Demo: A Virtualized Lab Testbed with Physical Network Outlets for Hands-on Computer Networking Education

Mark Schmidt   Florian Heimgaertner   Michael Menth
{mark-thomas.schmidt,florian.heimgaertner,menth}@uni-tuebingen.de
University of Tuebingen, Dept. of Computer Science, Tuebingen, Germany

ABSTRACT

This demo presents a testbed for computer networking education. It leverages hardware virtualization to accommodate 6 PCs and 2 routers on a single testbed host to reduce costs, energy consumption, space requirements, and heat emission. The testbed excels by providing dedicated physical Ethernet and USB interfaces for virtual machines so that students can interconnect them with cables and switches like in a non-virtualized testbed.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education; C.2.3 [**Computer-Communication Networks**]: Network Operations

Keywords

Computer networking education; virtual machines; VLAN

1. INTRODUCTION

Hands-on labs are essential for computer networking education. At the University of Tuebingen, two hands-on networking courses are offered. An undergraduate course based on [3] comprises networking basics, routing, DHCP, DNS, Transport Layer, and Email. The graduate course includes networking and security technologies like TLS, Firewalls, Virtual Private Networks, WiFi Security, and Voice over IP.

An existing lab setup featured 3 physical testbeds, each consisting of 6 PCs and 2 Cisco 2514 routers. Aging and degradation of hardware, along with limited lab capacity, excessive power consumption, and emission of heat and noise led to a re-design of the testbed setup. The goal was to provide testbeds for more participants and to overcome the issues of the old setup. Another objective was to keep hardware costs low and reuse the new equipment as workstations if not needed for lab exercises. These requirements called for a virtualization solution. Existing approaches include replacing physical computers and routers by virtual machines (VMs) on a testbed host, or even multiple testbeds on a single server. However, physical hands-on experience, i.e., connecting the VMs with cables and switches, should be preserved as well as interoperability with physical devices.

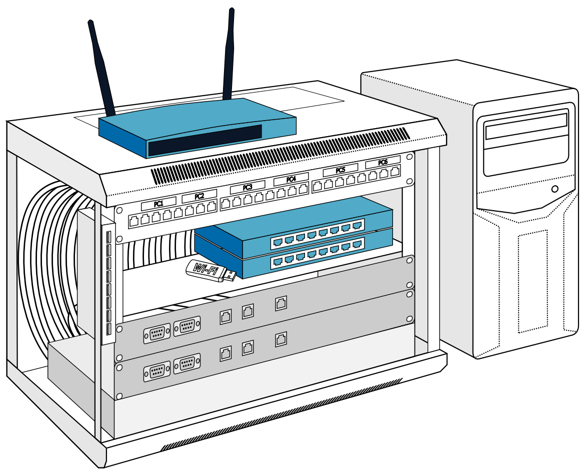
Figure 1: Networking cabinet and testbed host.

Our solution consists of PCs and routers running as VMs on a testbed host. Physically accessible ports for Ethernet devices of the virtualized PCs and routers are provided on an extra panel so that the same hands-on experience is possible as with an entirely physical testbed. The passthrough of interfaces from the VMs to the physical outlets was achieved by unconventional use of advanced networking and virtualization concepts on commodity hardware.

2. TESTBED ARCHITECTURE

The nodes (6 Linux PCs and 2 routers) of the testbeds are realized as VMs. The physical computer (testbed host) is running the virtualization platform built by the Linux Kernel based Virtual Machine (KVM) [1], libvirt and QEMU.

To enable efficient virtualization and device passthrough, hardware support is required. Intel-VT [5] is used to enable hardware accelerated virtualization of x86 systems. Single-Root I/O Virtualization (SR-IOV) [4] supports advanced device virtualization by providing multiple virtual functions (VF) per physical function (PF). A PF is a full-featured PCI device whereas VFs are lightweight PCI devices that are managed by a PF.

Figure 1 shows a lab testbed. The networking hardware is placed in a standard 19-inch wall mount cabinet. At the bottom a managed switch is mounted facing backwards. Above the switch, two front panels with DB9 and RJ45 outlets represent the virtual routers. Unmanaged desktop switches and other networking devices for use in the lab exercises are placed on a rack-mount shelf. The topmost rack unit contains a labeled patch panel providing Ethernet ports for the virtualized PCs.

For experiments involving IEEE 802.11 technology, a wireless access point is available as a physical device. WiFi adapters for the PCs are provided as USB devices that can be passed through to the VMs.

To enable the students to physically interact with the VMs, e.g., interconnect them with real cables and switches, we provide physical access to the virtual interfaces. We achieve this objective for Ethernet interfaces by using VLAN technology [2] and a VLAN-capable managed switch. IEEE 802.1Q supports insertion of *tags* into the header of Ethernet frames which enables multiplexing several VLANs over a single physical link or port.

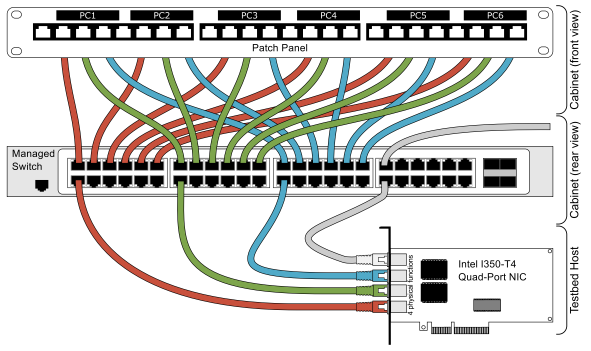

Figure 2: A managed switch connects the quad-port NIC of the testbed host to the patch panel.

Figure 2 shows an SR-IOV capable quad-port NIC, a managed switch, a patch panel, and twisted pair cables to connect them. Each of the four ports is implemented as a separate PF providing 7 VFs. Interfaces of the router VMs are omitted in the figure. The VF devices are separately passed through to the VMs and serve as their Ethernet interfaces. We use dedicated VLANs with unique IDs for each VF. The PFs act as an Ethernet bridge and forward data for the VFs as tagged VLANs. Those VLANs are then demultiplexed by the managed switch, stripped of their tags, and diverted via separate switch ports to the patch panel.

The Intel I350 network adapter used in our setup provides a feature called *PF loopback*. It allows to bypass an external switch when source and destination are VFs of the same PF. While PF loopback can improve performance for data center applications, it breaks our setup because the patch panel and the cables placed by the students would be bypassed as well. This issue was solved by setting the internal bridge of the PF to Virtual Ethernet Port Aggregator (VEPA) mode.

In addition to Ethernet interfaces, we make physical USB devices like WiFi adapters available in the VMs. They are passed through to the VMs, which works slightly different than the passthrough mechanism for PCI devices. USB devices are identified by their address on the bus and a unique identifier. The passthrough is realized by mapping the entire address into the VM. This can be achieved by libvirt with the help of special udev rules.

To retain the look and feel of real routers that provide a Cisco IOS like user interface, we use the routing software suite Quagga. Quagga provides a CLI called `vtysh` that behaves similar to the text-based interface of IOS. To not only keep the user interface impression, but also the physical one, we designed a special front panel for the cabinet that provides the typical look of a Cisco router. The serial link between different routers is implemented by a PPP connection, a null modem RS-232 serial cable, and USB/Serial converters. However, PPP by default provides flow control for its connections which has a bad effect: packets are not dropped if the physical link is broken but are queued until the link becomes available again and then sent at once. This results in the measured round-trip-time (RTT) being dominated by the downtime of the link and not being the real delay. Therefore, we explicitly disable flow control to get the desired behavior.

3. DEMONSTRATION

Our demonstration shows parts of a lab experiment adapted from [3] for our undergraduate course. A video[1] giving a short impression is available at our webserver.

First, we provide a glimpse of the lab and then focus on one testbed where we present the graphical user interface to interact with the VMs. After starting and connecting to the VMs, we show the topology for a dynamic routing experiment created by plugging cables. Integrating an additional serial link between the two routers triggers changes in the routing tables, pointing out the shortcomings of the hop-count metric for routing. This can be seen by an increased RTT for ICMP messages between two VMs, while the packets take a shorter path (3 instead of 4 hops) using the slower serial link.

Afterward we demonstrate how to switch the testbed from testbed mode to workstation mode running a regular desktop Linux installation. This way, the testbed resources can be used for programming and simulation courses while no hands-on networking course is running.

4. ACKNOWLEDGEMENTS

The authors would like to thank W. Braun, M. Hoefling, A. Stockmayer, and S. Veith for support in assembling the testbeds and preparing the demo.

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG) under grant ME2727/1-1. The authors alone are responsible for the content of the paper.

5. REFERENCES

[1] A. Kivity et al. kvm: the Linux virtual machine monitor. In *Linux Symposium*, 2007.

[2] LAN/MAN Standards Committee of the IEEE Computer Society. *IEEE 802.1Q: Virtual Bridged Local Area Networks*, 2003.

[3] J. Liebeherr and M. E. Zarki. *Mastering networks – an internet lab manual.* Pearson Education, 2003.

[4] PCI SIG. Single Root I/O Virtualization and Sharing Specification 1.1, 2010.

[5] R. Uhlig et al. Intel Virtualization Technology. *IEEE Computer*, 38(5):48–56, 2005.

[1]`http://kn.inf.uni-tuebingen.de/demos/sigcomm14`