Introduction to Virtualization

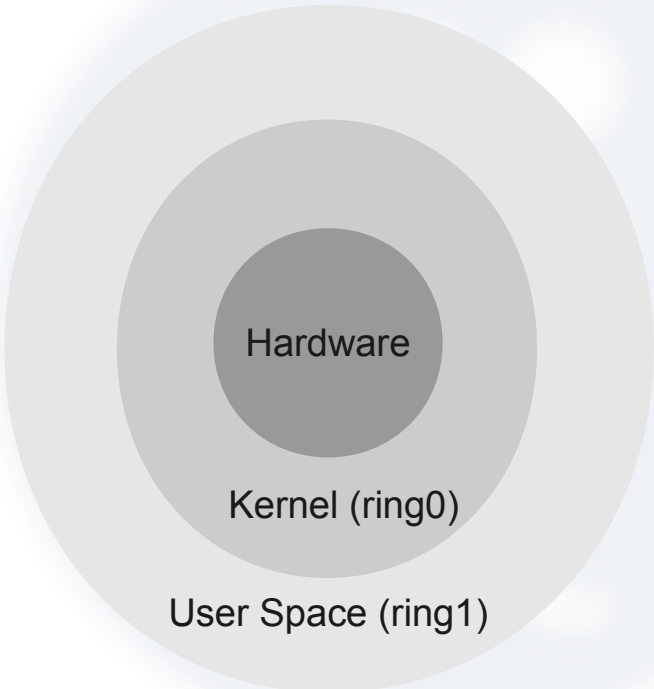
Jose L. Muñoz, Oscar Esparza, Juanjo Alins, Jorge Mata
Telematics Engineering
Universitat Politècnica de Catalunya (UPC)

Outline

OS Rings

- **Kernel**. Kernel manages hardware and essential operations with the hardware.
- **User space**. Processes of users.

- Interfaces:
  - Hardware to Kernel: **privileged operations** (low level operations). E.g. manage CPU -> CPU scheduler.
  - kernel to user space applications: **system calls**. E.g. ask for creating a new process.



Decoupling Hw/OS

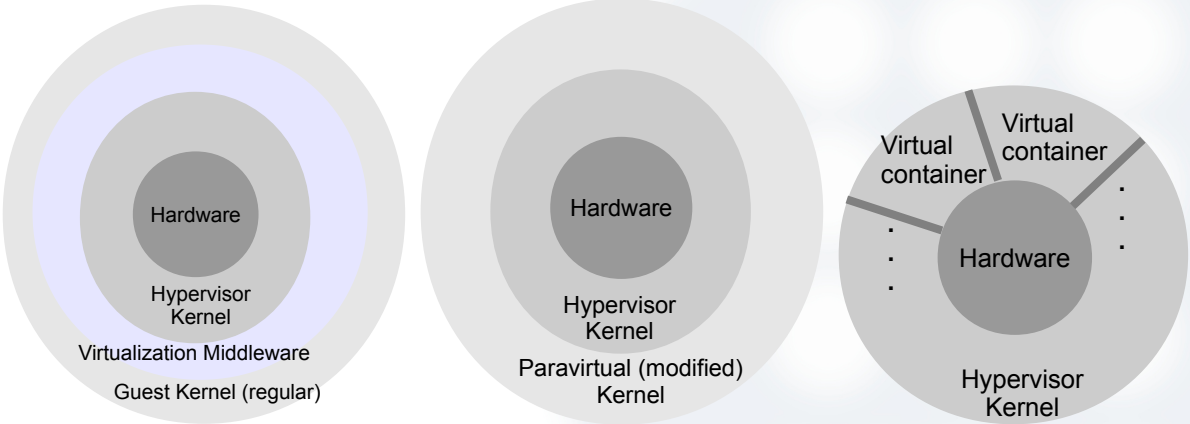
- Virtualization **"decouples"** the previous existing tight relationship between the hardware and the OS:
  - Breaks de paradigm **"one physical machine one OS"**.
  - Virtualization allows to run multiple virtualized operating systems (OS) over a common hardware.
  - Now we can have OS that do not directly run over hardware.
- Server virtualization has been a **game-changing technology for IT**:
  - Is the key innovation technology that enables **cloud computing**.

Hypervisor & Guests

- **Hypervisor or Physical Host (phyhost).** This is the hardware, the operating system and any other software needed to run the virtual OS (guests).
- **Guests**. These are the virtual OS running over the phyhost.
  - A guest might be a traditional OS running just like if it was on a real host.
  - To do so, the host emulates all the system calls for hardware.
  - This makes the guests feel like if they were in a real computer.

OS Virtualization Types

- We have three main technologies for virtualizing OS: hardware emulation, paravirtualization and virtual containers.



Hardware Emulation

- **Hardware Emulation or Virtual machines (VMs)**
  - Allows an hypervisor (**phyhost**) to run an arbitrary guest operating system.
  - The guest OS is not modified and it is not aware that it is not running over real hardware.
  - The main issue with this approach is that some OS instructions require to be in supervisor mode and this causes problems since the guest OS is being executed in the user space of the hypervisor.
  - As a result, we need a virtual machine monitor (VMM) in the hypervisor to analyze executed code and to make it safe on-the-fly.
  - This VMM is part of the “virtualization middleware”.

Paravirtualization

- **Paravirtualization**
  - Most of the work of the VMM is implemented in the guest OS code, which is modified to avoid the use of privileged instructions.
  - The paravirtualization technique also enables running different OSs on a single server, but requires them to be ported, i.e. guest kernels must “know” that they are running in a user space of an hypervisor.

Containers

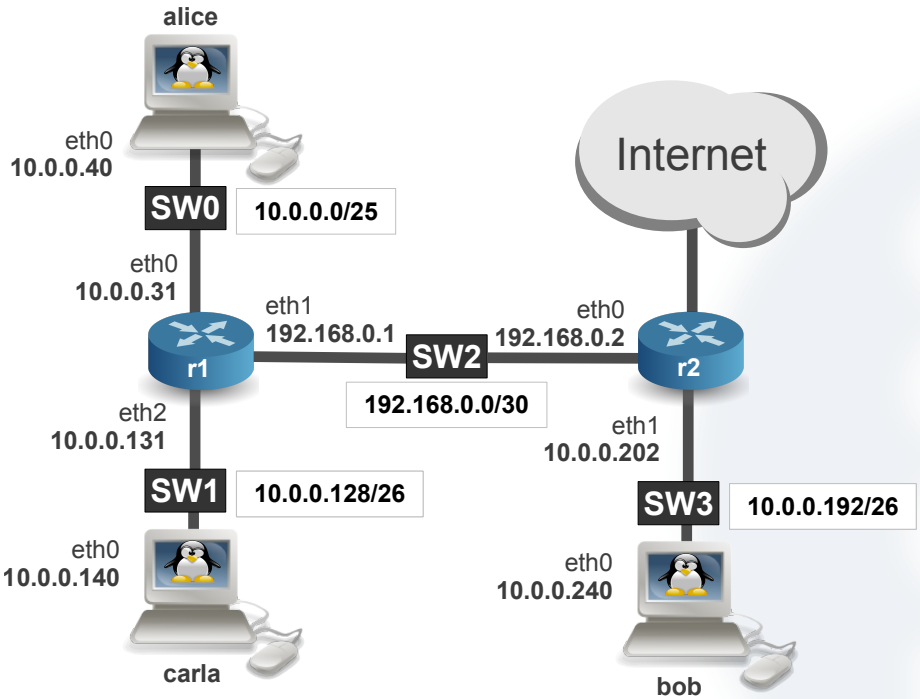
- **Virtualization on the OS level, a.k.a. containers virtualization**
  - This technique shares a kernel for several virtual OS called “containers”, where each container has an isolated and secure environment.
  - With containers you can even run different distributions but the kernel is shared among containers.

Comparison

- Hardware emulation has a wider scope of usage (many OS), but the poorest performance.
- Paravirtualization has better performance than hardware emulation, but can support fewer OSs because these OS have to be modified.
- Containers virtualization has by far the best performance but imposes that all the containers share the same kernel.

Outline

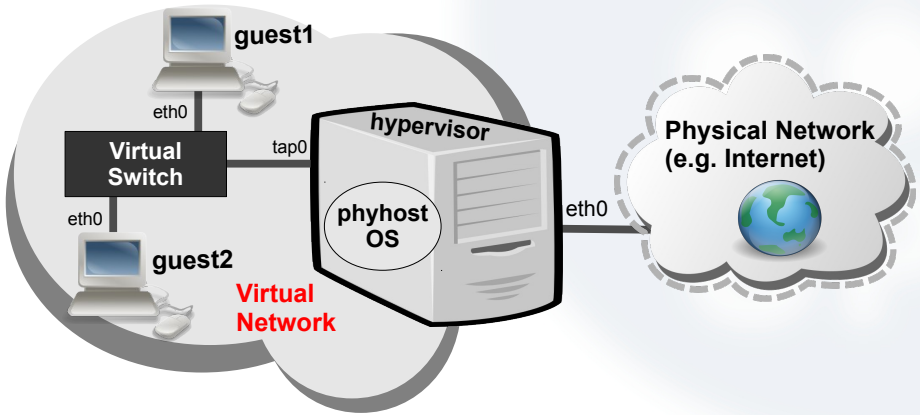
A Physical Topology



If you want to **physically implement** this topology, you need a certain investment for creating the network with real equipment: switches, cables, routers, hosts, servers etc.

Virtual Networks

- In a **virtual network**:
  - Everything is implemented (real) like in a physical network.
  - The difference is that switches, routers, hosts, servers and interconnections are made in software.
  - For example, **virtual switches** are just processes.
- Typically, the **phyhost** also provides **external connectivity** for its guests:
  - This allows guests to exchange traffic with other physical networks and even with the Internet.
  - To do so, the **phyhost** can have a virtual interface connected to the virtual switch.



Why Linux for Building Virtual Networks?

- A Linux system can be used as:
  - Switch or bridge (with `brctl` utils).
  - Router (activating forwarding and also dynamic routing with Quagga).
  - For doing NAT (with `iptables`).
  - As firewall (also with `iptables`).
  - As almost any type of server: DHCP server, DNS server, Web server, etc.
- Everything for free, it is open source!!