Using Virtual Machines in System Administration Education

Kyrre Begnum∗ Karst Koymans† Arjen Krap‡ John Sechrest§

Abstract

System and network administration education faces a significant challenge supplying students with both theory and hands-on practical experience. Students need real systems with root access in order to install and test popular services as a basis for learning key issues through experience.

We present how networks of virtual machines can be used in education to provide the necessary environments for students to work on and expect to see cost savings, more challenging student assignments, a protected test environment for every student or group and better scalability for larger classes and most importantly, the ability to reconfigure and restore networks rapidly.

This paper also presents a tool for the building and administration of virtual networks based on User-mode Linux[1]. This effort is a joint venture between lecturers from the University College of Oslo, University of Amsterdam, University of Linköping and other experienced lecturers in the field of system and network administration.

1 Introduction

System and network administration is the process of bridging the structure and configuration of a set of computers with the needs and skills of the people using the systems. To truly understand the issues around system administration, it is vital to get specific experience with both how systems work and how people use them. A theoretical basis for study does not provide adequate understanding to do system and network administration. It is important to have hands on experience with configuration, deployment and maintenance of computer systems. Achieving this experience in a university environment can be challenging. But it is vital to good education that students have hands on systems to work with in order to install and test services and to learn from the practical experiments on these specific systems.

At the University College of Oslo and the University of Amsterdam, courses are taught in the field of system administration. The key for system administration education is to give students a realistic testbed to test advanced distributed services and configurations. This project is a joint venture with other universities and experienced lecturers in the field of system administration education.

Our goal is to:

Give students and researchers a way to create and use a protected and versatile virtual network environment where they can install services and solve problems related to the subjects of system and network administration.

∗Faculty of Engineering, Oslo University College, Norway
†Informatics Institute, University of Amsterdam, The Netherlands
‡Informatics Institute, University of Amsterdam, The Netherlands
§Alpha Omega Computer Systems, USA
We explore two approaches to using systems in system and network administration courses. The first approach uses a generic filesystem that configures itself based on boot parameters. The second approach is a virtual machine administration tool called MLN that can be used to configure and build large virtual networks based on User-mode Linux using an abstraction layer in the form of a configuration language[2].

Our approach shows how we were able to save money and space and at the same time create the opportunity to give students more complex assignments without an additional cost of time.

This paper is not only intended for people in the field of system administration education, but also for system and network administrators who need a realistic testbed for applications, technologies and configurations without affecting their real network.

2 Educational Context

As the use of the Internet expands into more areas, the field of system and network administration is becoming more important as well. As this grows, the education of new administrators grows in momentum too. Several schools offer courses on system administration and a few, like the University College of Oslo and the University of Amsterdam, also offer masters degrees in system and network administration.

Network labs are set up to provide environments where the students can learn and solve problems. Entire rooms are often allocated for this purpose and contain different network hardware as well as a number of machines. It depends on the specific course and the students experience whether the lab has been set up ahead of time or if they have to do it themselves.

Network Labs for system administration courses require the following features:

- Students need root/administrator access to their test systems in order to accomplish their tasks
- Assignments will need flexible lab scenarios that are suitable for both small to longer periods of time and that might change every term
- Course support teams need a way to remember configurations in order to repeat the procedure for new students
- The labs need to scale so that the assignments work for many students at a time

Although staff works hard to set up labs which meet these criteria, it is difficult to achieve. These labs are usually difficult to manage successfully. The different courses and the number of machines they might need lead to conflicting requirements. Shortages of available space and finances are a concern and limits the number of students in a lab and the amount of available equipment. A typical experience was a course teaching firewalls and intrusion detection. A lab of 15 hosts simulating a simple network was deployed where each student group had one machine for themselves. Running this lab for nearly six months had the following challenges:

- A whole room had to be allocated for the physical network for the duration of the course. Space is often one of the most expensive assets of a school. Only the course staff got key-access to that room.
- Hardware is expensive and the labs had to be limited in size and consisted of old hardware.
- Students pointed out that the networks in the lab were too “simple” with regard to topology and the tools they tested could therefore not be used to their full potential.
• Some of the old hardware broke down during the course and had to be replaced. This applied an additional workload on the course staff.
• Students needed physical access to the lab to do rescue operations on their systems. This access had to be supervised by course staff.
• It took a long time to build the lab and to configure all machines properly beforehand. Different hardware components on the machines contributed to this.
• The lab was difficult to scale. We had limited access to more machines and network switching hardware.

Additionally, faculty and staff are often too busy to set up new labs and install new test machines for the students from the ground up.

Virtual networks can solve some of these challenges. Grid computing is starting to explore virtual machine based services[3] and virtual appliances are being explored for commercial service models[4]. This suggests that a virtual network based lab can provide a more complex network topology at a lower cost than a physical based lab. A significant savings can be realized when the whole virtual lab runs on a single machine. This machine can stand in the college’s server room and won’t take up much space. There is no need for student access to the main server in order to recover from problems. And so, the old network lab room can now be reallocated.

Some aspects of system administration fit poorly into a simulated environment. For example, specific hardware configurations and high performance systems are difficult to simulate on virtual machines. Virtual networks are a valuable addition to our field and can take over some parts of the educational context, but must be integrated into the project in an appropriate manner.

3 User-mode Linux

User-mode Linux (UML)[1] is a project where the GNU/Linux kernel is modified to run in user space, like a regular process. It runs and behaves like a kernel as compared to instruction set emulators, like VMware or Virtual PC which can run any kernel. UML is bound to Linux where VMware and Virtual PC can allow for arbitrary Operating systems in each Virtual Machine Instance, but it comes at the expense of additional resource demands and at the ability to automate complex configurations.

It is possible to start several User-mode Instances as normal processes running on the same physical machine. It is similar to the jail environment found in FreeBSD, but it provides a completely separate kernel, allowing each Virtual Machine Instance to run completely different configurations. UML provides a mechanism to automate the set up and maintenance of several instances. With several Linux projects allowing a Linux system to run as a router, UML can be used to set up a complex network of several nodes.

User-mode Linux provides several ways to connect virtual machines together and to the real world. Part of the available tools is a switch emulator called uml_switch, routing packages and virtual networks. These can be used to create large networks of arbitrary topology. Through the use of Linux bridging software, these networks may be set up with a link to the real internet or to make the virtual machines appear on the local network.

User-mode Linux is run from the command line where system properties like memory and attached disks can be supplied as command line options. This makes it possible to automate the process of starting and stopping many hosts in the background. It also solves the need for a login account for the students on the servers which host
the virtual networks. They can access the virtual machines directly using SSH over the network without login into the server first.

Every host instance in the virtual network has their own filesystem image (as a file) or a directory which they regard as their root filesystem. Several hosts may even share filesystems read-only and write their own changes to a separate file (using copy-on-write). Since filesystems can be kept small as well as memory usage, it allows an average workstation to run several light weight virtual networks.

In order to build networks of virtual machines, each virtual machine needs to be configured. There needs to be a way to start and stop all of the virtual servers, either together or individually. This can be a cumbersome task when done by hand. This is a configuration management task, and can be addressed in different ways. The next section will describe the approach taken at the University of Amsterdam, where configuration management features where built into the filesystem used by the virtual machines.

4 UML used at the University of Amsterdam

User-mode Linux is used at the Master of Science program in Network Administration at the University of Amsterdam¹. "Internetworking and Routing" is an eight week course where the first six weeks have lectures and practical assignments. INR is about layer 2 (bridging and switching) and layer 3 (routing) networking. The practical assignments match the subjects of the lectures².

4.1 Technical details

Each student who follows this course has to create networks with multiple routers or bridges. It is very expensive to acquire enough hardware for each student, especially for use during a short six week period. Since Linux can function as a router as well as a bridge, User-mode Linux is an ideal platform to replace real routers and bridges for this purpose.

The focus of the INR course is on theoretical knowledge, not on specific practical experience. Therefore it does not matter whether we use free software or commercial hardware. It only has to work correctly, as specified in the RFC’s. User-mode Linux runs the same software as regular Linux. This makes UML suitable for almost every task. For INR it was possible to install network diagnostic software like tcpdump and ethereal, which is not available on hardware based routers and bridges.

One server was available for running virtual machines and it was paramount that the students didn’t have root access to that system and also that filesystem usage was kept low. Keeping filesystem usage low is difficult as long as students build their own networks with a resulting arbitrary amount of virtual machines.

In order to control filesystem usage they utilized a special filesystem driver for User-mode Linux: hostfs. With the hostfs filesystem User-mode Linux can access the filesystem on the host computer. This works similar to NFS. Multiple UML instances can mount their (read-only) root filesystems using hostfs at the same time. This will save disk space and avoid disk image creation or duplication. The result is less physical disk io, which improves overall performance.

The hostfs driver only allows root filesystems to be mounted in read-only mode. Most Linux distributions assume the root filesystem is writable and cannot handle read-only root filesystems correctly. For this course the virtual machines were to function as routers and bridges. Linux distributions based on floppy disks contain all the

¹Information about this program can be found at http://www.os3.nl/
²A complete list of lecture subjects and practical assignments can be found at http://www.os3.nl/INR/inr.html.
In the simplified examples below the Linux kernel is started with three boot parameters.

```
% linux ip_eth0=192.0.2.1 mask_eth0=255.255.255.0 bc_eth0=192.0.2.255
```

The startup script will contain lines like these:

```
if [ -z $ip_eth0 ]; then
  ifconfig eth0 $ip_eth0 netmask $mask_eth0 broadcast $bc_eth0
fi
```

Figure 1: Using kernel parameters to configure network settings.

necessary software for routing and assume a read-only root filesystem. BusyBox is an example of these distributions which supports low resource requirements due to stripped-down and optimized binaries.

4.2 Using boot parameters to enable features

The Linux kernel can be customized at boot time. The most frequently used parameters are setting the root filesystem and the startup runlevel. Many other kernel parameters do exist, most of them are used for hardware driver configuration. Parameters unknown to the Linux kernel will be passed on to the init process as environment variables.

These boot parameters can be used to configure each UML instance by passing on special parameters that are parsed by a configuration script inside the running instance once it boots. See figure 1 for an example. The assignments done in the course assignments usually differed in what routing protocol was used and the number and configuration of network interfaces. With the ability to pass all of these parameters at boot time there was no need to modify any configuration files of each host. This feature combined with a very small read-only filesystem produced a framework for building virtual networks of functioning routers and bridges all based on one single setup saving space and machine resources. A single filesystem based on BusyBox and only consuming around 20MB of space could now be shared among all students.

4.3 Students created their own virtual networks

Students needed to write a script for each virtual network configuration. This script would handle the starting and stopping of UML and uml_switch and pass the correct parameters to the respective instances. Since this was a master level course the students were given the opportunity to use whatever scripting language or framework they wanted.

There was some difference in scripting skills among the students. The students who were comfortable with scripting did not have significant problems. For example, one student went so far as to create a XSL style sheet to create his script. Not all students were engaged at this level, several students had more problems and did not enjoy this part of the assignment. Most of the students having problems would have preferred a graphical configuration tool.

4.4 Assignments

The INR course was taught in February and March 2004. The first assignment was to explore User-mode Linux. The purpose was to practice with UML and get comfortable with configuring and running virtual networks. This was important because the students had no experience with UML and needed to have some skill development.
In the second week the students had to translate network diagrams into workable UML configurations, including IPv4 number plans. The week after that, the assignment is to add IPv6 support to the configurations created in the previous week. This includes creating IPv6 number plans.

The fourth week assignment was to examine the Spanning Tree Protocol. The students created a network with 4 bridges in a ring topology with a normal host connected to each bridge. See figure 3. The bridges where configured not to use STP. As soon as one of the workstations attempted to communicate with another machine, the network filled up with recycling packets. Students could monitor this from one of the bridges using tcpdump.

The last two weeks were used to examine routing protocols. We began by implementing both RIP and OSPF. The last assignment was to examine BGP. BGP is a exterior routing protocol, it is used to connect multiple autonomous networks together. To simulate this, an Internet Exchange was created (also with UML). Each student connects his own network to the Internet Exchange. External routing information is exchanged using BGP with the other student networks. This assignment provides both practical technical experience and organizational skills.
4.5 Student perceptions of UML

All the students seem to enjoy working with UML, although some preferred to work with real hardware. It is important to be clear about the learning goals. Like in all professions, there is a balance point between the importance of theoretical knowledge and the dependency on practical skills. Not only do students vary in technical competence, but there are also differences in attitude.

4.6 Future developments in Amsterdam

Since some skills are needed to use UML as a learning or research tool, practice and good instructions are essential. An easy to use program to design virtual networks could be helpful. But it should not be a replacement for knowing how UML works.

Two students of the System and Network Administration education, Gert Jan Verhoog and Jeroen van der Ham, have explored the possibilities to develop a tool for this[9].

5 Growing a VM administration tool

When students end up building custom scripts in order to describe each network, they often get lost in the details of the process. As a result, they are unable to reach further into higher levels of activity, because they are lost in the details.

As the effort in Amsterdam showed, using preconfigured filesystems with software-triggers can be useful. In some courses the assignments focus on the students installing and configuring the applications themselves instead of just the basic networking. In order to install higher level applications, the students need read-write access to the filesystems, more space to work in and a broader range of well-known distributions. When the systems grow in the number of configurable options like adding users or startup commands the boot parameter approach becomes long and complicated, leading to more student errors. All of these suggest the need for a way to abstract away the detail of the configuration, and to find a way to focus more directly on the higher level issues.

In addition to students needing more support for configurations, faculty and staff have issues with virtual machines. Specifically, in some scenarios, the administrative cost of using virtual machines may be just as high as using real machines. This causes course staff to acknowledge the benefits of a virtual network, but to dismiss it as too difficult or time consuming to manage. A tool that facilitates the configuration and management of a virtual network can simplify the user interface and save time for both novice users and faculty staff. It may also help keep the focus on the assignment itself, and not the configuration details of the UML instances. Additionally, UML virtual networks can be managed as logical groups using a management interface as a front-end, which can lead to pre-configured assignments being used to rapidly explore a particular problem or scenario. This allows both the students and faculty staff to focus more on the cognitive task at hand. When students get lost or confused, it is possible to recover previous configurations easily and to restart the process.

Some system administration and networking classes like to dynamically create problem scenarios randomly from templates. Using a language that describes a system, it is possible to automate deploying random scenarios individually suited to each student team with a small amount of scripting.

Some of these scenarios can grow to be complex. Complex scenarios that involve large topologies can hide both features and problems. The fact that the configuration is written down explicitly, makes it possible to do semantic checks in order to identify typing or design errors.
5.1 Versatile filesystems

As students explore applications setup and deployment, using a wider range of distributions becomes more important. The size of the Linux systems can vary based on the base distribution. Floppy-based distributions can take very little space but are limited in their feature set. Other more mainstream distributions, like RedHat are several hundred megabytes while offering more software. A typical GNU/Debian distribution can start out with a modest base system of around 100MB but will grow as more software is added. Each of these distributions may be needed depending on the purpose of the virtual network. Some network scenarios may end up using more than one distribution. For example, a network scenario might consist of a network with a router based on BusyBox and with a web server based on Debian.

Not all filesystems work directly with a UML instance. A template filesystem may need to be built for a specific distribution. As you add features and add different distributions, a VM management tool should provide you with a way to organize and manage those file system templates.

5.2 Users own different sessions

More frequently these days, students are being asked to work in teams. As a team works on a project, it would be nice to have the ability to delegate specific machines to specific teammates from the same configuration file.

Even if the user has root control over the host, this does not mean that the virtual machines should run with all privileges. Making virtual networks run as a separate user is a good security policy. With a tool that assigns user ownership to machines, it is possible to give a user exclusive access to the console of the virtual machine, without giving him root to the physical machine.

5.3 Preconfigured users

As an application installation progresses, it often is necessary for a user to be created to support that application. In order to automatically configure some scenarios, having a pre-established password entry for a user, makes it possible for that automatic script to proceed.

This leads to the ability to model more complex systems, which can be used to provide actual services to a client base. If a customer for a class wanted to see what a network configuration looked like, it would be possible to directly provide an account for the customer in the config file.

In some cases, a class might want to provide students with user level access to many virtual machines, but not provide them with any root access.

5.4 Scale the number of students supported

While a VM management tool allows configurations to become larger and more complex, it allows the details to be hidden. This allows the students to stay more focused at the problem at hand. UML has a command line infrastructure, which allows for a low overhead way to startup and manage services. This efficiency combined with the detail hiding should make it possible to scale assignments to support larger classes.

5.5 Management of host clusters

By grouping machines together into projects, it is possible to allow students and faculty to think about problems as a project, instead of at the machine level. A higher level definition of a virtual network helps keep projects apart. Users may issue management
commands to the network as a whole and individual VMs. Giving them the flexibility
to work at a higher level of abstraction.

5.6 Higher level of abstraction

This higher level of abstraction through a configuration language provides many ben-
efits. It is possible to describe a large and complex network quickly. It is possible to
visualize that network. It is possible to coordinate resources between different team
members and to facilitate the documentation of a project.

In addition, professionals can use a configuration language to simulate a problem
space before applying a solution on a production network.

5.7 Rapid deployment

Using a pre-written configuration file, a faculty member can quickly deploy a large and
complex network. Dozens of nodes can be created and deployed in a few minutes.

5.8 Versatile and expandable

Using an administration tool may vary according to the needs of the particular project.
It is an important aspect that the configuration can be modified if necessary and doc-
umented clearly. The configuration language has to be versatile enough to create both
very simple networks but also complex ones. Both the topology of the network and
the complexity of the virtual machines need to have this versatility.

6 My Linux Network

MLN (My Linux Network)[2] is a virtual machine administration tool developed at the
University College of Oslo as part of our effort to improve virtual machine adminis-
tration. Its goal is to provide a management interface to all the aspects of creating and
running virtual machines based on User-mode Linux. Through MLN one is able to
keep track of different virtual networks and start and stop them independently. Every
network is described in the MLN configuration language, which encompasses both
network hardware and host machines.

Here is an example:

    global {
      project simple-network
      $console = xterm
    }

    switch lan {
    }

    superclass hosts {
      term $console
      template Debian-3.0r0.ext2
      size 85M

      network eth0 {
        switch lan
        netmask 255.255.255.0
        broadcast 10.0.0.255
      }
    }

    host starfish {
      superclass hosts

      network eth0 {
        address 10.0.0.1
      }
    }

    host catfish {
superclass hosts
      network eth0 {
          address 10.0.0.2
       }
    }

host blowfish {
      superclass hosts
      network eth0 {
          address 10.0.0.3
       }
    }

This example shows simple network consisting of three hosts connected together by a switch. See Fig 4 for a graphical representation of the same network.

Figure 4: A visualisation of the network described in the MLN language in Section 6.

The network shown above consists of three hosts connected to a single switch. All three hosts inherit settings from a common superclass called hosts.

6.1 MLN overview
MLN reads configuration files and builds the hosts filesystems based on templates. Start and stop scripts are also generated. MLN has commands for building, starting, stopping, upgrading and monitoring virtual networks. Hosts can be rebooted individually and their console can be accessed from the physical machine, meaning that virtual machines without properly functioning network interface can be accessed nonetheless by students. This removes the need for supervised access to a machines console.

MLN also supports upgrading the configuration of a running virtual network, like adding more machines, where only the affected machines are rebuilt. This is practical for networks that are already in use in class.

The MLN configuration language supports variables and inheritance. Complex network topologies can be described in this language and the machines can be configured with respect to kernel modules, network interfaces, startup commands, users, filesystem size and available memory just to mention a few. One does not have to use all of these settings.

6.2 Benefits
With MLN virtual networks can be created easily and used in scenarios in the field of configuration management, intrusion detection and packet filtering. Administration of the networks is possible through a simplified interface that can start and stop the entire network. Using the configuration specification files, scenarios can be taken down and reproduced quickly for the next class or student group.
From the perspective of a student, there are also some benefits. MLN also supports building and running systems as a different user than root, which allows students to create their own test networks. MLN adds security and stability by making sure a virtual machine is not started twice or rebuilt on top of an already running system.

MLN works as a command-line tool on Linux systems. Section 13 will briefly describe other projects that will utilize MLN as a backend. MLN and its documentation is freely available at http://mln.sourceforge.net.

An example of a tool which can process a configuration file and provide an overview of the design is mln2dot. Mln2dot will convert an mln configuration file into a dot diagram file, making it easy to visualize the topology of a configuration.

7 MLN used at Willamette University

In teaching an introductory networking class at Willamette University, the class needed to work in teams to create a specific network configuration. In the process of defining the assignment, it was possible for the lecturer to build an MLN simulation for the class assignment. This assignment was able to be demonstrated directly, showing the specific tasks needed and to allow for some exploration of the system in class. See Fig 5 for an example.

This enabled a more detailed discussion of how things needed to be formulated. This discussion also lead to a more rapid deployment of the assignment by the students.
8  MLN used at AO Computer Systems

MLN can be used in industrial scenarios as well as in an educational environment. In the process of doing contract jobs, staff at AO Computer Systems has done the initial install of system configurations in an MLN configuration as a test situation. This allows for experimentation of many situations rapidly and cheaply on a local machine, before production machines are altered. On several occasions, a possible system configuration was tested and simulated using MLN, which illustrated a specific problem, which was resolved before the configuration was put into production use on the real systems.

MLN has turned into a useful tool for several projects, enabling a more rapid development of a set of machines and reducing system outages and problems. More than 50% of system failures and outages are caused by system administration processes and errors[10]. By creating a simulation and testing that simulated environment ahead of time, many of these errors can be eliminated.

MLN is also being used at part of a project to deploy a grid using virtual servers called GridVern. This project aims to create a market for services based on dynamically allocated virtual servers.

9  MLN used at Oslo University College

The University College of Oslo will teach the course "Intrusion detection and firewall security" in the fall 2004 as part of the Master Degree program in System Administration. Last time this course used a lab consisting of physical machines. This time, it will utilize virtual machines and have prepared scenarios for the students. MLN was used to generate these scenarios of which some counted over 40 virtual hosts and 24 switches. Smaller scenarios will be used, when performance is of greater importance. MLN allows to effectively make changes and deploy new test cases in only few minutes. A single server with two AMD Opteron CPUs and 2 GB of RAM will carry the entire virtual lab. The benchmarks of our test networks give encouraging results.

Senior students have used MLN in projects where they needed a distributed environment to test their problems. One example was the development of a benchmark suite for several machines and the analysis of the data. In this case MLN was used to quickly build a simple network with ssh access and helped keep the focus on data analysis and statistics, which was the theme for the assignment.

The room which hosted the physical lab during last years course is now being reallocated and has been turned into a meeting room for our master students.

10  Discussion

Our collective experience is that several approaches can be used depending on factors like available hardware, number of users and their skill-level. As for the approach taken at The University of Amsterdam, they focused on keeping resource usage and privileges to a minimum. Letting students design and run virtual networks removes the control over how many virtual machines may run at a point in time. By making the UML instances read from the same filesystems, disk resources could still be controlled.

How the students design their networks plays a role too. Skilled students may have no need for cognitive abstraction levels and may also combine the task of designing virtual networks with that of learning a scripting language, as they did at The University of Amsterdam.
The benefits of this approach is that a highly specialized filesystem can be crafted which only uses a small number of resources. However, it is worth considering whether faculty and staff have enough time to specialize a filesystem for the course or assignment. In some cases more than one filesystem might be needed.

One solution to the need for hand crafted filesystems and network scenarios is to engage senior students in the task of setting up and deploying specific filesystems or network scenarios as part of senior projects work. Unexperienced users may need a more controlled environment and a predefined configuration language for their networks. Faculty and staff can also benefit from a protected environment. When the main focus is on the tasks that run on the virtual network rather than how it is built.

One aspect of our approach, is that we use only GNU/Linux systems. Linux provides a broad set of tools that allow students to explore many different aspects of networks and server configurations. While we miss the opportunity to use Windows systems, the underlaying issues we want to explore are not as broadly represented in the Windows environment. Linux supports projects like the Linux Router project, Cisco Router Emulators, Bridge configurations, as well as being important in the server market. Projects like the BusyBox distribution are aimed at small system footprints.

UML is primarily a command line oriented environment, without a graphical interface for students to interact with the system like in [6]. Most system administration tools for Linux are developed for the console so there is no need for a graphical environment for either running or configuring virtual networks. A GUI requires significant resources. An X-based VM instance requires additional disk space and extra memory. In a student lab environment, where resources are at a premium, a smaller VM image provides advantages. The more resource friendly the virtual machines become, the more students can be supported. Working on the console will pose a challenge for some students. There is a tendency for current students to be comfortable working with a GUI interface, which leads some students to difficulty when they end up working in a command line environment. However, working in a command line environment is a good skill to have for system admin students and network administrators. Building good UNIX/shell skills is a part of this approach.

Through this exploration of a virtual environment to provide hands on services for networking students, the result show that more resources can be given to the students with virtual machines. These virtual machine based resources lead to thoughts about wider course offerings. With a virtual machine based lab, it would possible to create a web-based course around networking. It has been shown that courses like Introduction to Calculus can be taught satisfactorily with a web-based mechanism[11]. It would be a logical next step to try to do the same thing with computer networking and system administration courses.

11 Related Work

Simulation software is widely used in education as a way to gain direct experience. Schools on all levels have tried out virtual environments to improve learning. Three dimensional environments are often deployed to create virtual worlds or laboratories for classes. [5] Using network simulations in the same way makes sense as a way to increase the direct experience with network issues for system administration students.

Other projects have looked at the process of using Virtual Machines to simulate networks. VELNET is a project based on virtual machines running on VMware [6]. In another project [7], Stockman describes the benefits of using virtual machine software,
like VMware and Virtual PC, together with remote desktop tools in order to make them accessible in class and networking labs.

There are also virtual machine administration tools based on User-mode Linux, like VNUMUL, UMLAZI and umvs.

Another philosophy different from building virtual environments, is to make sure the existing hardware is utilized to a maximum, thus keeping the costs down. In this article [8], Miller et. al describe how they used removable harddisks in all their classes. Students checked out their personal harddisks in the beginning of each class from a person acting as a lab assistant. This is a good solution as long as the students only need one machine each, and they don’t need them up and running all of the time.

12 Conclusion

Virtual machines open doors in system and network administration classes. Assignments using more advanced technologies can now be given without the need of more hardware.

With the help of VM administration tools, we get virtual networks that are:

- Replicatable
- Duplicatable
- A configuration method that incorporates all aspects of the systems (topology, network "hardware", host micromanagement)
- Fast and easy to deploy
- Free
- Uses less hardware resources

Time is saved in the case where the course staff needs to manage all the virtual machines, or where the students lack the necessary skills. This relates to common system administration topics too, namely automatic customized installs and the configuration management problem. That is why MLN might be of interest for sysadmins too. The programming language used in MLN illustrates a centralized configuration management approach that would fit nicely into directory based database models, like LDAP. Work is in progress on this task.

Getting the students to write their own scripts and build their own virtual networks by hand can prove an learning experience for the students. This approach, however assumes some skill and is mostly applicable to senior students. A broader range of students can use UML based systems when there is a configuration tool. And with this tool a broader range of assignments and tasks can be accomplished. This allows the students to have a better learning experience and to delve deeper into course material.

Specifically, by using MLN, courses can achieve focused assignments in a short amount of time, with a small number of resources. This creates opportunities for:

- Larger problems
- Deeper problem solving
- Immediately testable configurations

These same features make MLN an interesting tool for professional system administrators. Simulating solutions and configurations before they are deployed can reduce production outages.
Future work

MLN is being used as a part of several development projects. As a result, it is still undergoing change and evolution.

One project aims at creating a LDAP database representation for the data model used in MLN. This database can prove helpful in creating graphical user interfaces to MLN or expert systems that might change the properties of a running virtual network. This also provides a mechanism for reducing the distance from customers to system provisioning.

The GridVern project is looking at how to create networks that are able to manage themselves, even with topology and hardware, through a market based mechanism. This project makes heavy use of virtual machines instead of real hardware and uses MLN as an backend.

Other projects are exploring how system configuration can be described in a higher level language. One possible target for these projects is to create MLN configuration files.

As the number of projects grows, the demand for new features expands. And the number of system templates and supported distributions increases accordingly.

References