

Virtualizing I/O

Yiying Zhang

Administrivia

- Quiz this Wed: cover everything until (including) today's lecture
 - Closed book
 - Quiz will be 15 minutes (T/F, MC, short answers)
- Project proposal due this Thur (1/25), submit on Canvas

Outline

- Software-based memory virtualization
- Hardware-assisted memory virtualization
- Memory management
 - Reclaiming
 - Sharing

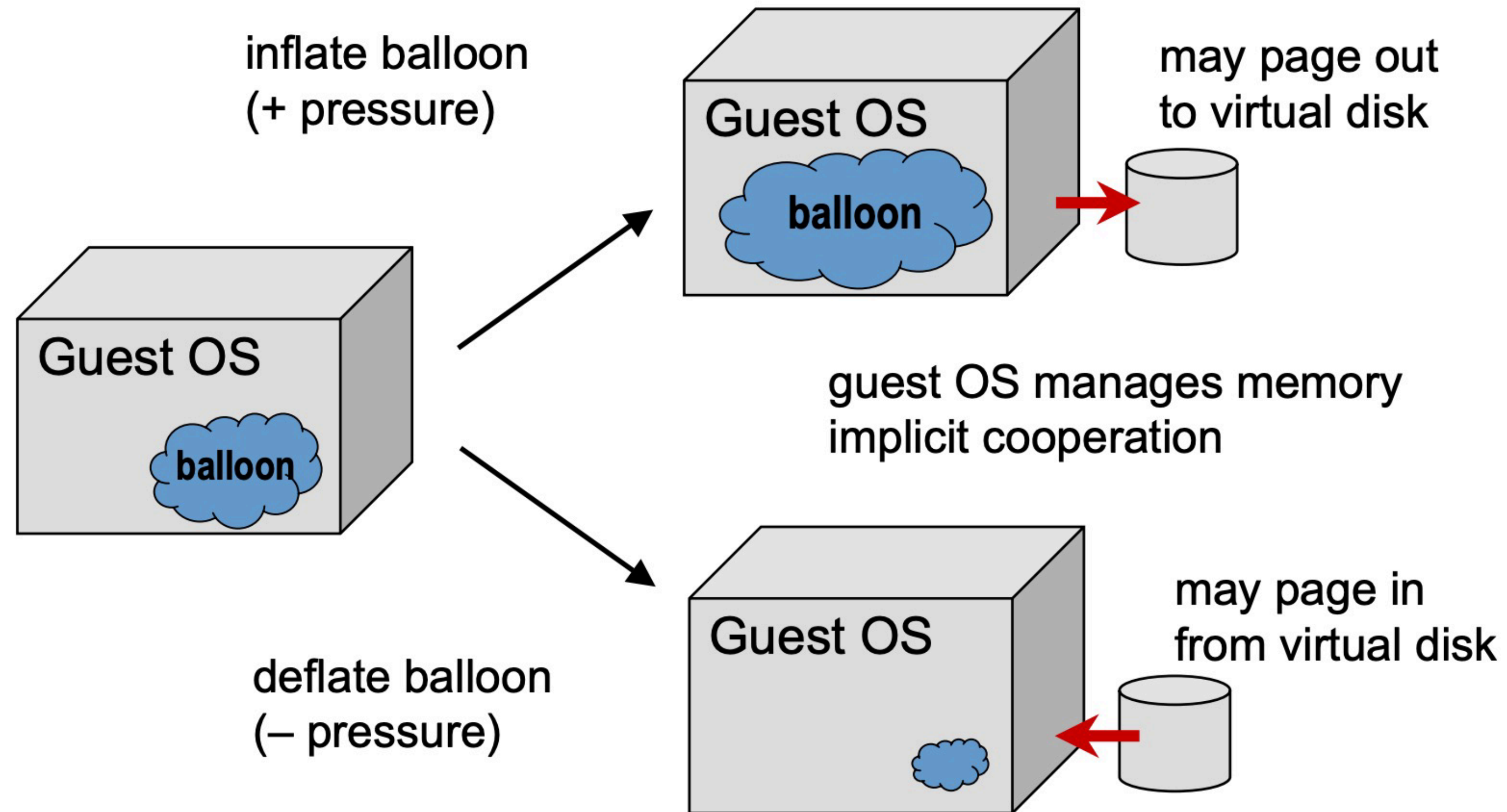
Reclaiming Memory

- ESX (and other hypervisors) allow overcommitment of memory
 - Total memory size of all VMs can exceed actual machine memory size
 - ESX must have some way to reclaim memory from VMs (and swap to disk)

Reclaiming Memory

- Traditional: add transparent swap layer
 - Requires “meta-level” decisions: which page from which VM to swap
 - Best data to guide decisions known only by guest OS
 - Guest and meta-level policies may clash, resulting in *double paging*
- Alternative: implicit cooperation
 - Coax guest OS into doing its own page replacement
 - Avoid meta-level policy decisions

Ballooning



Ballooning Details

- Guest drivers
 - Inflate: guest decides which pages to page out, PPNs communicated to hypervisor via balloon driver
 - Use standard Windows/Linux/BSD kernel APIs
- Performance benchmark
 - Linux VM, memory-intensive dbench workload
 - Compare 256 MB with balloon sizes 32 - 128 MB vs. static VMs
 - Overhead 1.4% - 4.4%

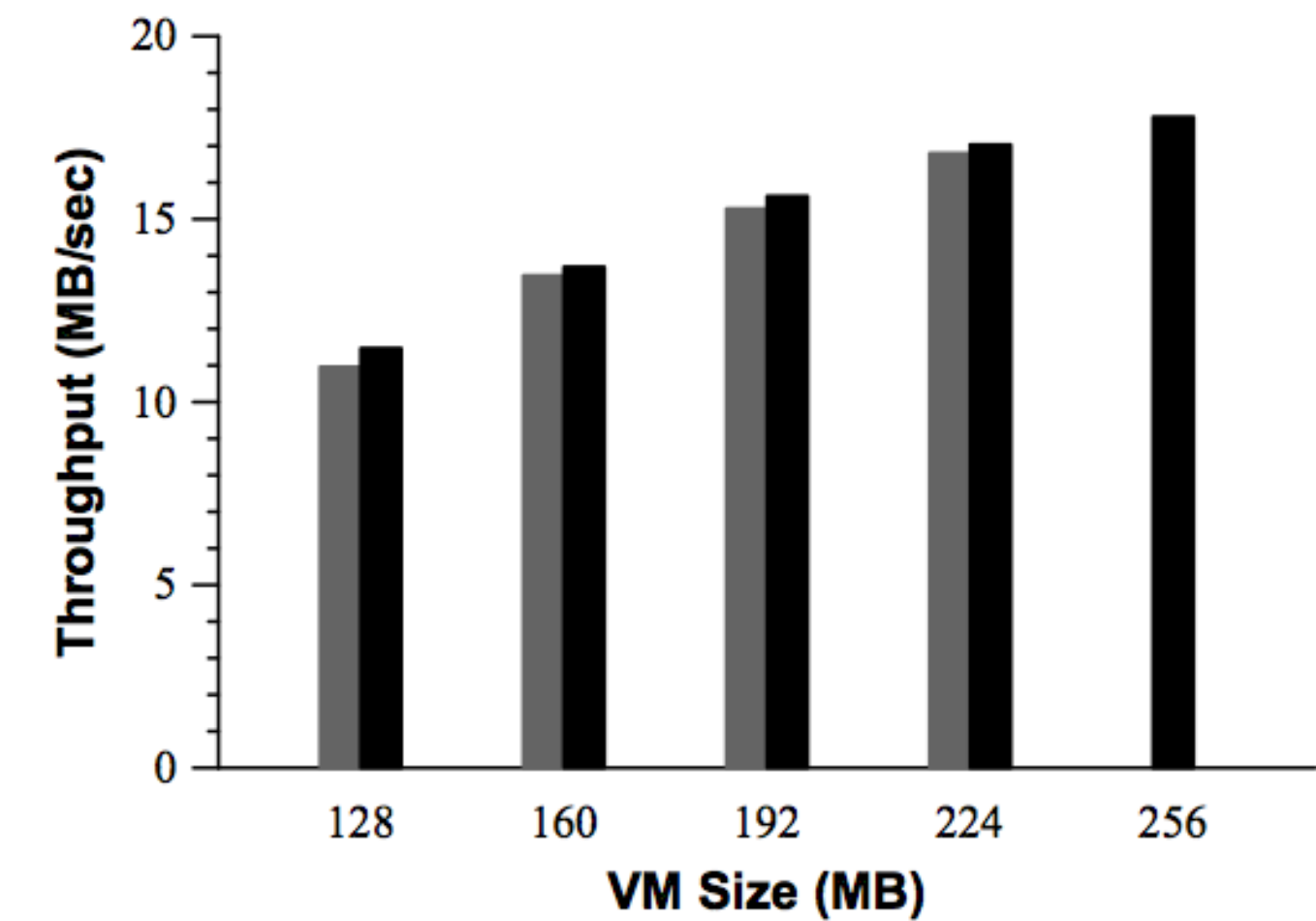
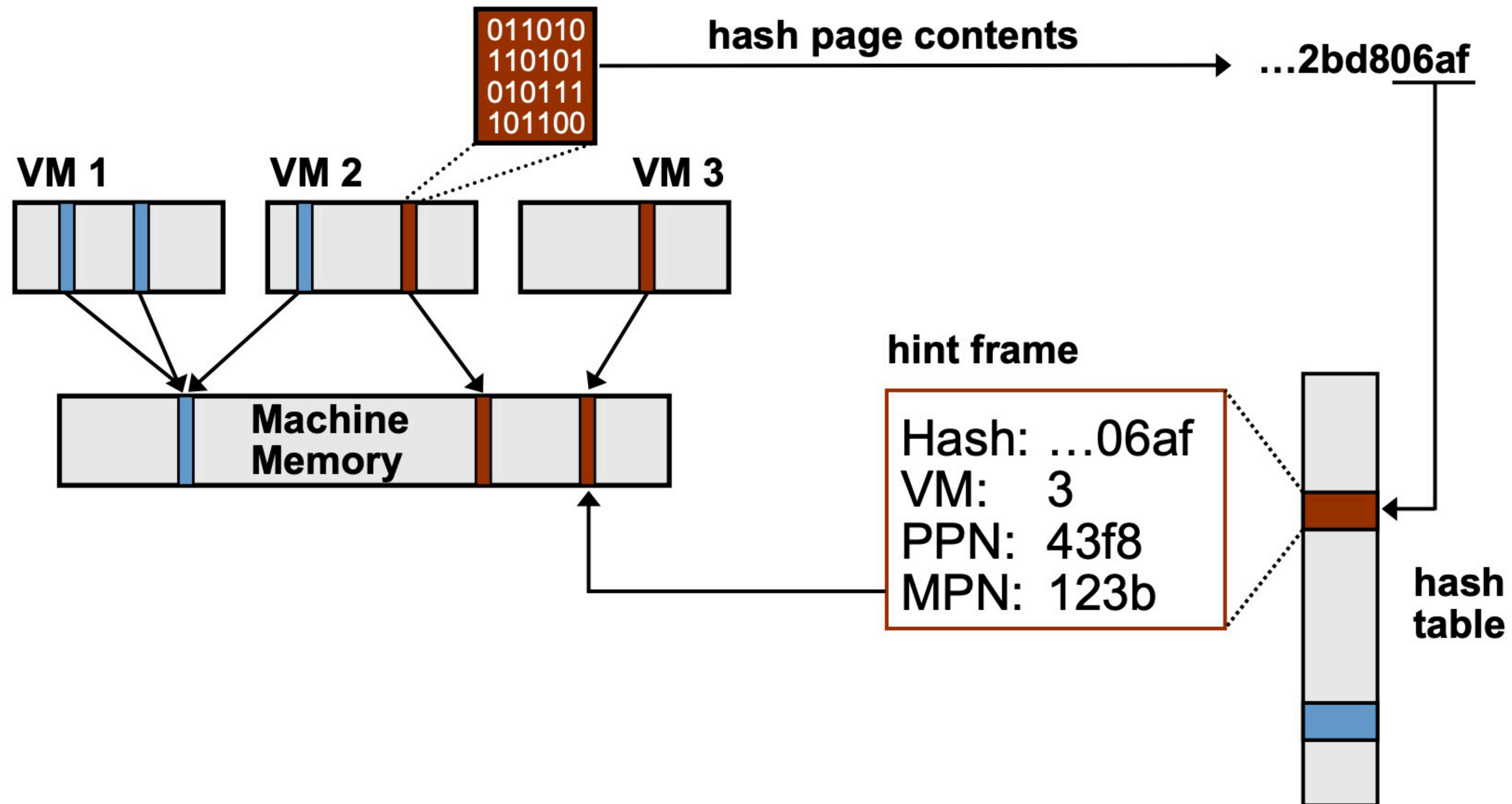


Figure 2: **Balloon Performance.** Throughput of single Linux VM running dbench with 40 clients. The black bars plot the performance when the VM is configured with main memory sizes ranging from 128 MB to 256 MB. The gray bars plot the performance of the same VM configured with 256 MB, ballooned down to the specified size.

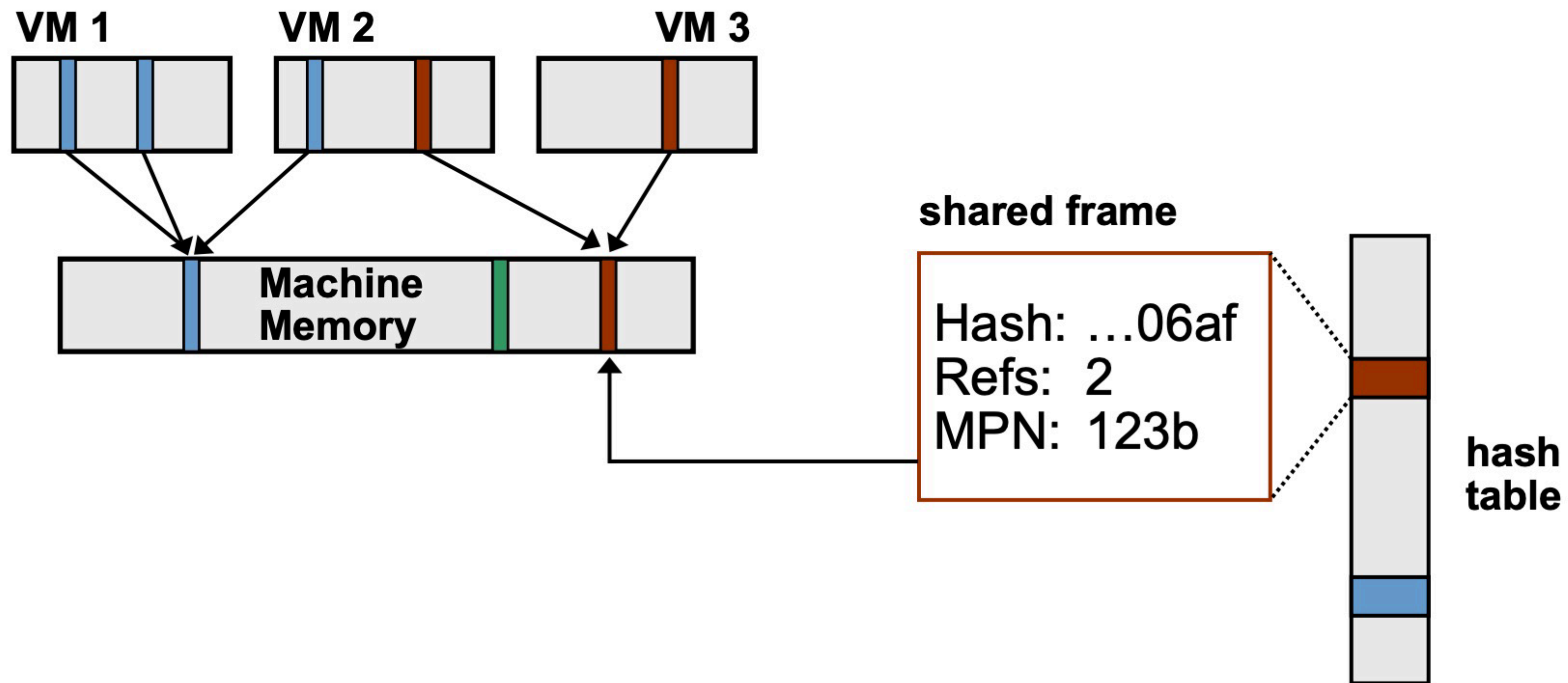
Memory Sharing

- Motivation
 - Multiple VMs running same OS, apps
 - Collapse redundant copies of code, data, zeros
- Transparent page sharing
 - Map multiple PPNs to single MPN (copy-on-write)
 - Pioneered by Disco, but required guest OS hooks
- New twist: content-based sharing
 - General-purpose, no guest OS changes
 - Background activity saves memory over time

Page Sharing: Scan Candidate PPN



Page Sharing: Successful Match



Question

- What is the benefit of keeping a "hint" entry for each scanned (but unshared) page (as compared to not maintaining anything for the page)

Real-World Page Sharing Results

		Total	Saved	
Workload	Guest Types	MB	MB	%
Corporate IT	10 Windows	2048	673	32.9
Nonprofit Org	9 Linux	1846	345	18.7
VMware	5 Linux	1658	120	7.2

Corporate IT – database, web, development servers (Oracle, Websphere, IIS, Java, etc.)

Nonprofit Org – web, mail, anti-virus, other servers (Apache, Majordomo, MailArmor, etc.)

VMware – web proxy, mail, remote access (Squid, Postfix, RAV, ssh, etc.)

Conclusion

- Software and hardware solutions for memory virtualization both have pros and cons
- More things to take care of besides the basic mechanism of memory virtualization
 - Allocation, sharing, overcommitment and reclamation

Outline

- Device virtualization techniques
- Datacenter network virtualization

Acknowledgment: some slides from Scott Devine's lectures at Columbia, Teemu Koponen's NSDI'14 presentation, and Ben Pfaff's Network Virtualization lecture

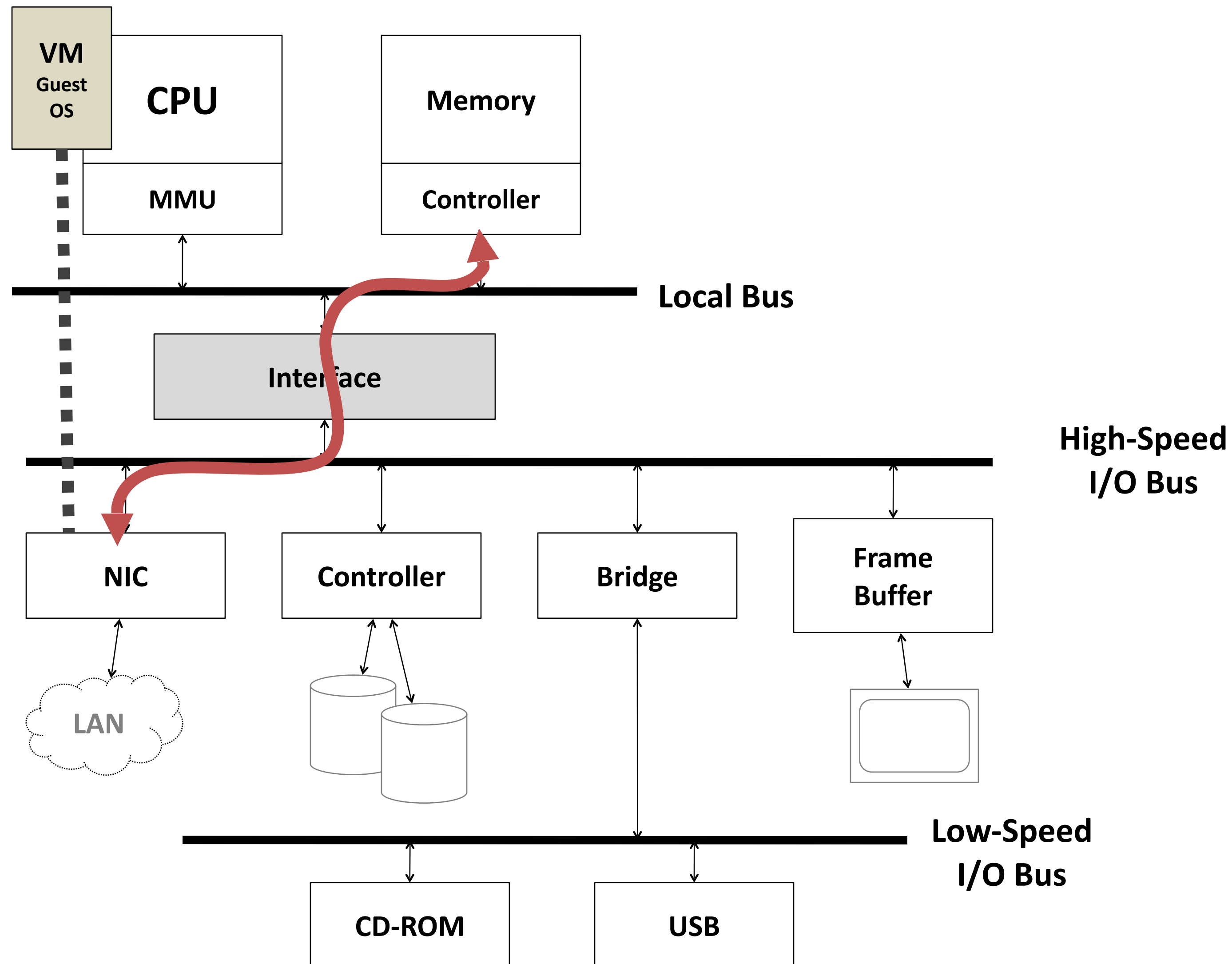
I/O Virtualization

- Goal
 - Multiplexing device across guest VMs
- Challenges
 - Each guest OS has its own device driver
 - How can one device be controlled by multiple drivers?
 - What if one guest OS tries to format its disk?

Possible Solutions of I/O Virtualization

- Direct access: VM exclusively owns a device
- Device emulation: VMM emulates device in software
- Para-virtualization: split driver into guest part and host part
- Hardware assisted: hardware devices offer isolated “virtual interfaces”

Sol-1: Direct Access Device Virtualization



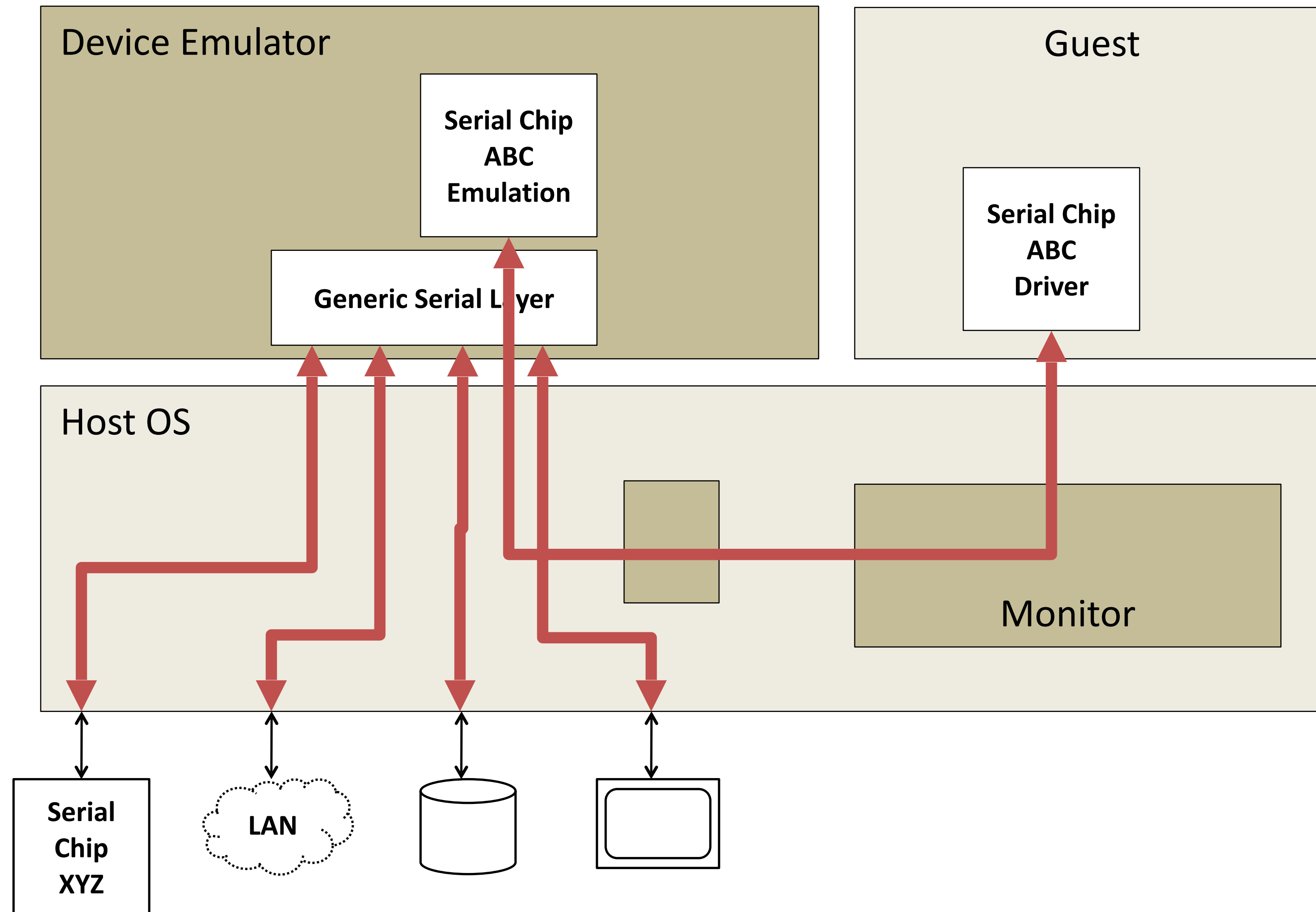
Direct Access Device Virtualization

- Positives
 - Fast, since the VM uses device just as native machine
 - Simplify hypervisor: limited device drivers needed
- Negatives
 - Hardware interface visible to guest (bad for migration)
 - Interposition is hard by definition (no way to trap & emulate)
 - Now you need much more devices! (imagine 100 VMs)

Sol-2: Emulating Devices

- Emulate a device
 - Implement device logic in pure software
 - Can even emulate non-existing devices

Example: Emulating Serial Port



Emulated Devices

- Positives
 - Platform stability (good for migration)
 - Allows interposition
 - No special hardware support is needed
- Negatives
 - Can be slow (it's software emulated)

Virtual Disk

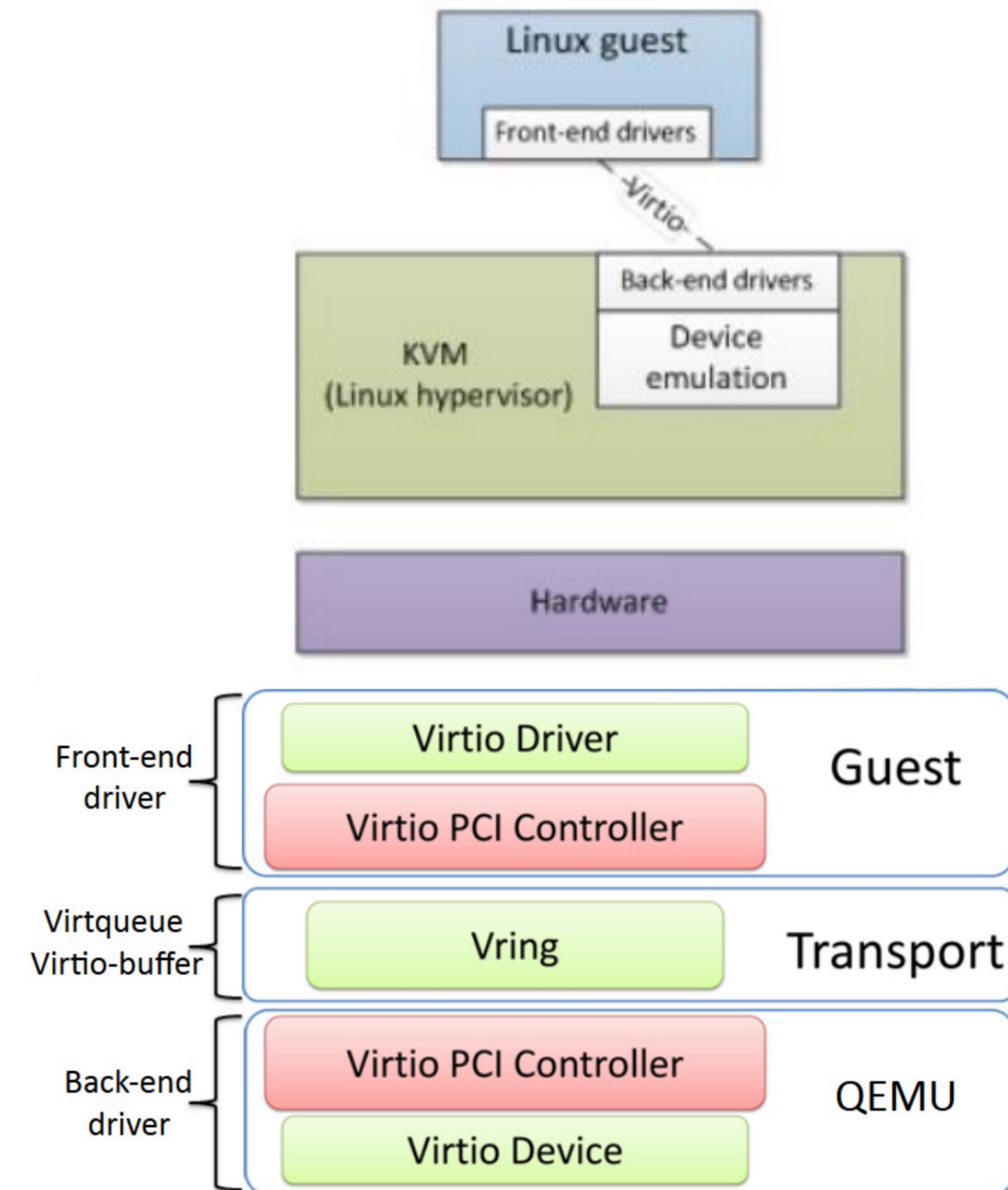
- A virtual disk is just a file in host file system
- Hypervisor maps disk blocks to file offsets
 - Flat file (fix sized virtual disk)
 - Indexed file (virtual disk can grow on demand)

Sol-3: Para-Virtualized Devices

- VMM offers new types of device
- The guest OS runs a new driver (*front-end driver*)
- VMM runs a *back-end* driver for each front-end
- VMM finally runs the real device driver to drive the device

virtio: Linux's paravirtualized I/O solution

- Front-end Driver
 - A kernel module in the guest OS
 - Accepts I/O requests from the user process
 - Transfer I/O requests to back-end driver
- Back-end Driver
 - Accepts I/O requests from front-end driver
 - Perform I/O operation via physical device
- Virtqueue
 - A memory region accessible from both guest and host OS
 - An interface implemented as vring

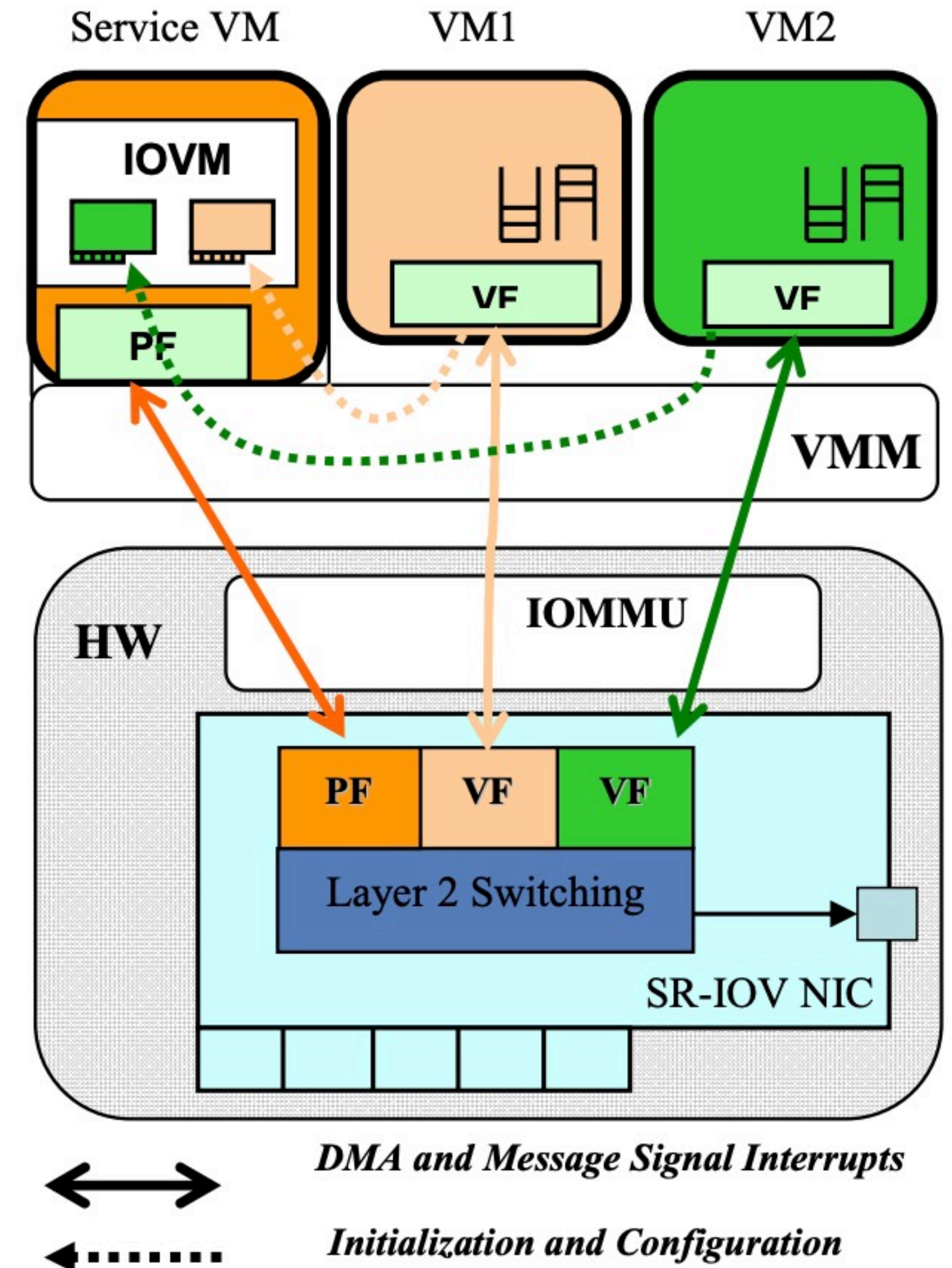


Sol-4: Hardware Support for I/O Virtualization

- A virtualization-enabled device can be configured to appear in the PCI configuration space as multiple virtual functions (VFs)
- The VMM assigns one or more VFs to a VM by mapping the actual configuration space of the VFs to the configuration space presented to the VM by the VMM

SR-IOV (Single Root I/O Virtualization)

- Hardware support for guest to access device without going through VMM
- Physical Function (PF)
 - A standard PCIe function
 - Can be associated with multiple VFs
- Virtual Function(VF)
 - A lightweight PCIe function (a unique PCIe xact source)
 - Each VF is isolated from other VFs
 - Has dedicated access to certain hardware resources
 - Share some other resources
- IOVM and PF driver: set up VFs and provide full features to each VF



Virtualization Technologies Summary

Virtualization	Software Solution	Hardware Solution
CPU	• Trap & Emulate • Instruction interpretation • Binary translation	• VT-x • Root / non-root mode • VMCS
	• Para-virtualization	
Memory	• Shadow page table	• EPT
	• Para-virtualization	
Device	• Direct I/O • Device emulation	• SR-IOV
	• Para-virtualization	

Outline

- Device virtualization techniques
- Datacenter network virtualization

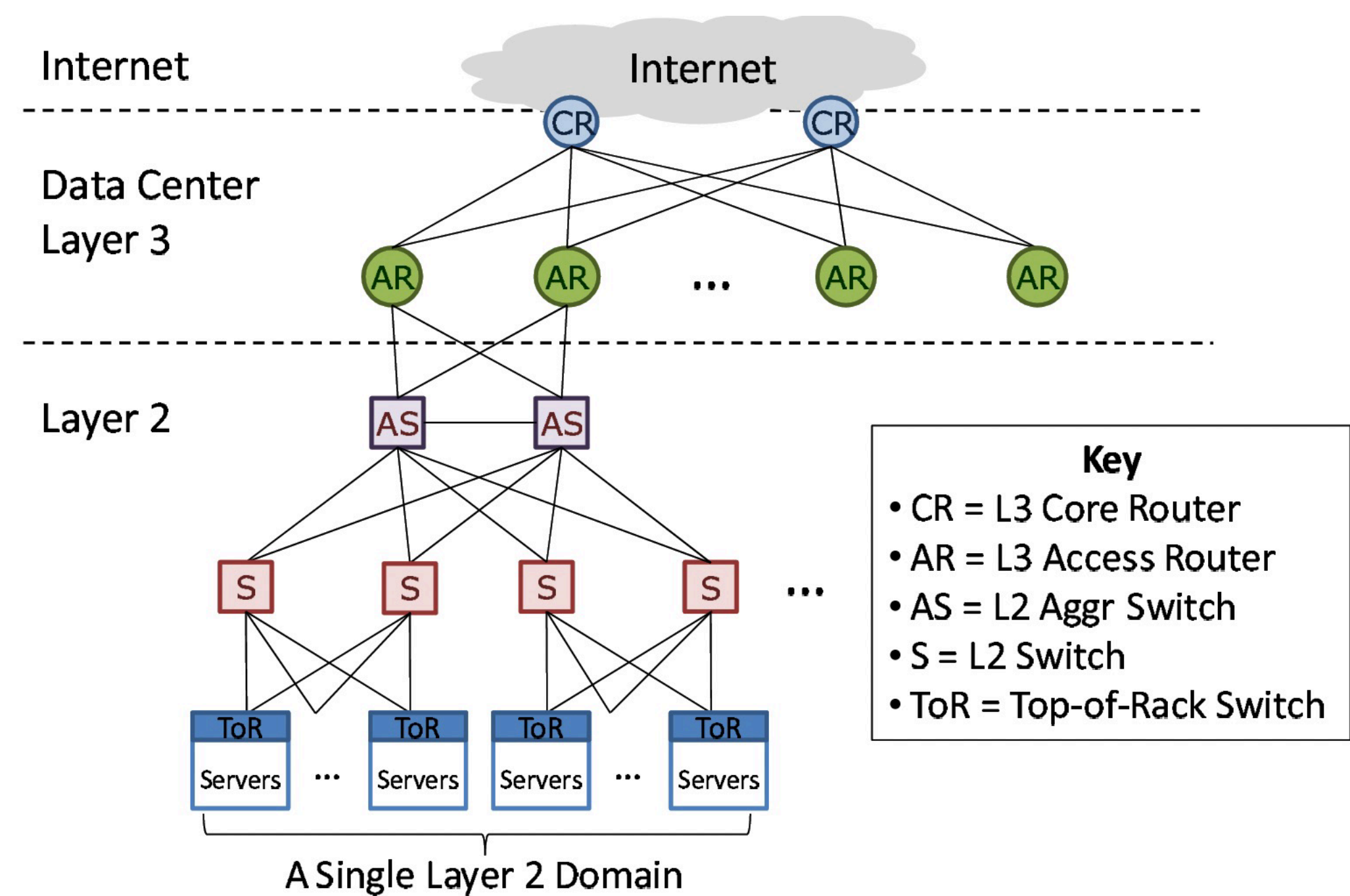
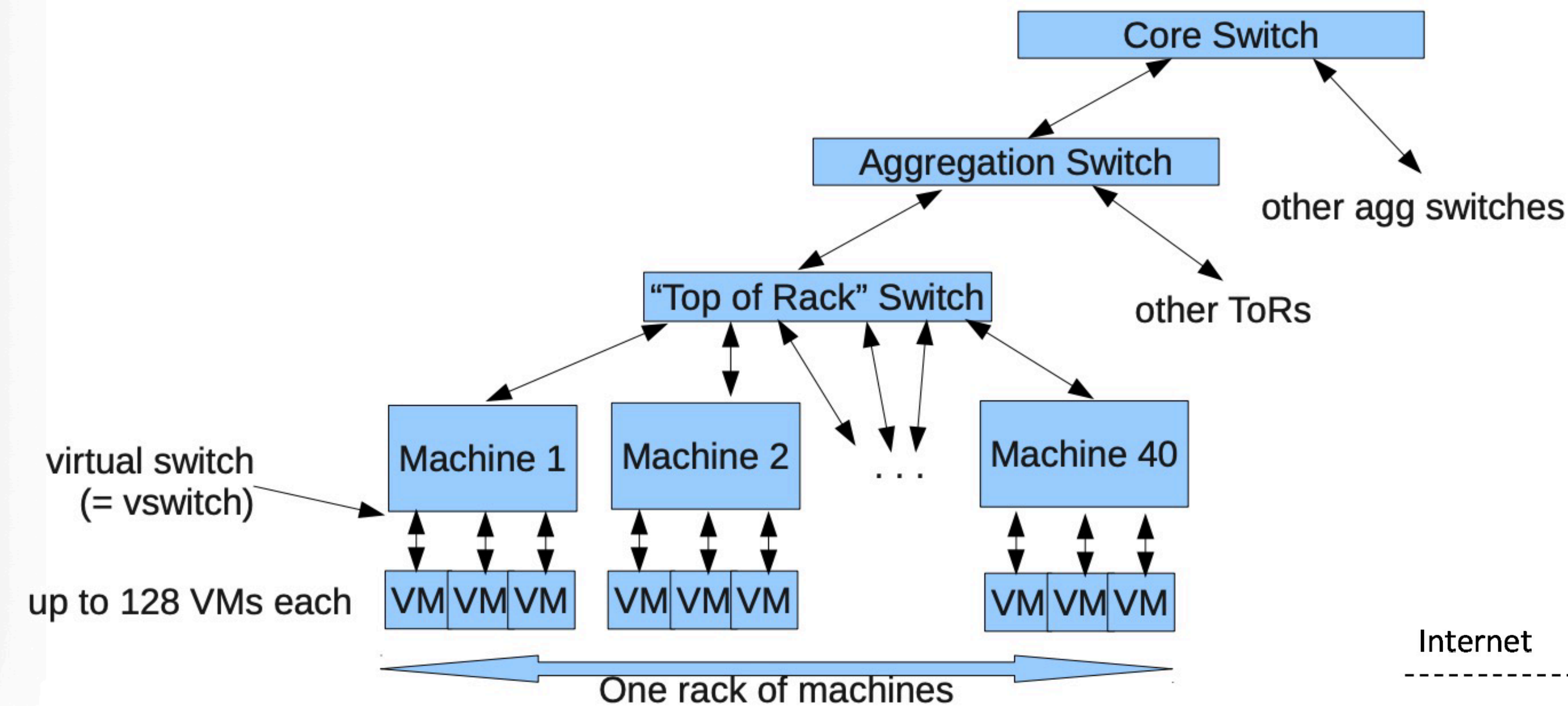
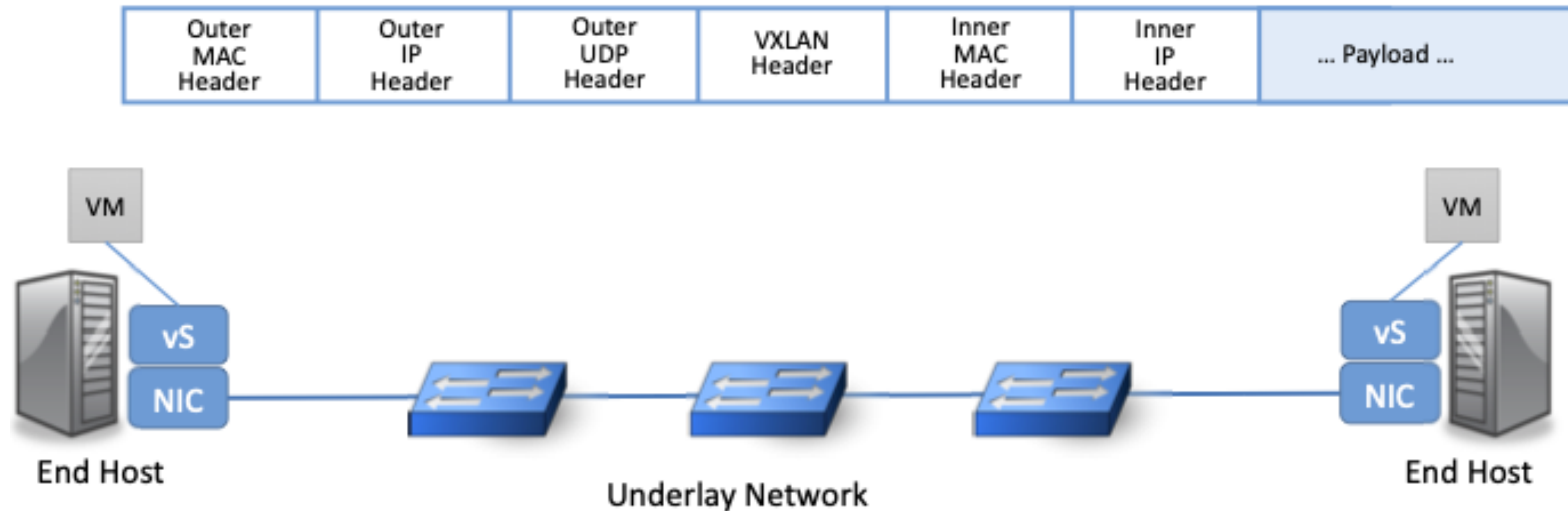


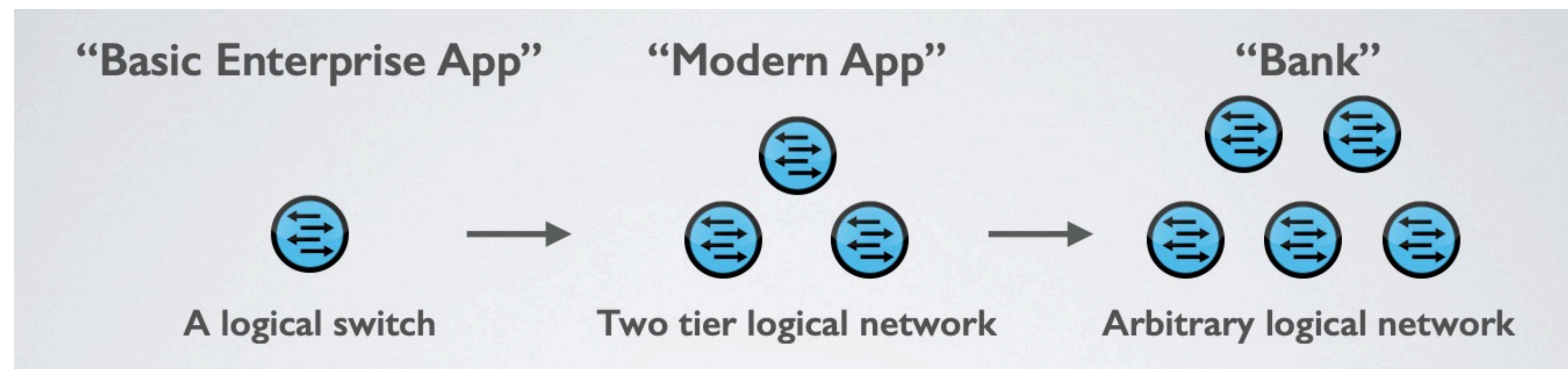
Figure 1: A conventional network architecture for data centers (adapted from figure by Cisco [5]).

Virtual Network Encapsulation (Data Plane Solution)

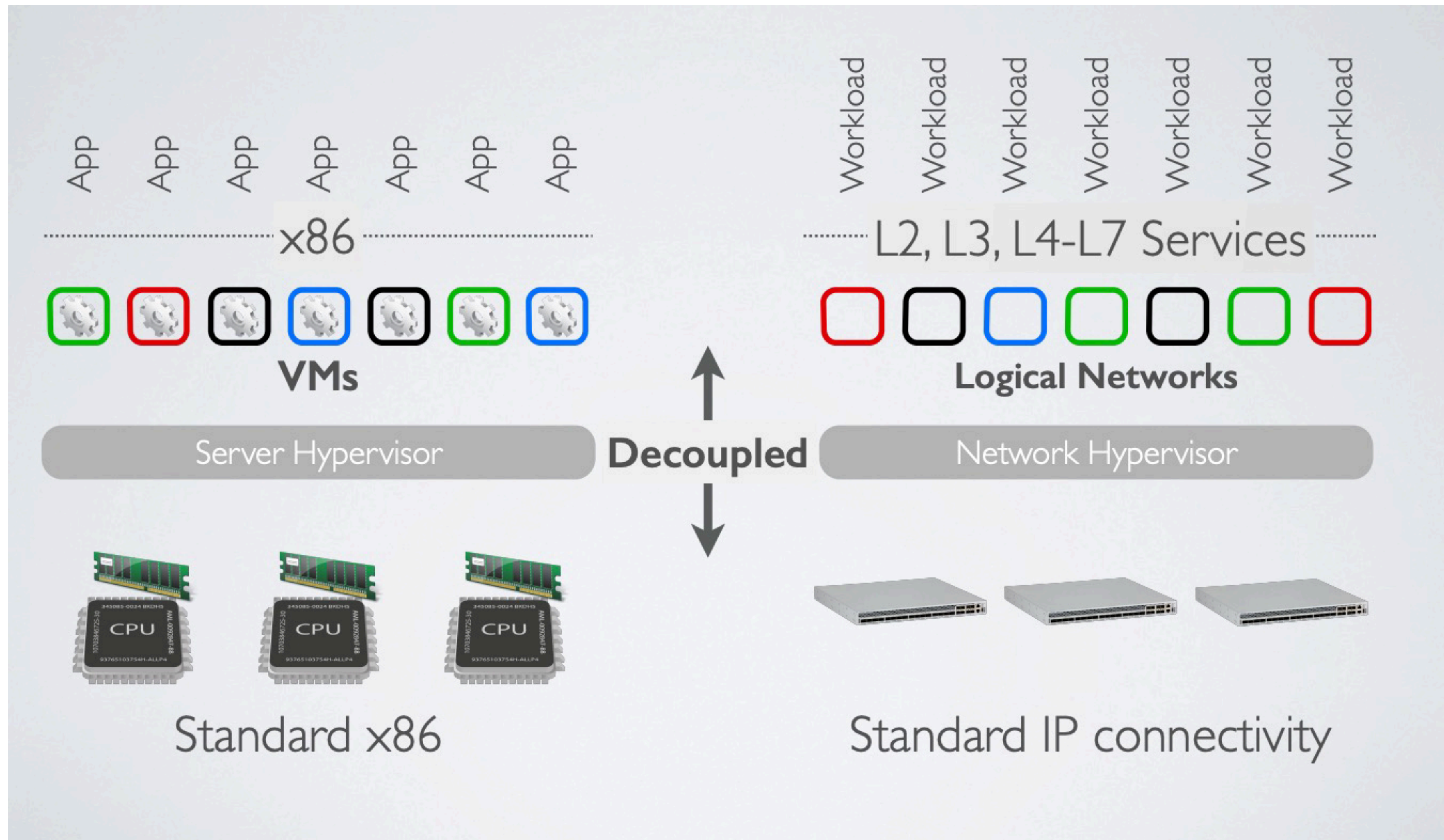


Problems?

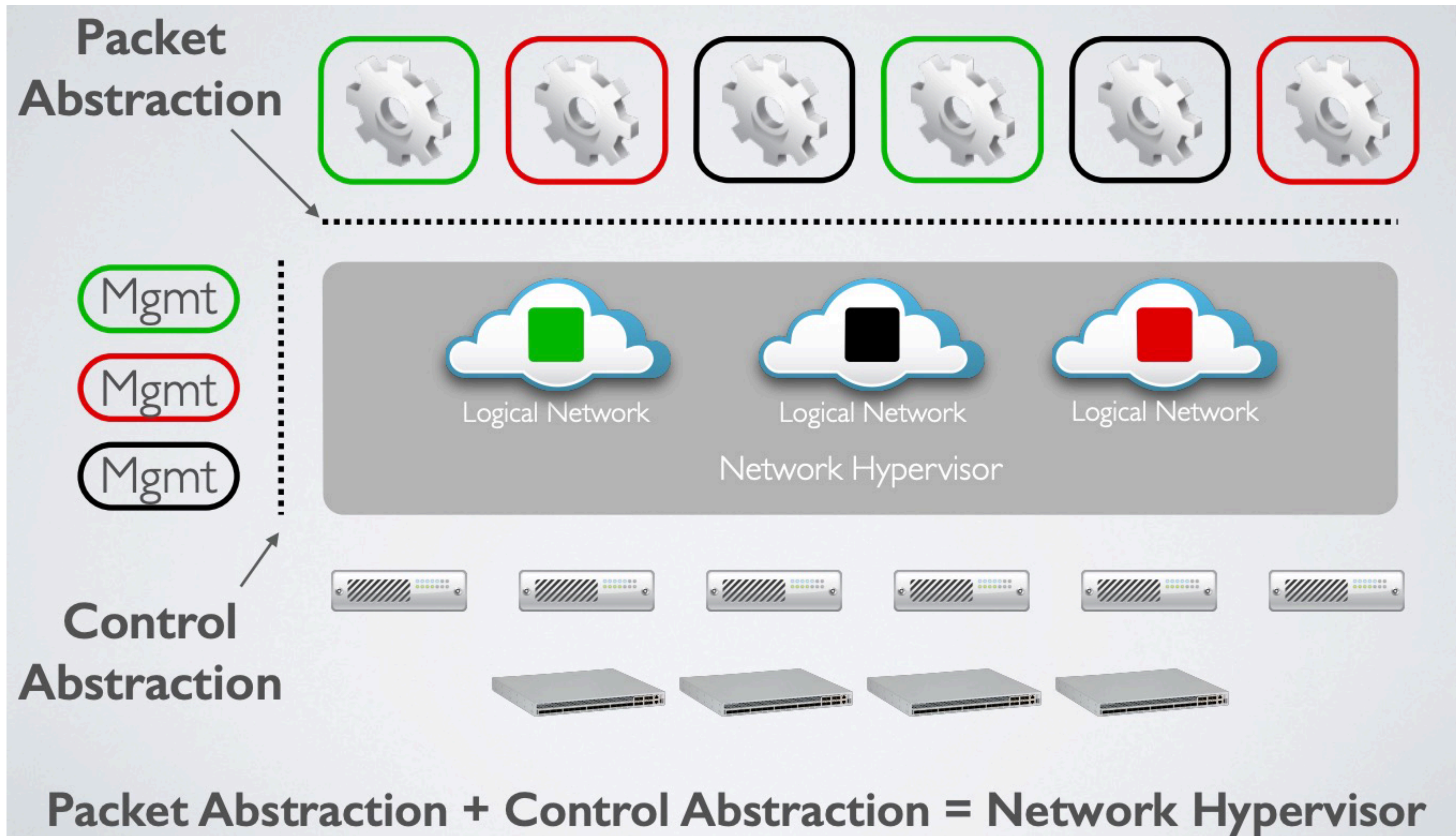
- A physical topology is hard to support different virtual topologies
- Virtualized workloads stay in the physical network address space (L2)
 - Slow provisioning
 - Limited mobility
 - Limited VM placement
 - Hardware dependent
 - Operationally intensive



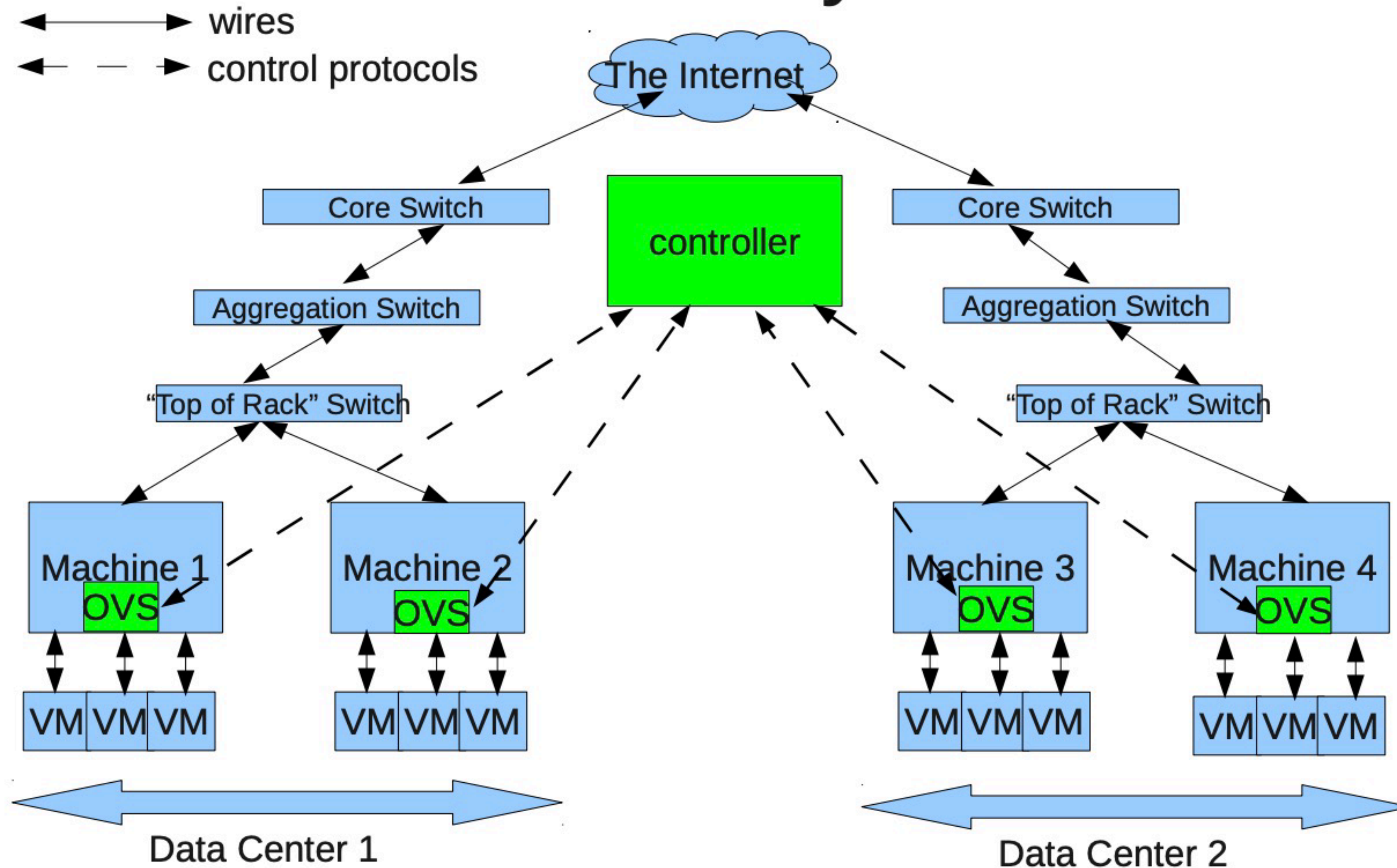
Goals (Paraphrase with Machine Virtualization)



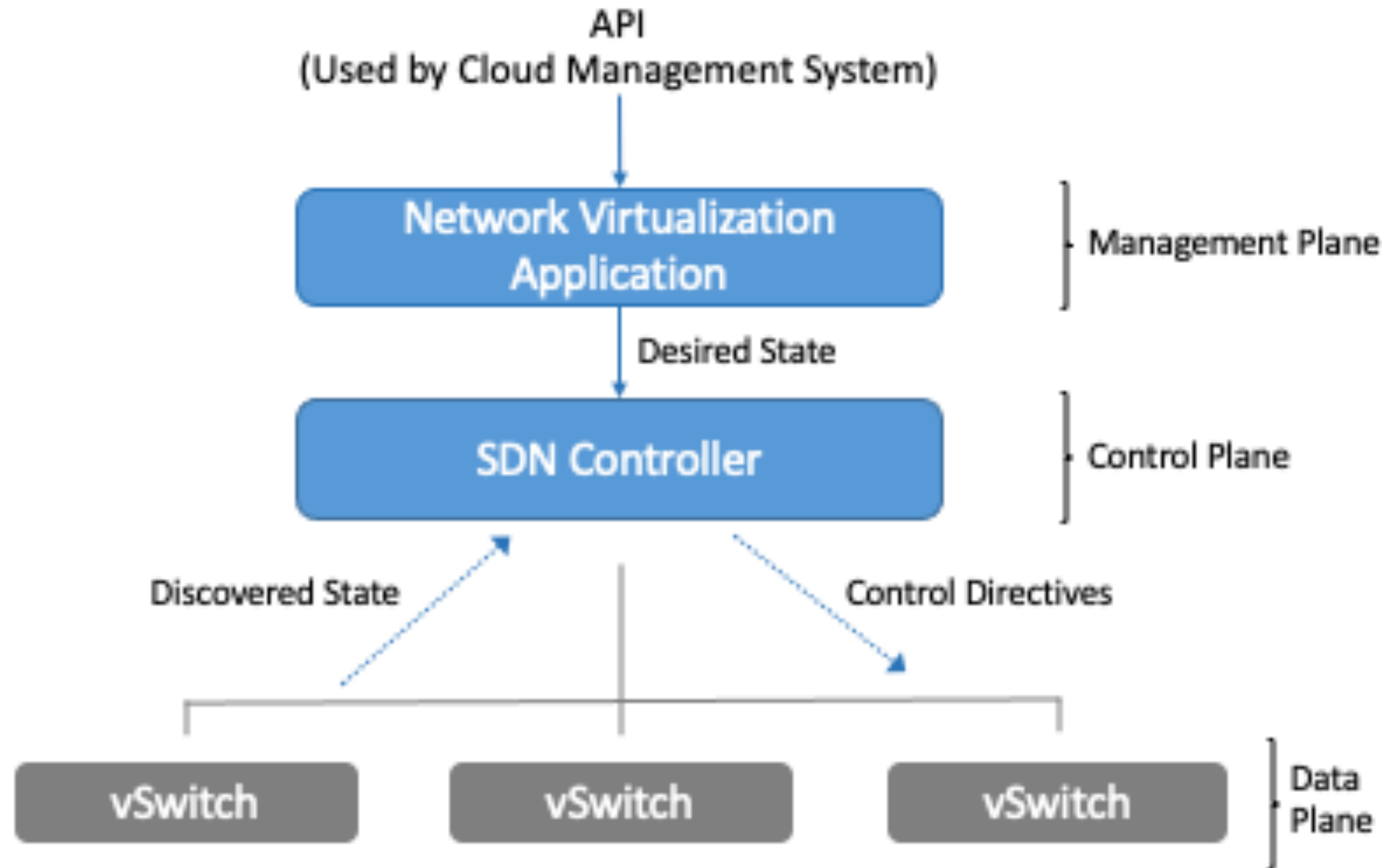
Network Virtualization



A Network Virtualization Distributed System



Management, Control, and Data Planes



Distributed Network Services

