähnlich ist. Da nicht erwartet werden kann, dass Studenten zu Hause Zugriff auf Hardware haben, die der in einer iLab-Insel verfügbaren gleichwertig wäre, erstellen wir ein System – das vLab OS –, das die gesamte Hardware einer iLab-Insel virtualisiert. Es ist dem iLab OS, das auf den Rechnern der iLab-Inseln läuft, sehr ähnlich, sodass Studenten, die das iLab OS gewohnt sind, keine Schwierigkeiten haben sollten, das vLab OS zu verwenden.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For many years the Chair for Network Architectures and Services of Technische Universität München (TUM) has offered the practical courses “iLab” and “iLab 2” with great success [1]. Both are lab courses that cover many technologies used in the internet today. The topics students learn in the course range from low-level details of how networking works on the link layer (e.g. LAN and WLAN) to very high-level services that are very important for the internet and used by many people every day (e.g. HTTP and e-mail). They aim to give students in-depth knowledge about most or all of those technologies while also offering weekly practical hands-on exercises. After participating in the iLab courses the students will not only have a very deep theoretical background about how the internet works but will also be able to set up new or work on existing real-life scenarios like home networks but also large scale ISP scenarios.

Independent from the iLab courses TUM started a MOOC (Massive Open Online Course) program in 2017 that is aimed at but not exclusive to prospective students of a master’s degree of computer science. Because the iLab courses proved to be very successful, the Chair for Networking Architectures and Services is currently working on an adaptation of the iLab as a MOOC called iLab^{x} [2]. This MOOC will be very similar to the iLab course and in particular also have practical exercises.

The main part that makes the iLab unique is that students can directly experience working on large network setups. For this they have access to a lab room that is equipped with all the hardware needed for the practical exercises. The iLab^{x} MOOC aims to keep the spirit of the iLab courses so that students will be just as motivated to work on it. In this Interdisciplinary Project we created a system called “vLab OS” which allows all participants of the iLab^{x} MOOC to work on its practical exercises, regardless of how powerful their computer at home is or which operating system they use.

1.1 Outline

The thesis is structured as follows. In Chapter 2 we analyze the current situation of the iLab Isle and how it can be virtualized. Chapter 3 presents the iLab OS which this work is largely based on. In Chapter 4 we present the vLab OS that was developed in this Interdisciplinary Project. There we show the difference and the new features of the vLab OS and document how it can be built and run. In the end in Chapter 5 we describe how the vLab OS can be improved in the future.

Chapter 2

Analysis

Currently, in the iLab courses students work on their practical exercises in a lab room. This room has multiple so called “iLab Isles” which is the main workspace for all participants of the course during a semester. Figure 2.1 shows such an isle. As students typically work in groups of two, each iLab Isle consists of two full workspaces side by side. Because many exercises try to emulate complex real-world network setups, each iLab Isle is equipped with special hardware. The most important component is the iLab PC. An iLab PC has an integrated ethernet port and an extension card that adds four more ethernet ports. It also has another extensions card for wireless LAN. To ensure that students can fully focus on their exercises the iLab PCs use a custom Debian-based Linux system called “iLab OS”. It comes with all software that is needed for the exercises pre-installed. As already mentioned, many exercises simulate large network setups, so each student has three iLab PCs for a total of six per iLab Isle. On each side there is also a Cisco router which helps simulating ISPs and of course LAN switches to be able to connect the hardware in many different ways.

On the other hand, students participating in the iLab^{X} MOOC will work on the exercises on their own computer at home. This means that among all participants there will be a large variety of operating systems used, at least Windows, Mac OS X and Linux should be expected. Also most home computers are hardly comparable to an iLab PC in terms of their networking capabilities. Despite all of this, the practical exercises in the iLab^{X} MOOC should not differ much from the ones in the iLab courses. A goal of the iLab^{X} MOOC is to keep the appeal of the iLab which lets students experiment on large networking scenarios they don’t usually come in contact with.

To achieve this, a system is needed which allows students to work on a Virtual iLab Isle. Specifically, this means that they should be able to start different PCs, connect them to each other and to network switches, and analyze all network traffic. Such a system could also not only be used for the iLab^{X} MOOC but would also be beneficial for students of the iLab courses that want to continue experimenting on setups they

Figure 2.1: Workspace for students of the iLab course called “iLab Isle”: Students typically work in groups of two, so each each isle offers two full workspaces side by side. To be able to simulate complex network setups each isle also has six special iLab PCs with five ethernet ports each and two Cisco routers.

started on an iLab Isle at home. We call this system vLab OS.

2.1 Virtualization as Common Ground

As already mentioned, the vLab OS will eventually run on many different computers. This means that at least the major operating systems Windows, Mac OS X and Linux should be supported. There are different ways to achieve this. Creating separate versions of the vLab OS – one for each operating system – is an obvious solution but is not really feasible. Apart from the obvious overhead of managing three separate versions, another issue is also that the three operating systems are also fundamentally different in how they realize virtualization of networks and computers. For the vLab OS it would be desirable to have a common ground, i.e. an environment that is available on every computer. Using virtual machines is a perfect candidate for this. Programs that run virtual machines (e.g. VirtualBox, VMware) are available for all three operating systems, so providing the vLab OS as a virtual machine means that a single version of it can run on all operating systems without requiring any changes.

2.2 Operating System

Because virtual machines are essentially entire computers in themselves, this means that the vLab OS must be an entire operating system and not only a single program. Eventually, the vLab OS will be distributed to all participants of the iLab^{x} MOOC.

Because the course should be free of cost for anyone, this leaves Linux as only sensible choice for the operating system the vLab OS will use. Fortunately, Linux supports creating virtual ethernet interfaces and switches. Together with its capabilities to create containers this further strengthens our choice of using Linux. Being able to run containers is very useful as they are basically a very light-weight alternative to virtual machines. They behave like separate machines on the same system that only share the kernel but are otherwise mostly independent from each other, which allows to emulate many computers without requiring powerful hardware.

2.3 Linux Distribution

There are many choices of Linux distributions which make good candidates for the vLab OS. The distribution of choice should be easy to use and have all packages related to networking that could be useful for practical exercises. In the iLab courses the question of which distribution to use was solved by creating a new, custom one called “iLab OS”. It is based on Debian and is tailored to the needs of the iLab: It has all programs needed for the practical exercises pre-installed and is configured so that students can sit down on an iLab Isle and directly start working on it. As such it also a good choice for the vLab OS. As the vLab OS is mostly based on it, we describe it in more detail in Chapter 3.

2.4 Emulation of Network Setups

Te main feature needed for the vLab OS is of course the virtualization of the iLab Isle. As already mentioned, Linux supports virtual interfaces and containers. This makes a good backend for the network virtualization. Still, manually creating the virtual interfaces and containers requires a lot of knowledge about Linux and advanced usage of the terminal which would make working with the vLab OS unnecessary hard. Instead, we want a preferably graphical interface in which students can easily create and connect new virtual PCs and switches. We found two possible candidates for this task: the systemd-nspawn tool that is part of the systemd init system [3] and the Common Open Research Emulator (CORE) [4]. Table 2.1 shows the differences, advantages, and disadvantages of both programs. To summarize, the main plus point of systemd-nspawn is that it is built-in into systemd which is installed already, so no new packages are needed. On the other hand CORE comes with an easy to use GUI in which new PCs, switches, and connections can be created by just clicking. For this reason we choose to use CORE in the vLab OS.

	systemd-nspawn	CORE
Quick Facts	• included in systemd • LGPL License • written in C • low level command line tool	• from U.S. Naval Research Laboratory • BSD License • written in Python and C • high level GUI
Configuration of Scenarios and PCs	Scenarios can be created in the GUI and saved to XML or JSON-like files. More complex setups can be created with shell scripts that run in the virtual PCs.	Write shell scripts that create network interfaces, bridges, etc., and start all PCs and configure individual PCs with shell scripts as well.
Starting and Stopping PCs	By clicking on them in the GUI. Persistent configuration must be configured explicitly.	Again, run a shell script to start and stop containers. Reboot is possible without losing changes.

Table 2.1: Comparison between systemd-nspawn and CORE

2.5 Summary of Requirements

In this chapter we analyzed the requirements of a system that emulates the iLab Isle workspace. They can be summarized as follows:

- Must run on all major operating systems, so virtual machines are used
- Operating system running in the virtual machine must be free and support efficient emulation of networks, so use Linux
- Needs easy-to-use interface to create network setups, use CORE

Lastly, we have a few additional technical requirements for the vLab OS. As it should be able for all interested students to run the vLab OS, it should require as little system resources as possible and run on low-end hardware. Also, since the vLab OS is essentially an entire operating system, it could potentially be a large download that would be required for each student. To make it possible to participate in the iLab^{x} MOOC even with a low-bandwidth internet connection, we want the vLab OS to be as small as possible. A file size of 2 GB or smaller is desirable which should be possible to be downloaded within a few hours even with slow internet connections.

Chapter 3

How the iLab OS Works

As mentioned earlier, the iLab OS is based on the Debian Linux distribution [5]. Debian and the iLab OS itself are open source and can be freely used. The iLab OS consists of two components: The client image which is the actual disk image that the iLab PCs boot from and the iLab control server. There is also a short documentation about both components. All URLs for the iLabOS can be found in Table 3.1.

iLab OS documentation:	`https://ilab.net.in.tum.de/ilabos/`
iLab OS client image git repository:	`git://git.net.in.tum.de/ilab/image.git`
iLab OS control server git repository:	`git://git.net.in.tum.de/ilab/server.git`

Table 3.1: URLs to iLab OS documentation and components

The client image is the most important component of the iLab OS. It is essentially a disk image of a Debian installation with many additional packages. Among those the most important ones are GNOME (desktop environment), Wireshark (network sniffer) and several internet service programs, like an IPv6 router advertisement daemon, a DNS server, and many more. It also comes configured so that is convenient to use for participants of the iLab courses. The main differences between the iLab OS and a normal Debian installation is the factory reset functionality. Because over the course of a week many students will work on the same iLab Isle, every time an iLab PC is rebooted it is reset to its initial state unless the student explicitly chooses otherwise. This means students can freely change anything on a running iLab PC without having to fear a crash. More details about how the factory reset works are explained in Section 3.2.

The other component is the iLab control server. It is mainly used to provide the client image to the iLab PCs that use network boot. The control server is connected to all iLab PCs in the lab room in the so called management network. Because the vLab OS does not use a control server, we will not describe it here but focus on the client image.

3.1 Building the iLab OS

To build the iLab OS client image the source repository (see "iLab OS client image git repository" in Table 3.1) is needed. Table 3.2 shows a description of the most important subdirectories in the source repository.

Subdirectory	Description
`auto/`	Contains scripts that are used to create the actual image in the end. In particular the script `auto/build` uses the Debian image builder program `live-build` to create it.
`config/`	All configuration that affects how the client image will be built and actual files that should be included in it are located here.
`config/hooks/`	This directory contains scripts that will be run inside the new client image at the end of building it. Those scripts disable the automatic startup of several services and configures the `ilab` user for example.
`config/includes.chroot/`	The contents of this directory are copied as is into the client image by the build script. It contains several scripts for the initramfs system that customize the boot process (described in more detail in Section 3.2) and a few customizations for the iLab OS, like a custom start page for the browsers.
`config/package-lists/`	This directory contains a list of apt packages that are installed in the client image. This list contains many programs useful for networking, e.g. Wireshark, radvd, bind9, etc.
`config/preseed/`	This directory is also used by apt while building the client image. It contains preselected configuration options for choices that are normally given to the user when installing Debian.

Table 3.2: iLab OS client image source repository subdirectories

As already mentioned, the program to create the image is called `live-build`. It was originally made by the Debian project to make their live boot installer images. Since a live boot, i.e. a system that configures itself every time it is booted, is exactly what is needed for the iLab OS, it makes sense to use this existing software.

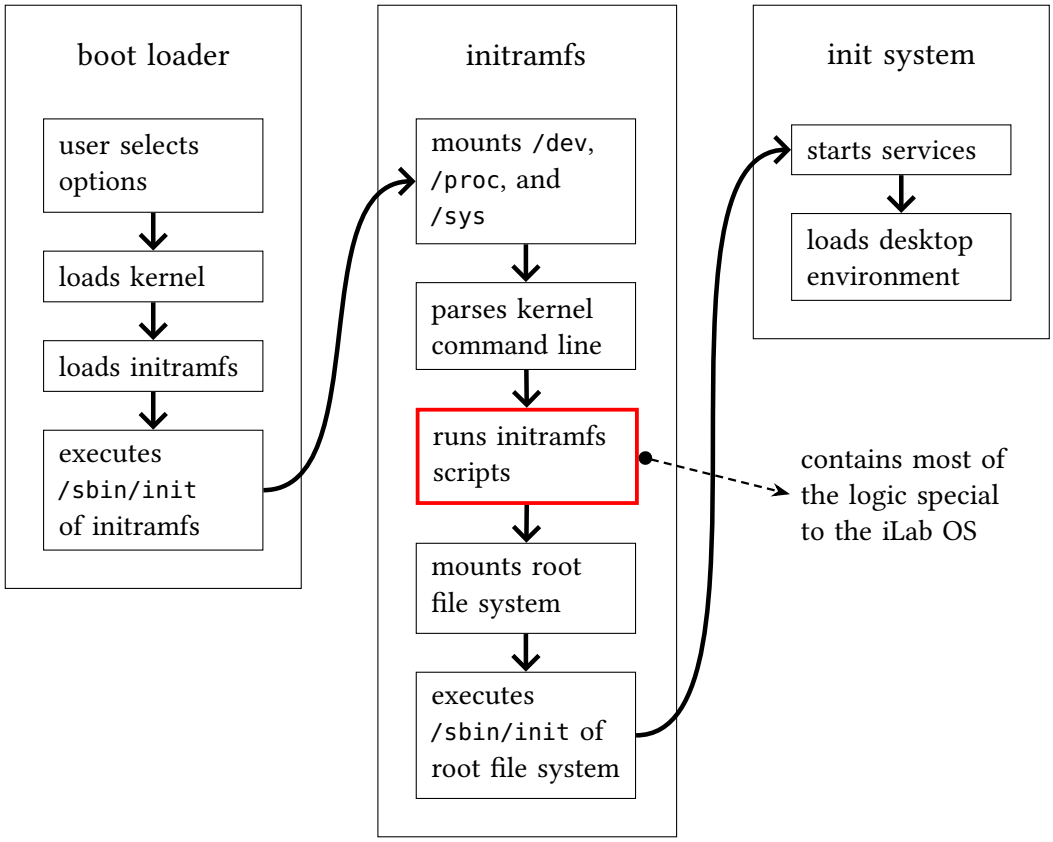

Figure 3.1: Boot process of the iLab OS: It is very similar to the boot process of any other Linux system. Starting with the boot loader it first loads the kernel and the initramfs. The initramfs then does a first initialization of the system and eventually starts the system by executing the init daemon. Most logic that differentiates the iLab OS from any other Linux system is contained in the initramfs scripts.

3.2 Boot Process

To implement the factory reset function and to give the user a readily usable system after booting (i.e. no need to login or provide any password) the iLab OS needs to customize the boot process. An overview of the most important steps of the boot process can be seen in Figure 3.1.

As in most Linux systems shortly after powering on the boot loader is executed. It usually loads the kernel and initramfs from the disk into memory. The initramfs is a file system that lies entirely in memory and contains files that are needed to load the actual root file system. It is also possible to execute scripts there that can affect how the root file system is mounted. This is exactly the place where the factory reset function can be implemented.

The boot loader starts the kernel which itself starts by executing `/sbin/init` from the initramfs. There, first the basic virtual filesystems `/dev`, `/proc`, and `/sys` are mounted. Then the kernel command line which can be read from `/proc/cmdline` is parsed for any options that affect the boot process in the initramfs. After this comes the most important part for the iLab OS: the initramfs scripts are executed.

In the iLab OS one of the initramfs scripts contains the logic for the factory reset function. It checks if the string `ilab-reset` appears in the kernel command line. If it does, the system is reset to its original state. The boot loader is configured to always include this string in the kernel command line unless the user explicitly chooses an option to not reset the system.

Then the root file system is mounted and the real init daemon (systemd) is started. It starts all configured services, which includes the `live-boot` service that comes with `live-build` mentioned earlier. `live-boot` takes care of configuring the system and logging the user in automatically. Lastly, the desktop environment is started and the iLab OS is ready to use.

Chapter 4

The vLab OS

The iLab OS works very well in the iLab courses. It allows students to work on their practical exercises and integrates well with all the hardware available in an iLab Isle. It is not designed to run anywhere else but on the iLab PCs, however. This means that for students that want to work on exercises at home and especially students from all over the world that are participating in the iLab^{x} MOOC there is no easy way to use the iLab OS. So a new version of it is needed which allows students to create all network setups that could also be created on an iLab Isle. We call this new version vLab OS. It essentially virtualizes an entire iLab Isle while trying to stay very similar to the iLab OS. To use it students only have to download a virtual machine appliance and run it on a virtualization program (e.g. VirtualBox, VMware).

Just like the iLab OS the vLab OS is open source and can be freely used. The source code can be found in the iLab OS git repository (see Table 3.1) in the `vlab` branch. There is also a blog which contains how-to's related to the vLab OS and also links to download the vLab OS as virtual machine image. It can be found at `https://vlab.net.in.tum.de/`.

4.1 Features of the vLab OS

As the vLab OS is based on the iLab OS, they share many features. Especially all programs related to networking that are pre-installed in the iLab OS are as well in the vLab OS. The goal is that students of the iLab courses don't have to learn any new programs when using the vLab OS. However, one goal of the vLab OS is also that the downloadable image stays as small as possible. For this reason we decided to replace the Gnome desktop environment with XFCE. As a direct result the image size went down from over 3 GB to 2.5 GB. Further optimizations allowed us to shrink the image size down to 1.4 GB. They are described in more detail in Section 4.2.

Just like the iLab OS, the vLab OS also offers a factory reset functionality. Because the

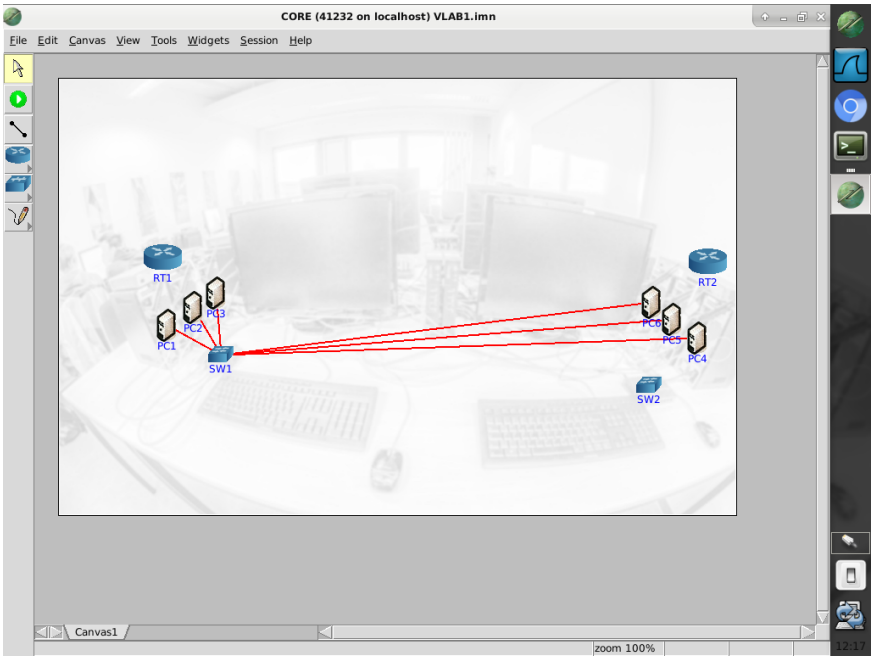

Figure 4.1: Screenshot of the vLab OS with CORE: The user interface of the vLab OS is limited to only the programs students will need: CORE, Wireshark, a browser, and of course the terminal. In CORE an example for a simple network setup is loaded.

vLab OS is geared towards students working with it at home instead of being installed on a computer that many students use, the default option in the vLab OS is to *not* reset it when booting. Still, when the user chooses the factory reset option when booting, the virtual machine will reset itself to its initial state. The factory reset is implemented so that it doesn't need to download the whole disk image. It uses snapshots of the BTRFS file system instead so that the factory reset can be done offline. More details about this can be found in Section 4.3.

As mentioned in Chapter 2 already, we use CORE for the network emulation. A screenshot of the vLab OS with CORE can be seen in Figure 4.1. Because CORE is open source as well, this allowes us to slightly customize it to our needs. In the original version CORE gives random, incomprehensible names to the virtual network interfaces it creates for the virtual PCs. Because we want students to easily find the correct network interfaces, we changed this so that CORE chooses descriptive names, instead. Our patches to CORE can be found here: `https://gitlab.dev.ds2os.org/mooc4masters/core-network`.

4.2 Building the vLab OS

The directory structure of the vLab OS source repository is the same as for the iLab OS as described in Table 3.2. In fact it is intended to be the same so that future updates to

the iLab OS that would also benefit the vLab OS can be easily merged into it.

Instead of `live-build` that was mentioned in Section 3.1 the vLab OS is built with a Python script called `create-vm.py`. The script runs only on Linux and requires the following programs to be installed:

- Python (version 3.5 or newer)
- systemd (in particular systemd-nspawn is needed)
- btrfs-progs
- debootstrap
- rsync

When the script is executed, it starts by installing a base Debian system with debootstrap in a temporary directory. Because debootstrap needs root permissions to run, the script must be executed as root, as well. When this is finished, the temporary directory will contain a very minimal system that does not even have a kernel installed.

So the next step is to install all packages. This is done by first copying the contents of `config/apt/` to `etc/apt/` in the temporary directory. Then all packages listed in `config/package-lists/` are installed by using systemd-nspawn. With systemd-nspawn a temporary container is created in which the installation can be executed so that the outer system (i.e. the one where the Python script is running on) is not affected by any services that are installed. In particular this prevents unwanted behaviour when the installer tries to start or stop services that exist on the outer system. In this step all packages are downloaded from a Debian mirror and are directly installed, so it takes several minutes.

When all packages are installed, all files from `config/includes.chroot/` are copied to the temporary directory. Just like in the iLab OS those are mainly configuration files that customize the appearance of the system. Most importantly in this step all custom initramfs scripts needed for the vLab OS are copied as well. How they work exactly is explained in Section 4.3.

Next the kernel is installed. It is not installed in the earlier step together with all other packages because in the previous step some files that affect how the kernel is installed are copied. Namely the file `/etc/kernel-img.conf` in which a configuration option is set that makes the installer create a symbolic link to the kernel. This is important for the boot scripts to find the kernel.

After that all scripts found in `config/hooks/live/` are executed in the new system, again with the help of systemd-nspawn. As when building the iLab OS those scripts disable services that are only needed for some exercises, create an `ilab` user, etc. When this is done, the new system is almost entirely configured.

Only then the actual disk image is created. Until this point the whole system is only installed into a temporary directory. In this step the disk image is prepared by partitioning and formatting and then mounting it. Then all files from the temporary directory are copied into the mounted disk image. The reason why the previous steps are not executed directly on the disk image has to do with making the disk image smaller. In the previous steps many packages are installed which means that many files are created but also moved around and removed again. Especially removed files are problematic for the size of the final disk image because modern file systems remove a file by just removing some inodes from a list. This means in particular that the data of removed files is still there and can be reused for new files later but is not empty (i.e. only bytes with value 0). Then, when the disk image is compressed to its final version that must then be downloaded by students, this makes a huge difference in space. When files are only copied to the disk image, no files are ever removed or moved so this issue does not arise. We managed to reduce the size of the compressed disk image from 2.5 GB to 1.4 GB by doing this.

In the end the fstab is created and the boot loader is installed onto the disk image. Then it is ready to be used as raw disk image in a virtualization program.

4.3 Boot Process

The boot process of vLab OS works very similar as in the iLab OS. The structure is still as shown in Figure 3.1. Also most components described in Section 3.2 work the same in the vLab OS. One difference is how the factory reset function is implemented.

The root file system of the vLab OS is formatted with the BTRFS file system. When it is booted for the first time, the initramfs script in the vLab OS creates a snapshot of the root file system. This means that the original state of the file system will still be available later. Indeed, when the user chooses the factory reset option, the current snapshot is removed and a new snapshot is created. Because the main file system from which snapshots are created is never directly used, this way it is always possible to reset the system to its original state.

The vLab OS also has one additional service that is started in the init phase after the initramfs: the VirtualBox Guest Additions. Because we expect most iLab^x MOOC participants to run the vLab OS on VirtualBox we want to ensure the best compatibility with it. So when the vLab OS detects that is started in VirtualBox, it will install the VirtualBox Guest Additions (without needing an internet connection).

Chapter 5

Conclusion

In this Interdisciplinary Project we created the vLab OS which fulfills all the goals mentioned in Chapter 2: It can be downloaded as single file with moderate size (1.4 GB) which should be possible to download within a few hours with most internet connections. Also, it is a virtual machine appliance that can be run by different virtualization programs like VirtualBox and VMware that themselves run on all major operating systems. Most importantly it achieves the goal to let students run entire network setups at home very well by using the CORE network emulator.

5.1 Future Work

For the factory reset there is a possible improvement that could be implemented in the future. The idea of the factory reset is that the virtual machine can be reset to a working state again regardless of what a user did. But currently the kernel and initramfs files in the `/boot` directory are not protected. This means that if they are deleted, the system will not boot anymore, not even when the factory reset option is chosen. While it is unlikely for someone to accidentally delete any of those files, it would still be better if the vLab OS was not vulnerable to this. Protecting the boot file system just like the root file system, i.e. by creating a snapshot of it in the initramfs, could solve this problem.

From the usability perspective it would also be good to make the vLab OS even easier to use for people who never used Linux before. All functionality that is needed for a practical exercise of the iLab^{x} MOOC should be reachable with one mouse click. Ideally the users should have a “gaming console experience” in that they don’t have to know the operating system in detail but can instead focus on the tasks.

When talking about the iLab^{x} MOOC one could also add a mechanism into the vLab OS that automatically loads the latest information about the course. An overview of all

past, present, and future tasks that can be reviewed and started with one click could also add to the “gaming console experience”.

Bibliography

[1] “iLab – build your own Internet.” [Online]. Available: http://ilab.ds2os.org/?site=teaching/ilab

[2] “iLabX – the Virtual Internet Laboratory.” [Online]. Available: http://ilab.ds2os.org/?site=mooc4masters

[3] “systemd System and Service Manager.” [Online]. Available: https://www.freedesktop.org/wiki/Software/systemd/

[4] “Common Open Research Emulator (CORE).” [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/core

[5] “debian – The universal operating system.” [Online]. Available: https://www.debian.org/