# Saving and Recovering Systems

Philipp Tekeser-Glasz, Sebastian Gallenmüller*, Manuel Simon*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: philipp.tekeser-glasz@tum.de, gallenmu@net.in.tum.de, simonm@net.in.tum.de

***Abstract*—The Chair of Network Architectures and Services operates multiple testbeds that allow researchers to develop and run reproducible network experiments. Reproducibility is achieved by running experiments on test nodes that have no persistent storage. Instead, test nodes boot a live system via PXE. This ensures that an experiment always starts with the same initial state. Users can reserve test nodes in advance using a calendar web interface. The facts that test nodes cannot store data persistently and that they are shared between users create a problem for users who are developing new experiments since all development progress is lost after a restart of the test node.**

**This paper presents an approach to overcome this problem by using virtual machines that can be saved to a host with persistent storage and later restored on a test node.**

***Index Terms*—virtualization, network experiments, testbeds, reproducibility, development**

## 1. Introduction

The testbeds at the Chair of Network Architectures and Services are managed by the Plain Orchestration Services (pos) which allows researchers to develop and run reproducible network experiments [1]. One of the goals of the testbeds and the pos framework is reproducibility, which means that different researchers are able to obtain the same result using the same experiment setup [2]. Each testbed consists of multiple test nodes that run the experiment code and a management node that controls the execution of the experiment. As a measure to achieve reproducibility, test nodes run live systems and do not have any persistent storage that could contain leftover data from previous users. This ensures a clean state at the beginning of an experiment.

However, during the development phase of an experiment, this architecture prevents users from saving the current state of a project between sessions since all data is lost after a reboot. This paper presents an approach to develop an experiment in virtual machines on a test node, which can be saved to the management node and restored later.

Section 2 gives a short introduction to architecture and the technologies used in the testbed. The problem of lost development progress that arises from the stateless architecture is described in Section 3 and a possible solution is explained. In Section 4, the implementation of that solution is presented. Section 5 shows an example workflow from the user's perspective. The limitations of the chosen approach are outlined in Section 6.

## 2. Background

This section explains the architecture of the testbed. The testbed consists of a management node and multiple test nodes. Test nodes run the experiment code. Some of them contain specialized network hardware for specific experiments. In order to guarantee a defined state at the beginning of an experiment, these nodes do not have any persistent storage that could contain leftover data from previous experiments. A live operating system is loaded over the network using the Preboot Execution Environment (PXE). Therefore, any changes to the system are lost after a reboot. In addition to the test nodes, each testbed has a management node to control the test nodes. On the management node, each user has a persistent home directory. Each test node contains a Baseboard Management Controller (BMC); a device that allows the management node to send commands to the test node even if the operating system is unresponsive or has not been started. The management node communicates with the BMCs over the network using the Intelligent Platform Management Interface (IPMI).

In order to ensure that users do not interfere with each other by accessing a node at the same time, pos offers a web interface through which nodes can be reserved in advance. Users can then log in to the management node using SSH and then use the pos command line tool to set up their test nodes.

Only the management node is accessible from the Internet. If a user wants to connect to a test node, they first have to connect to the management node via SSH and can then establish an SSH connection to the test node from there.

For development purposes, pos also offers the option to create multiple virtual machines on a physical test node. This is done by setting up the physical test node and launching virtual machines using the management software libvirt [3]. Virtual machines are managed and booted using the same protocols that physical test nodes use. In order to allow pos to control virtual machines, the host runs VirtualBMC [4] which accepts IPMI commands and passes them on to the virtual machines running under libvirt. The virtual machines receive their live operating systems via PXE and do not have any persistent storage either.

## 3. Problem

While the fact that test nodes do not store data persistently allows researchers to run automated and repro-

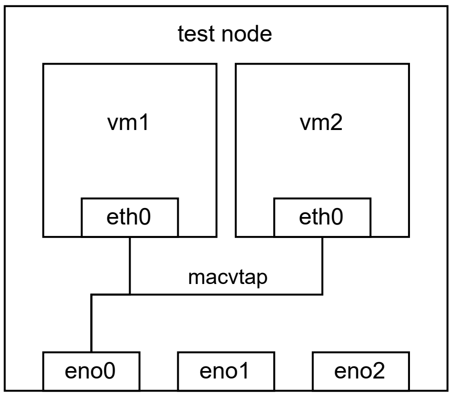

Figure 1: Networking on a test node

ducible network experiments, it is difficult to develop experiments in this environment since all files are lost after a reboot and users have to manually save and restore their progress between sessions.

This paper presents an approach to automatically save the state of virtual machines to the management node from where it can later be restored on another test node.

The chosen approach consists of creating memory images of the virtual machines on a test node, copying them to the management node where each user has a persistent home directory and later restoring the virtual machines.

Additionally, it is necessary to save the network configuration of the virtual machines. In a typical setup, there is a virtual management network that connects all virtual machines to the management interface of the host using a macvtap bridge device, as shown in Figure 1. The management interface is the one through which the node can communicate with the management node. This network allows pos to manage the virtual machines and users to connect to them via SSH from the management node. Other physical interfaces that are connected to other nodes might be present for experiments. For this reason, the correct interface has to be selected when setting up the virtual machines for the first time and after restoring since the new host could have other network interfaces than the old one. This selection is done automatically by searching the interface through which the default gateway is reachable.

To ensure that pos can find and control the virtual machines, the virtual network interfaces have to use MAC addresses that are derived from the host's IP address. When restoring the machines on a different test node, the MAC addresses have to be changed to match the new host's IP. This is done by removing the interface from the virtual machines before saving and adding a new interface with the updated MAC address after restoring. After that, the guest system has to be reconfigured to recognize the new interface with a different MAC address. Since there is no working network connection at this stage, the reconfiguration is performed using a virtual serial port. This reconfiguration of the management network allows restoring the virtual machines on any test node.

A simpler approach is to leave the network interfaces untouched, but this comes with the limitation that the virtual machines can only be restored on the test node from which they were saved because the MAC addresses will otherwise not match the pattern that is expected by pos and it will not be possible to establish a connection to the machine. The advantage of being able to freely choose a host is that users can work at any time without having to wait until a specific test node becomes available.

It is important to note that the chosen approach is not meant to replace the reproducible architecture of the testbed. While the approach simplifies the development process of new experiments, actual measurement results should only be acquired using the reproducible approach described in [1].

## 4. Implementation

The development workflow consists of three steps that will be explained in this section. For each step in the workflow, there is a Bash script that the user executes on the management node. The scripts connect to the specified test node to perform the necessary tasks. A simplified overview of the interactions between the nodes in each step is shown as a sequence diagram in Figure 2. Each of the following subsections corresponds to a frame in Figure 2.

### 4.1. Creating virtual machines

This process is based on the existing example code that is used to create virtual machines in the testbed [5]. In this step, the user starts a Bash script on the management node which starts the selected test node and installs the necessary virtualization software. After that, the virtual management network is created. Then a user-specified number of virtual machines is created. Each virtual machine has a network interface that is connected to the management network. The MAC address is calculated based on the IP address of the host and the ID of the virtual machine. For each virtual machine, a virtualBMC is started on the host that listens on a specific port based on the ID of the VM. Once this is done, pos can start the VMs, which then load their live operating system via PXE like physical machines. When the boot process is completed, getty is started on the virtual serial port in order to allow a reconfiguration of the network settings after restoring. Users can now establish SSH connections and start working on their projects.

### 4.2. Saving virtual machines

To save the virtual machines, a Bash script is executed on the management host which launches another Bash script on the test node that performs multiple steps. The first step is to create a file that contains a list of all virtual machines and their IDs. The second step is to remove the management interfaces from the virtual machines. If the user has configured other interfaces, the script will ignore those. In the third step, the definitions of the virtual machines are stored as XML files. These files contain general configuration data, e.g. memory size and serial ports. After that, the actual memory dump is created and compressed using gzip. The last step is to export the network configuration from libvirt and replace the host

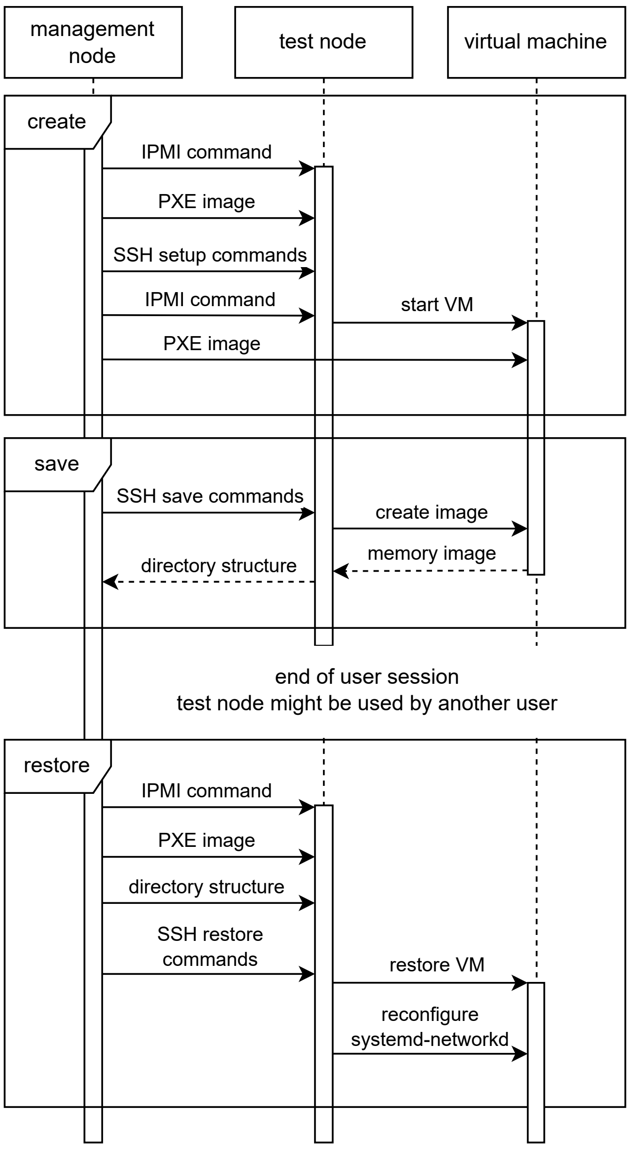

Figure 2: Simplified sequence diagram of the implementation

```
vm_save
|-- net
|   `-- net.xml
|-- vm
|   |-- vm1.gz
|   `-- vm2.gz
|-- vmmacs
`-- vmxml
    |-- vm1.xml
    `-- vm2.xml
```

Figure 3: Directory tree after saving

specific interface name with a placeholder variable that will be replaced on the new host when restoring. The final directory tree on the test node after all steps are completed is shown in Figure 3. This directory is then copied to the user's home directory on the management node using SCP. The compressed memory image of a Debian 11 system with no additional software installed has a size of 1 GB.

### 4.3. Restoring virtual machines

The restoration process begins like the creation process by starting the test node and installing the virtualization software. In addition to that, the saved directory is copied from the management node via SCP. First, the network configuration is read and the placeholder variable is replaced with the management interface. Then, the virtual machines are redefined using the XML files. After that, the memory images are decompressed and restored. Each virtual machine gets a new management interface with a MAC address based on the host's IP address and the ID that is read from the `vmmacs` file. However, the guest system does not recognize the new interface because of the changed MAC address. For this reason, an expect [6] script is launched to replace the MAC address in the interface configuration file of systemd-networkd on the guest. After this step, the virtual machines can be reached from other hosts on the network. In order to allow pos to send IPMI commands to the virtual machines, virtualBMC is started on the host system. This allows the user to reset the virtual machine to a clean state using the pos command line tool.

## 5. Example Workflow

This section shows the development workflow using the scripts. As an example, iperf3 will be installed on two virtual machines on test node `vmexp1`. These virtual machines will then be saved to the management node and restored on test node `vmexp0`. All shown commands are run on the management node.

First, two virtual machines with Debian 11 are created on the test node `vmexp1`.

```
./experiment.sh vmexp1
```

After the script has finished executing, there are two virtual machines called `vmexp1-vm1` and `vmexp1-vm2`.

It is now possible to establish an SSH connection and start developing. In this simple example, we will install iperf3 on both virtual machines.

```
ssh vmexp1-vm1 apt install iperf3
ssh vmexp1-vm2 apt install iperf3
```

Then, iperf3 is run in server mode on `vmexp1-vm1` and in client mode on `vmexp1-vm2`.

```
ssh vmexp1-vm1 iperf3 -s -D
ssh vmexp1-vm2 iperf3 -c vmexp1-vm1
```

This command will measure the throughput of the virtual network connection between the two virtual machines.

At the end of a session, the user can save all virtual machines to a new directory called `save_dir` using the following command.

```
./save_vms.sh vmexp1 save_dir
```

The directory `save_dir` will then contain the structure shown in Figure 3.

In order to start a new session on the node `vmexp0`, the following command is used.

```
./restore.sh vmexp0 save_dir
```

Now, the user can continue working on the virtual machines called `vmexp0-vm1` and `vmexp0-vm2`. In this case we can run iperf again in order to see that the installed package is present and that the network connection has been restored correctly. Note that the iperf server process is still running on `vmexp0-vm1` since we saved a full memory image of the virtual machines.

```
ssh vmexp0-vm2 iperf3 -c vmexp0-vm1
```

Like before, this command will show the throughput of the virtual network connection between the two restored virtual machines.

## 6. Evaluation

The chosen approach simplifies the development process of new experiments by allowing users to create backups of virtual machines and restore them later on a different host. It makes a more effective use of testbed resources possible since users only have to use one physical test node when developing experiments that would normally require multiple nodes.

However, there are some limitations. First, it is only possible to restore virtual machines on a different host when there is only a management network that has the structure shown in Figure 1. Other interfaces are ignored by the script and might therefore lead to an error when restoring on a new host where the interface is not available. More complex network setups, e.g. passthrough interfaces, are possible but restoring will only work on the same host where these interfaces are available.

When restoring network interfaces of type virtio on the new host, AppArmor caused an error. For this reason, AppArmor had to be completely disabled via a kernel parameter in order to be able to restore these network interfaces.

Another limitation is that the network reconfiguration on the guest system is currently only supported for systemd-networkd. It is therefore only possible to use specific operating systems in virtual machines. In this case, only Debian 11 was tested, which is commonly used on the testbed.

When an experiment is ready to run, it might be necessary to make changes to the code in order to run it on physical hardware.

## 7. Conclusion

We explained the stateless nature of nodes in the testbed and the advantages it has for reproducible network experiments. This created the problem of losing progress during the development phase of an experiment. We proposed a solution to this problem based on virtual machines which can be saved to the management node and restored on a different test node. An implementation of this approach using Bash scripts was presented. We also provided a demonstration of the implementation in which we showed the steps to create, save and restore a session.

In the future, this solution could be integrated into the pos command line interface in order to simplify the development workflow. Another topic for future work could include the development of a more complex example project using the presented approach.

## References

[1] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, “The pos framework: A methodology and toolchain for reproducible network experiments,” in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 259–266. [Online]. Available: https://doi.org/10.1145/3485983.3494841

[2] ACM, “Artifact Review and Badging - Current,” https://www.acm.org/publications/policies/artifact-review-and-badging-current, 2020, [Online; accessed 18-June-2023].

[3] “libvirt: The virtualization API,” [Online; accessed 18-June-2023]. [Online]. Available: https://libvirt.org/

[4] “How to use VirtualBMC — virtualbmc 3.0.2.dev5 documentation,” [Online; accessed 18-June-2023]. [Online]. Available: https://docs.openstack.org/virtualbmc/latest/user/index.html

[5] S. Gallenmüller, “vm-example,” https://gitlab.lrz.de/I8-testbeds/pos-examples/-/tree/master/tutorials/vm-example/bullseye, 2021, [Online; accessed 18-June-2023].

[6] D. Libes, “expect: Scripts for controlling interactive processes.” *Computing Systems*, vol. 4, pp. 99–125, 03 1991.