Technical Fundamentals of Cloud Systems (3)

Cloud Computing

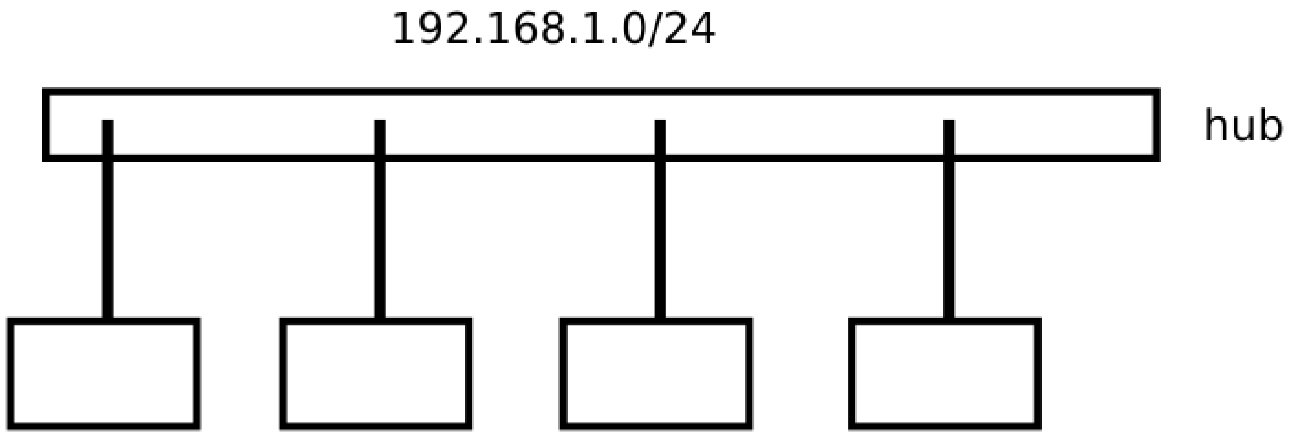
Universiteit Leiden

Networking

- The next thing we need is networking.
- Without network connection, we cannot reach the VM through the network.
  - Which is one of the requirements of Cloud Computing.
- First a very brief review of physical computer networking.

Computer Networks

- A **hub** sends out a received packet to all stations connected to this hub.
  - It “floods” packets to all connected stations on this segment, all sharing an IP subnet.
  - Successor of old token ring and coax network cabling.

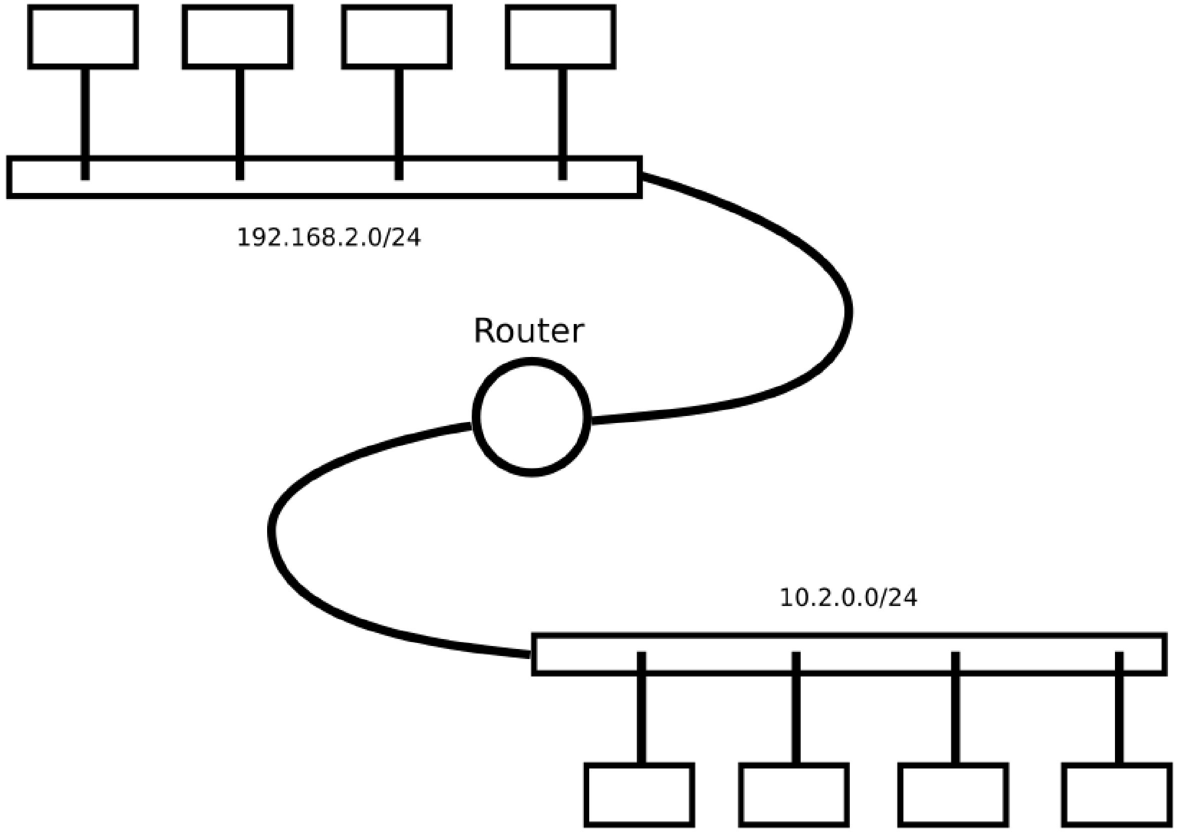


Computer Networks (2)

- In contrast, a **switch** learns to which ports certain stations are connected.
  - For a received packet, it knows where (to which particular port) to forward.
  - Done based on MAC address (OSI layer 2).
  - It if doesn't know, it will flood the packet and learn.
  - Saves bandwidth, saves collisions.
- A simple switch handles a single segment.

Computer Networks (3)

- Routers route packets between different network segments / IP subnets.



Computer Networks (4)

Traditional networking gear:

- High-end, expensive hardware
  - Custom hardware (ASIC) to achieve high forwarding and routing speeds
  - Up to 100Gbit/s ports
- Stand-alone device, communicates with neighbors
- Device also handles device management
  - Typically equipped with embedded processor to control the device
  - Configuration through Command Line Interface; vendor-specific
- Software included (proprietary)
  - You rely on the vendor for updates

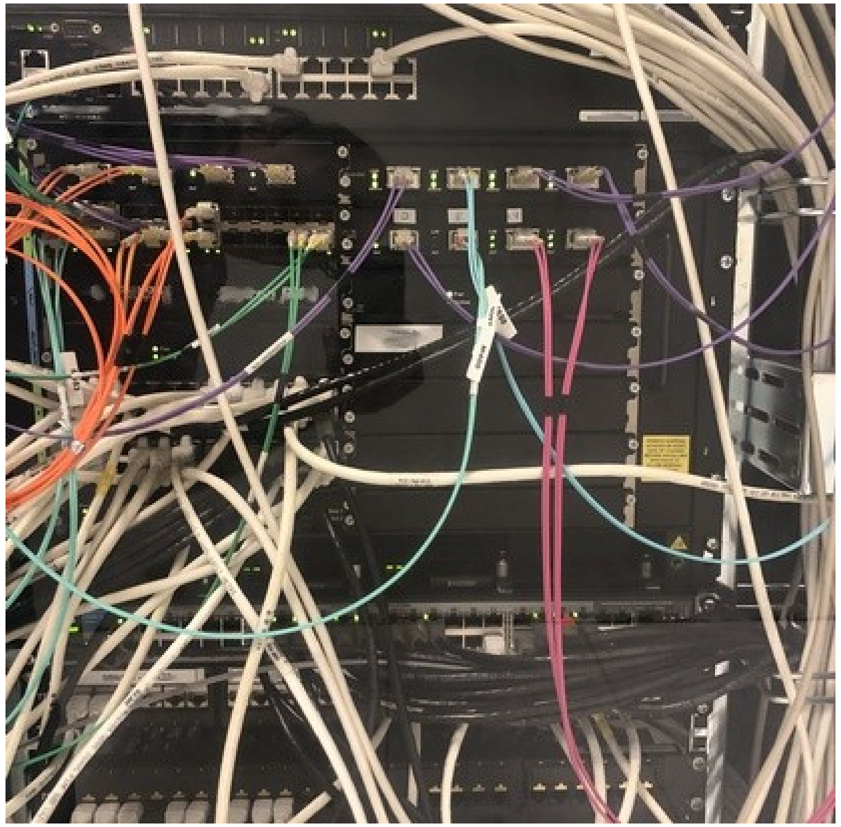


Image: Own work K. Rietveld

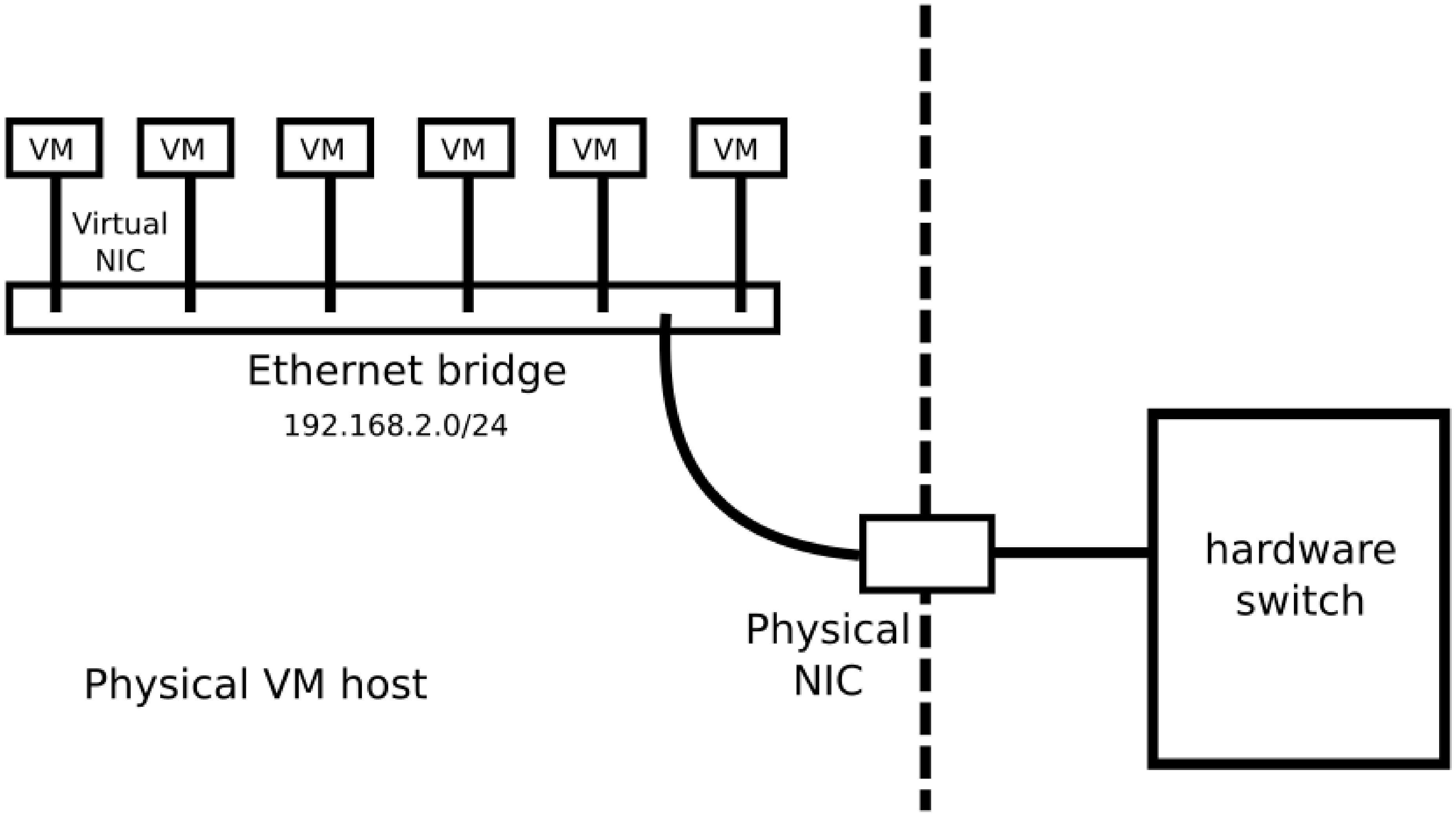
Interconnecting VMs

- Connecting physical computers is easy: you simply need to make a physical connection between computer and switch.
- Now what about VMs?
  - We could pass through a physical network interface to each VM.
  - Clearly, this doesn’t scale.

Interconnecting VMs

How do we interconnect ***N*** VMs running on a single physical computer?

- We give each VM at least one virtual network interface.
- All of these virtual interfaces are connected to an Ethernet bridge (a switch implemented in software).
- At least one physical network interface of the host is added to this bridge.
- Now all VMs have a path towards the physical network.


VM
VM
VM
VM
VM
VM
Virtual NIC
Ethernet bridge
192.168.2.0/24
Physical VM host
Physical NIC
hardware switch

Interconnecting VMs (2)

- All machines connected to the bridge share the same subnet.
  - The VMs could be assigned internal IPs, for use within an internal network.
  - Or public IPs, for public network. Works fine for singular VPS hosting public clouds.
- However, what about tenants with multiple VMs that must remain private?
  - E.g. moving servers hosted within an office network, or at own dedicated data center hosting, into the cloud.

Interconnecting VMs (3)

- If you want to have private VMs on a private network, using same subnet as other tenants not an option.
  - Other tenants have root access; could intercept traffic, mess with ARP, etc.
- So, you want to have the ability to create virtual networks, that are isolated from any other network created in the cloud.
  - And this all through a web-based control panel, in the spirit of clouds.

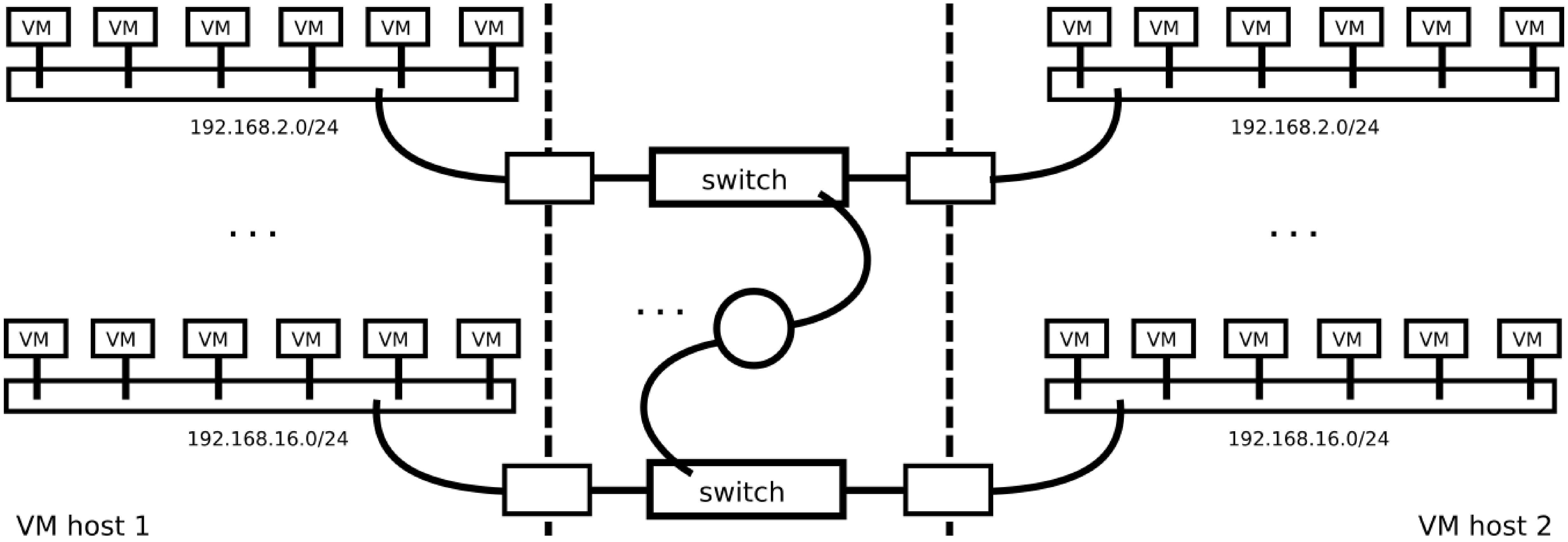
Creating Virtual Networks

- Cloud vendors offer this as a service. E.g. Google’s Virtual Private Cloud.
  - You can create a private IP subnet and connect your VMs to this subnet. Virtual Internal Network.
  - Multiple subnets can be created. Subnets can only span a single zone/region.
    - Why? Billing. Network traffic between different zones is charged per GB.
  - Through Cloud VPN (IPSec connections), you can connect physical office networks to this internal cloud environment.

Creating Virtual Networks (2)

The question now becomes: how to implement this?

- If we would stay within a single physical machine, we could simply add Ethernet bridges on the fly for new virtual networks.
- But we want to use multiple VM hosts.
  - Scaling, failover, etc., etc.
- This implies that different bridges at different physical VM hosts need to be interconnected.

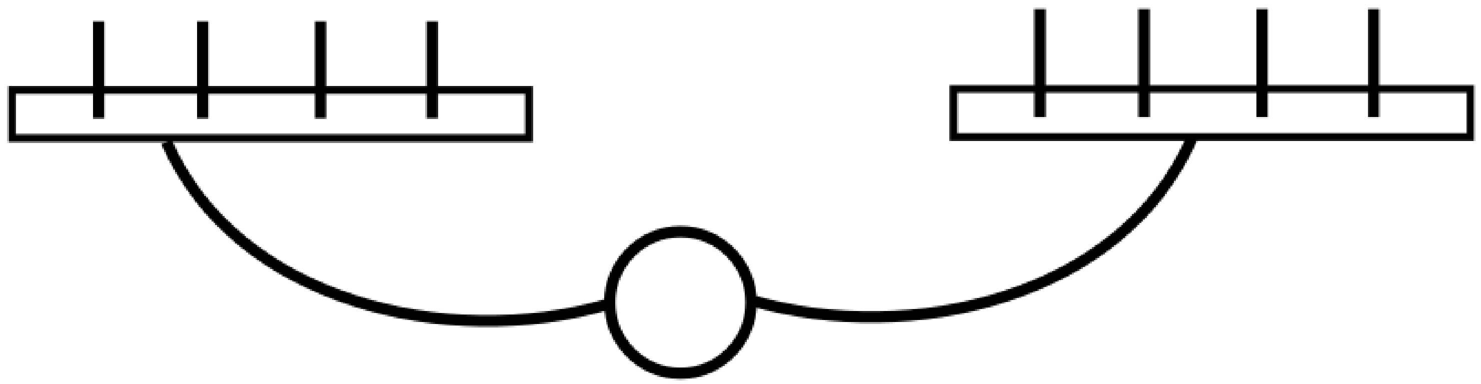


Note that in this case, physical network interfaces, as well as switches need to be replicated. One for each virtual network.

Can we do better? Can we virtualize the hardware switches?

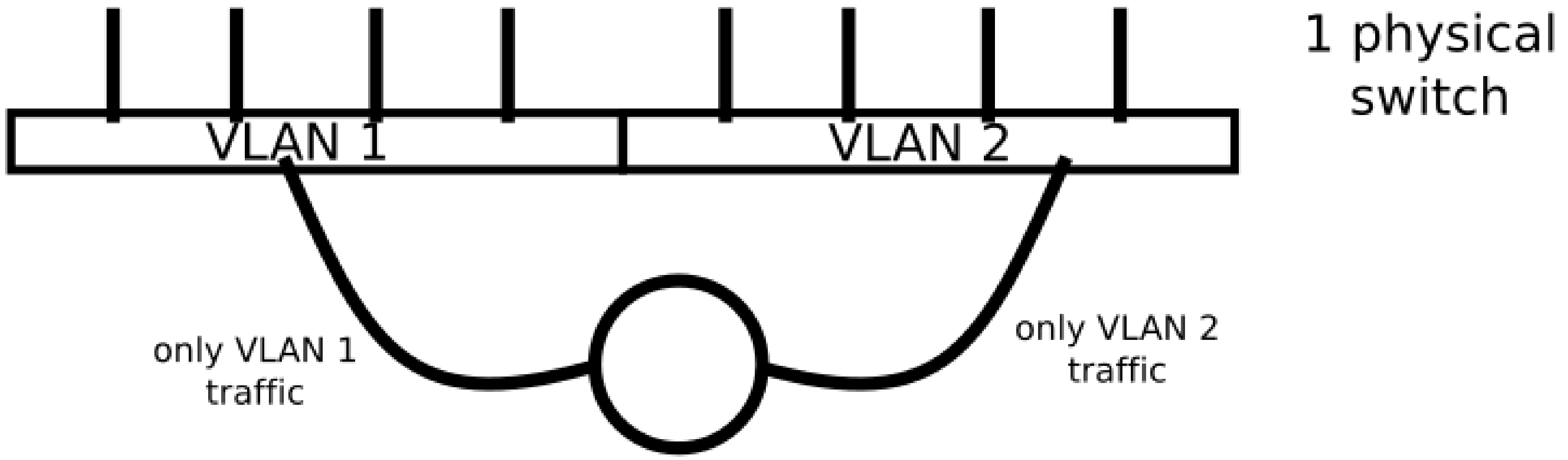
Virtualized Switches?

- Let us consider a small office network with two network segments (IP subnets).
- We need two switches and a router in between.



Virtualized Switches (2) ?

- Can we run both networks safely on a single switch?
- So, create two ***isolated*** network segments on shared physical equipment?



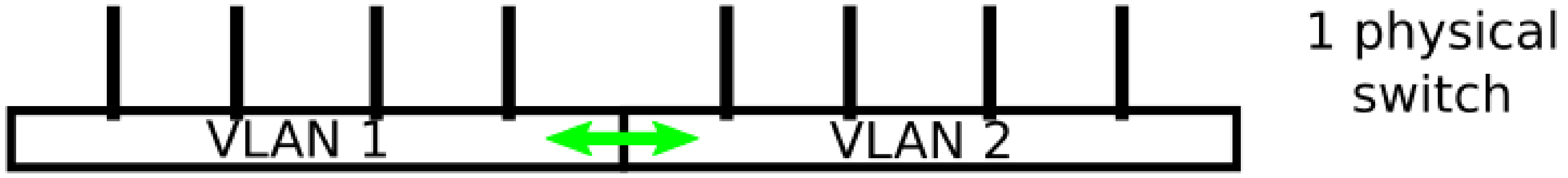
VLANs

Network hardware has the notion of VLANs

- Allows one to create private and isolated networks on shared physical infrastructure (so running on the same switch).
- Achieved by tagging Ethernet packets with a VLAN number
  - In this small office case, the tagging is only done internally on the switch.
  - So packet enters switch, is tagged, leaves switch and is untagged.
  - Traffic over the wire to/from the router & stations is unmodified.
  - This means we could move to a single physical switch, without modifying any of the other components!
- Maximum number of logical networks: 4094

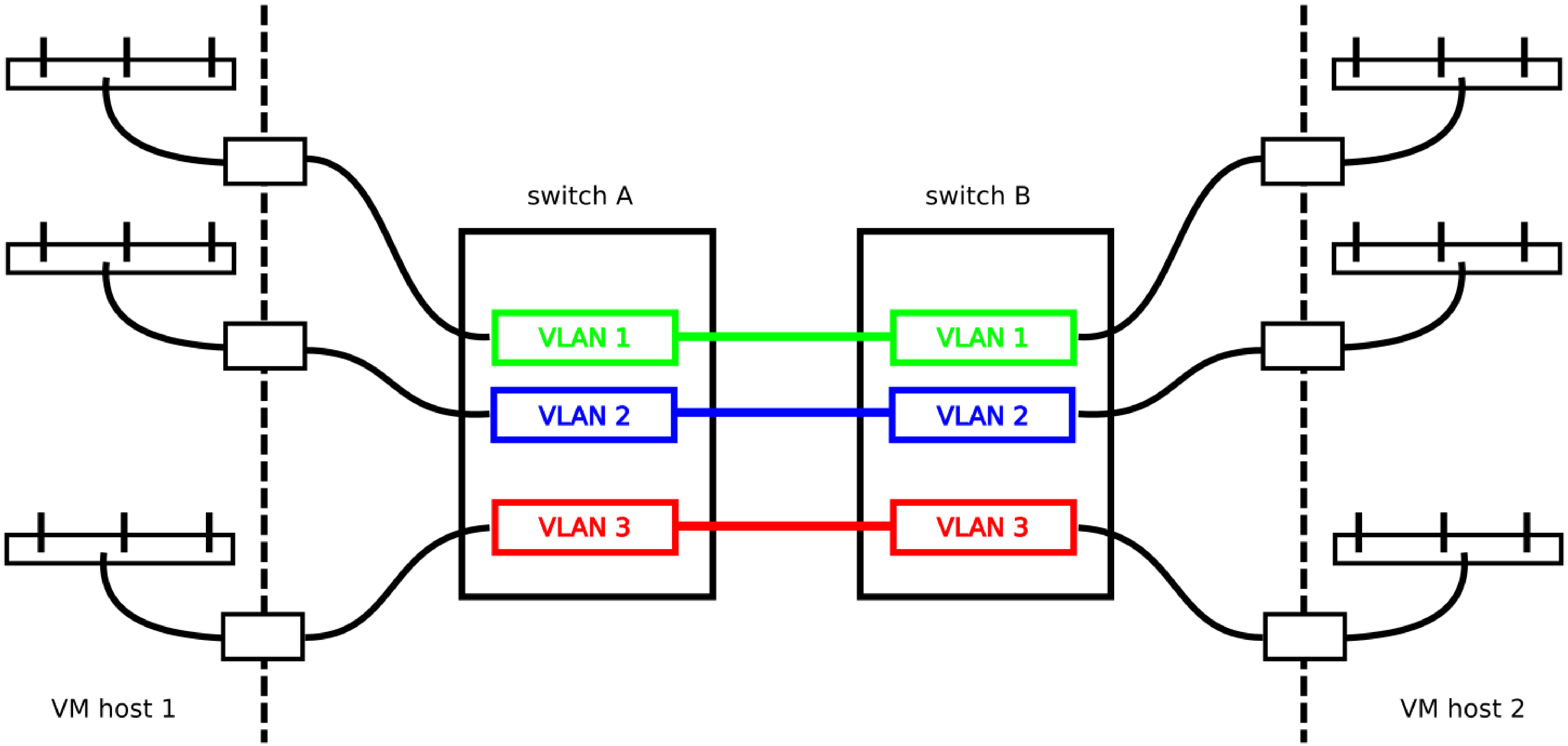
Layer-3 switches

- We can even eliminate the physical router, by moving to a **layer-3 switch**.
- A layer-3 switch can also perform IP routing in addition to switching.
- OSI model:
  - Layer-2: operates at data link layer (so MAC addresses)
  - Layer-3: operates (also) at network layer (so IP addresses)



Creating Virtual Networks (3)

- We can now go back to our running example and eliminate the multiple physical switches and router. See the next slide.
- A VLAN is created for each virtual network.
- Still, we need:
  - Multiple physical network interfaces in the virtual machine hosts.
  - A physical link between different switches in the data center for each VLAN.

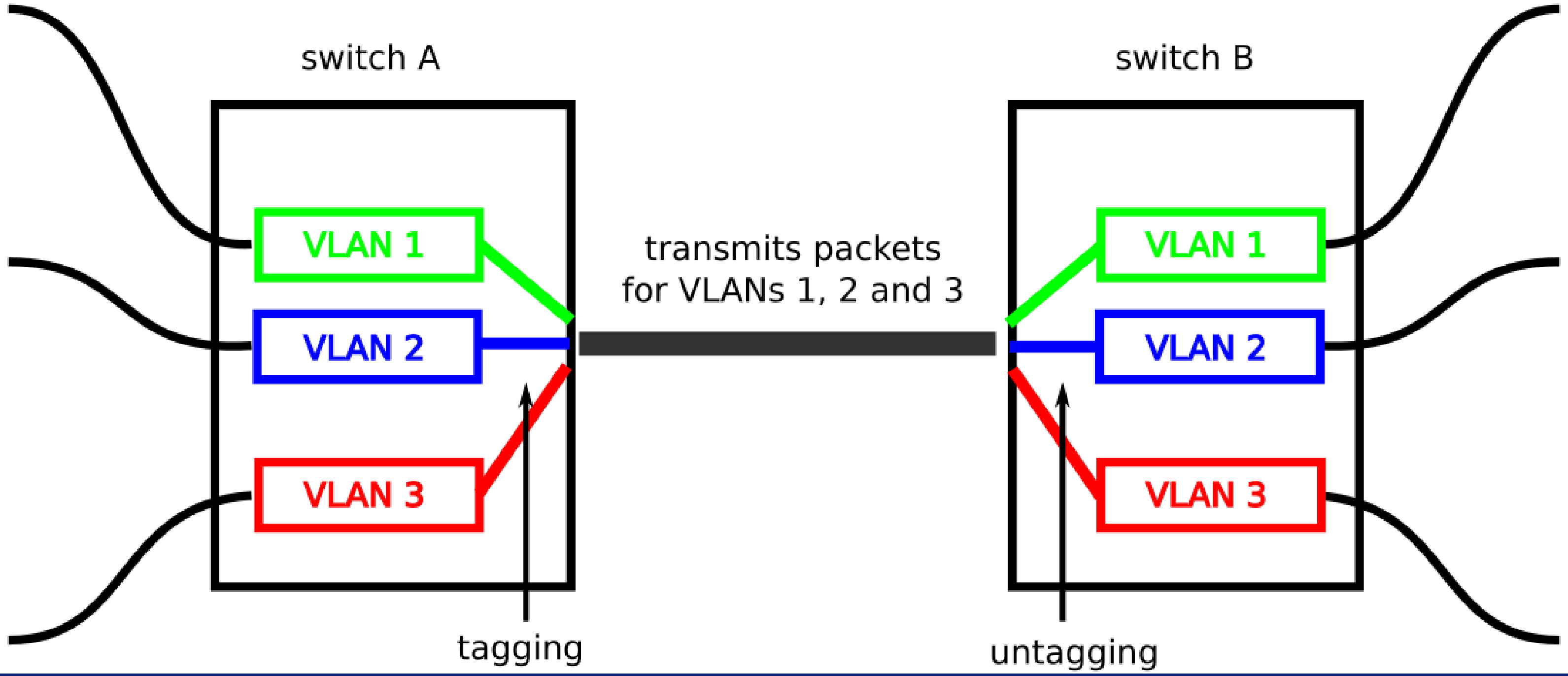

switch A
switch B
VLAN 1
VLAN 2
VLAN 3
VLAN 1
VLAN 2
VLAN 3
VM host 1
VM host 2

VLANs (2)

- In our small office example, we created VLANs on a single switch.
  - Isolated networks on a single physical switch.
- Now, we would actually like to create isolated networks that are transported over a single physical transport medium (or wire/fiber).
  - Goal: go from a separate link for each VLAN, to a single link.
- This is possible by performing VLAN tagging external to the switch, contrary to only internally as we saw before.

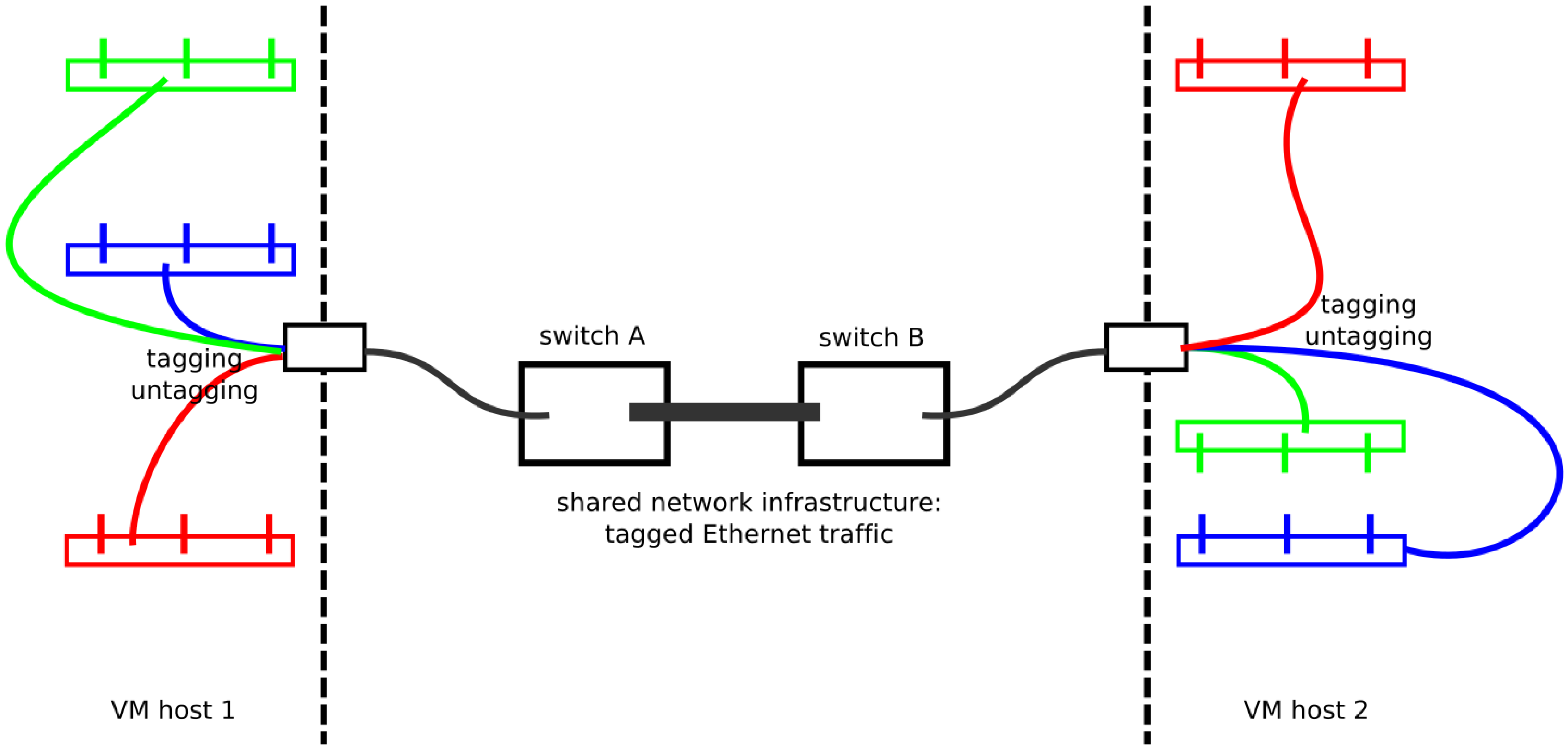
VLANs (3)

- Once a packet leaves a switch on a shared link, the packet is tagged.
- Once the packet reaches the other switch, it can be untagged and based on that tag be flowed to the corresponding VLAN.
- We have now actually ‘virtualized’ the wire.



VLANs (4)

- This tagging of packets is not limited to switches. (It is an open protocol).
- Our virtual machine host can also tag packets before they leave the machine through the physical network interface.
- Result: we can reduce the number of physical network interfaces. We tag packets based on the internal Ethernet bridge where they originated.


switch A
switch B
tagging
untagging
tagging
untagging
shared network infrastructure:
tagged Ethernet traffic
VM host 1
VM host 2

Virtualizing PCI devices

- Another way to reduce the number of physical network devices is to virtualize the actual PCI device.
- Supported through SR-IOV: single-root I/O virtualization.
- Compatible PCI devices can present multiple physical functions (PFs) and virtual functions (VFs).
- Example: assume a 4-port network card (NIC)
  - This could present itself as 4 1-port network cards to the hypervisor (PFs).
  - Each single-port network card can represent 128 virtual duplicates of itself (VFs).
- Such devices can be directly attached to virtual machines (device pass through); could lead to higher performance compared to device emulation and software Ethernet bridging.

Virtualized Networking (5)

- We have now seen how to interconnect these different bridges at different physical hosts.
- To do so, we can use VLANs, such that the VM hosts and networking hardware have knowledge about the different virtual networks.
- Note that all this physical network hardware will need to be managed.
  - VLANs can be created through ***manual*** configuration in the network hardware.
  - Can be automated; but tedious and error-phone (e.g. vendor updates the CLI interface).
  - What we need is a (much more) flexible means of managing networks and creating VLANs and Ethernet bridges in VM hosts on the fly.

SDN

- Idea: separate data plane and control plane.
  - Data plane: the hardware. Make the hardware 'stupid'.
  - Control plane: software controlling the hardware.
- No longer run the control software on a small embedded processor within the network hardware.
  - Instead, run this on a large array of commodity servers.
- SDN: Software-Defined Network

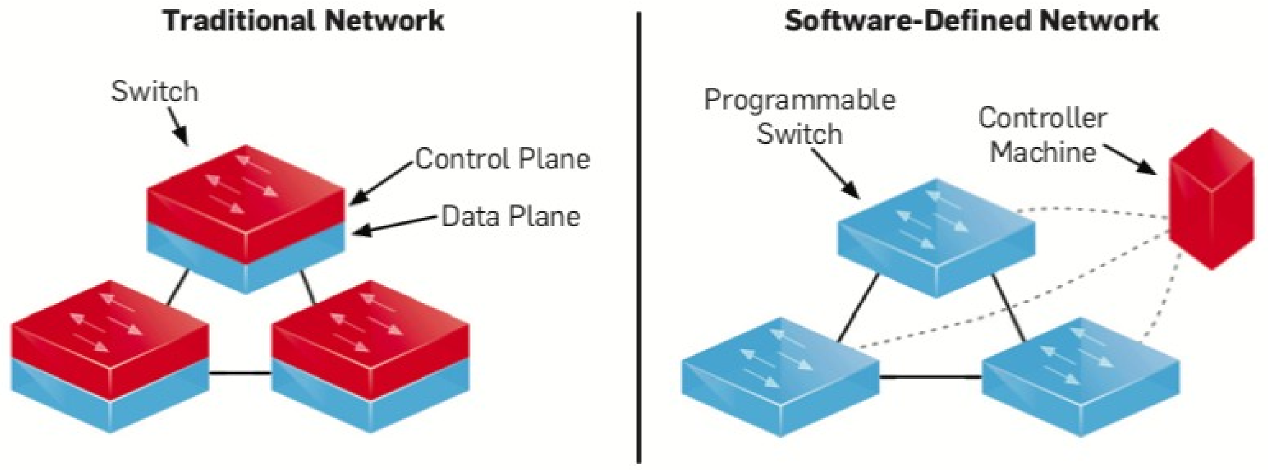


Image from: M. Casado et al. 2014. Abstractions for Software-Defined Networks. CACM 57, 10.

SDN (2)

- Move logic from hardware to software.
- Software computes and fills forwarding tables of hardware switches.
- Switches send encountered events to software and wait for orders.
  - E.g. “link up on new port”.
- Instead of a local controller communicating with neighbors, we now have centralized control (with a global view).

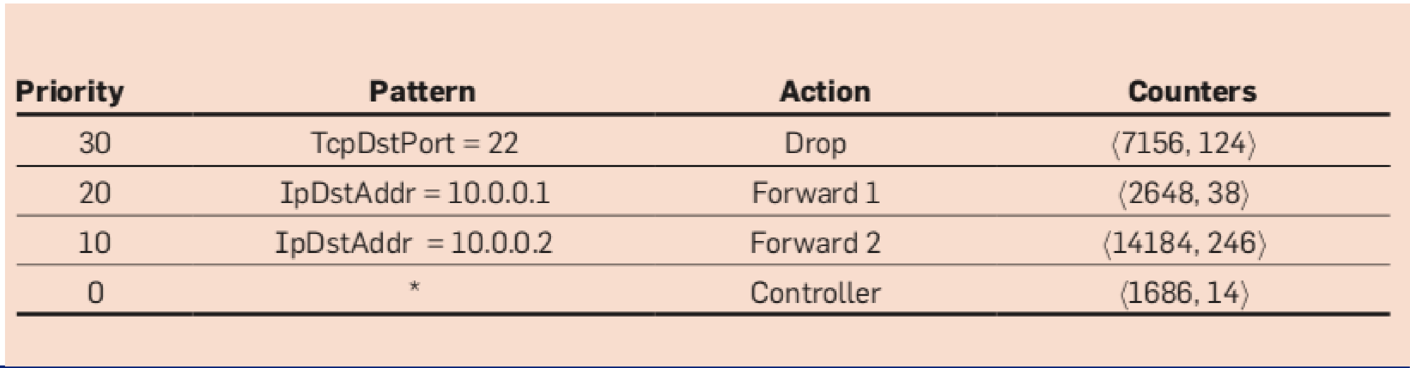
SDN advantages

- ***Virtualization***: virtualize network resources to be able to lease these to multiple tenants
- ***Performance***: change flows when congestion is detected; dynamic bandwidth allocation.
- ***QoS***: provide guaranteed bandwidth to specific users by manipulating network paths
- ***Energy efficiency***: in case of low utilization, consolidate VMs on less servers, update network paths, turn off hardware. Network elasticity.

J. Son and R. Buyya. 2018. A Taxonomy of Software-Defined (SDN)-Enabled Cloud Computing. ACM Comput. Surv. 51, 3 Article 59 (May 2018).

OpenFlow

- Network protocol to allow controller and agents (switches) to communicate.
- Data plane (switch): receive packet -> check flow table (match classifier) -> perform action.
- Controller updates flow table entries within the switches.
- Switch can send information to controller:
  - Changes of port status (port up, down)
  - If desired, send packets to controller that switch couldn't handle (didn't match anything in flow tables).



Priority	Pattern	Action	Counters
30	TcpDstPort = 22	Drop	⟨7156, 124⟩
20	IpDstAddr = 10.0.0.1	Forward 1	⟨2648, 38⟩
10	IpDstAddr = 10.0.0.2	Forward 2	⟨14184, 246⟩
0	*	Controller	⟨1686, 14⟩

Image from: M. Casado et al. 2014. Abstractions for Software-Defined Networks. CACM 57, 10.

NFV

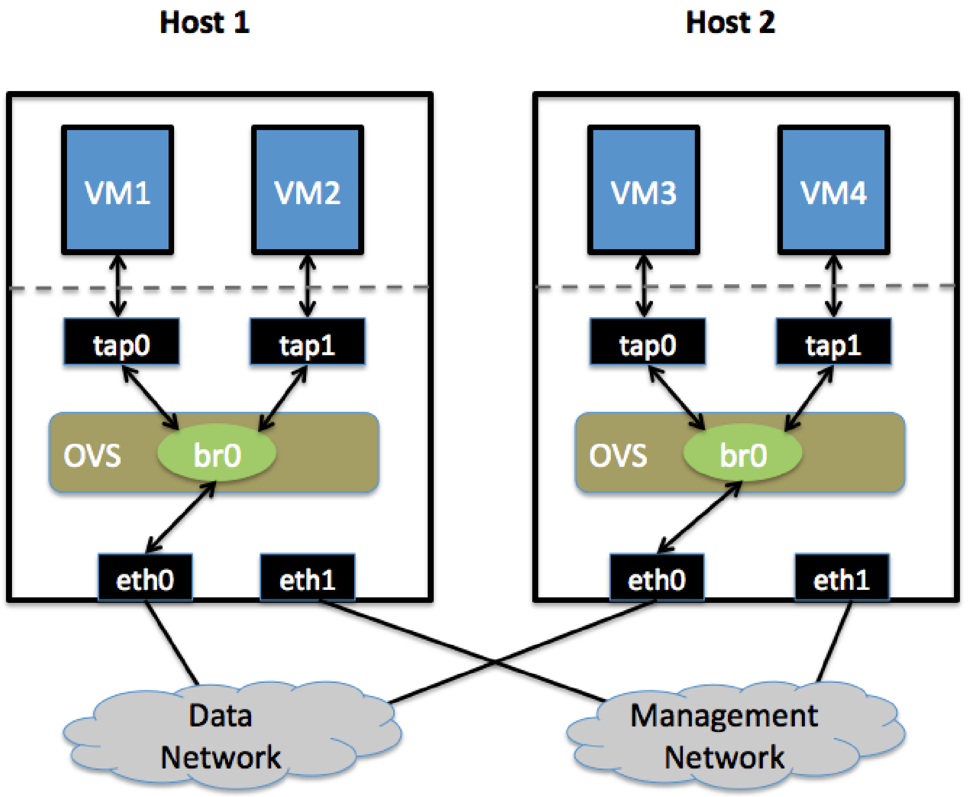
- NFV: Network Function Virtualization
- Implement network components in software and run in a virtual machine on commodity (server) hardware
  - Load balancers, firewalls, intrusion detection, wireless LAN controller, SIP components
- Note that device function is abstract here; contrast with SDN which moved device control to software.
- Benefits
  - Use modern software development techniques (CI)
  - Scale up/down; handle failures

NFV: what about hardware vendors?

- What are network hardware vendors doing?
- E.g. Cisco ported their network operating system usually running on routers/switch to x86.
  - And is also building x86 blades that can be integrated into router chassis.
- Juniper is selling switches that include x86 CPUs that support virtualization.

NFV example: Open vSwitch

- Open source multi-layer virtual switch
- Runs on hypervisor hosts to provide network
- Managed through OpenFlow
- Can handle multiple VLANs on single switch
- Interconnect with switches in other hosts through VLAN/VXLAN.



Source: http://docs.openvswitch.org/en/latest/howto/vlan/

VXLAN

- One of the things supported by Open vSwitch (and newer generation hardware switches) is VXLAN
- VXLAN extends the number of supported virtual networks to 16 million
  - This is achieved by encapsulated the Ethernet packets in UDP frames.
  - These UDP frames can also be processed by switches that don't support VXLAN.
  - So we can connect hypervisor hosts running Open vSwitch using VXLANs using switching hardware that does not support VXLAN.

Google B4 Case Study

- Google WAN connects Google's different data centers.
- Very high bandwidth requirements.
- High-end equipment required, that is usually ...
  - designed for Internet backbones.
  - used for many different customers at many different locations; large routing tables.
  - used in cases where there is no idea and influence on the traffic patterns; deep packet buffers.
  - engineered for a generic case.
  - expensive; not many customers for this kind of hardware and need to recoup development cost.

Google B4 Case Study (2)

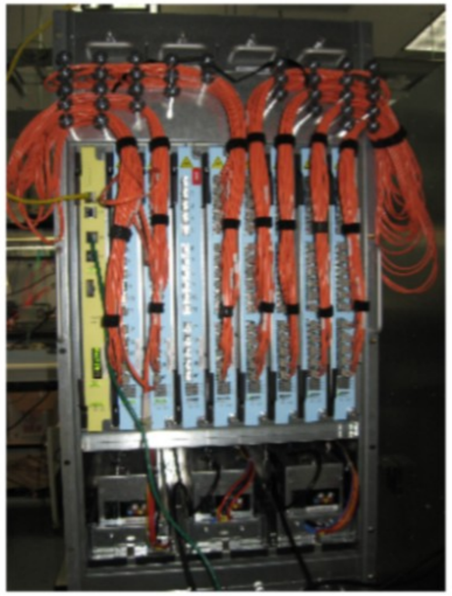
- Differences for Google:
  - Only has approx. 12 data centers.
  - All traffic belongs to Google (single customer).
  - Have knowledge and can have influence on applications running on the platform. Bandwidth control.
- Feasible to control everything from centralized controller.

Google B4 Case Study (3)

- Interest in using SDN.
- Google has enough commodity hardware to host the controller.
- Flexibility; experiment with new protocols (no need to wait for vendors).
- Wanted to do traffic shaping/engineering.
- Wrote their own OpenFlow compatible controller software.

Google B4 Case Study (4)

- Given Google's scale it was cost effective to built their own switching hardware.
- OpenFlow: only data plane, control logic is in software.
- Bought silicon switch chips and built their own hardware.



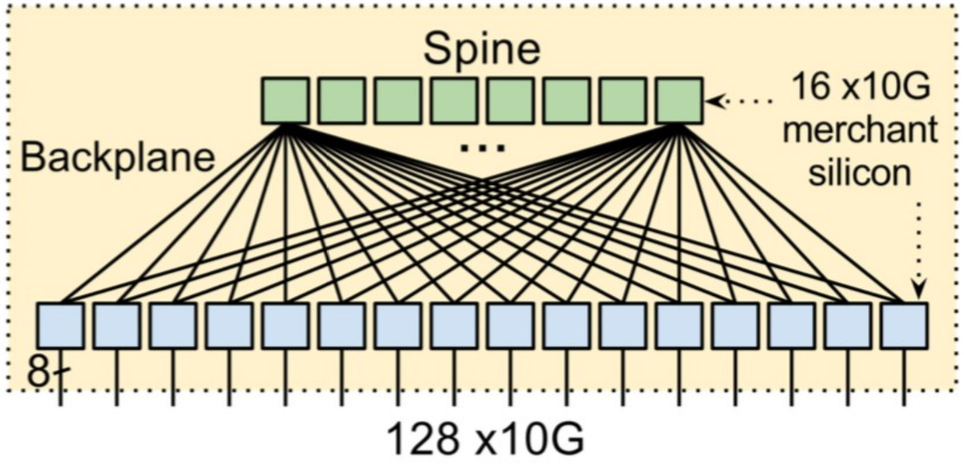
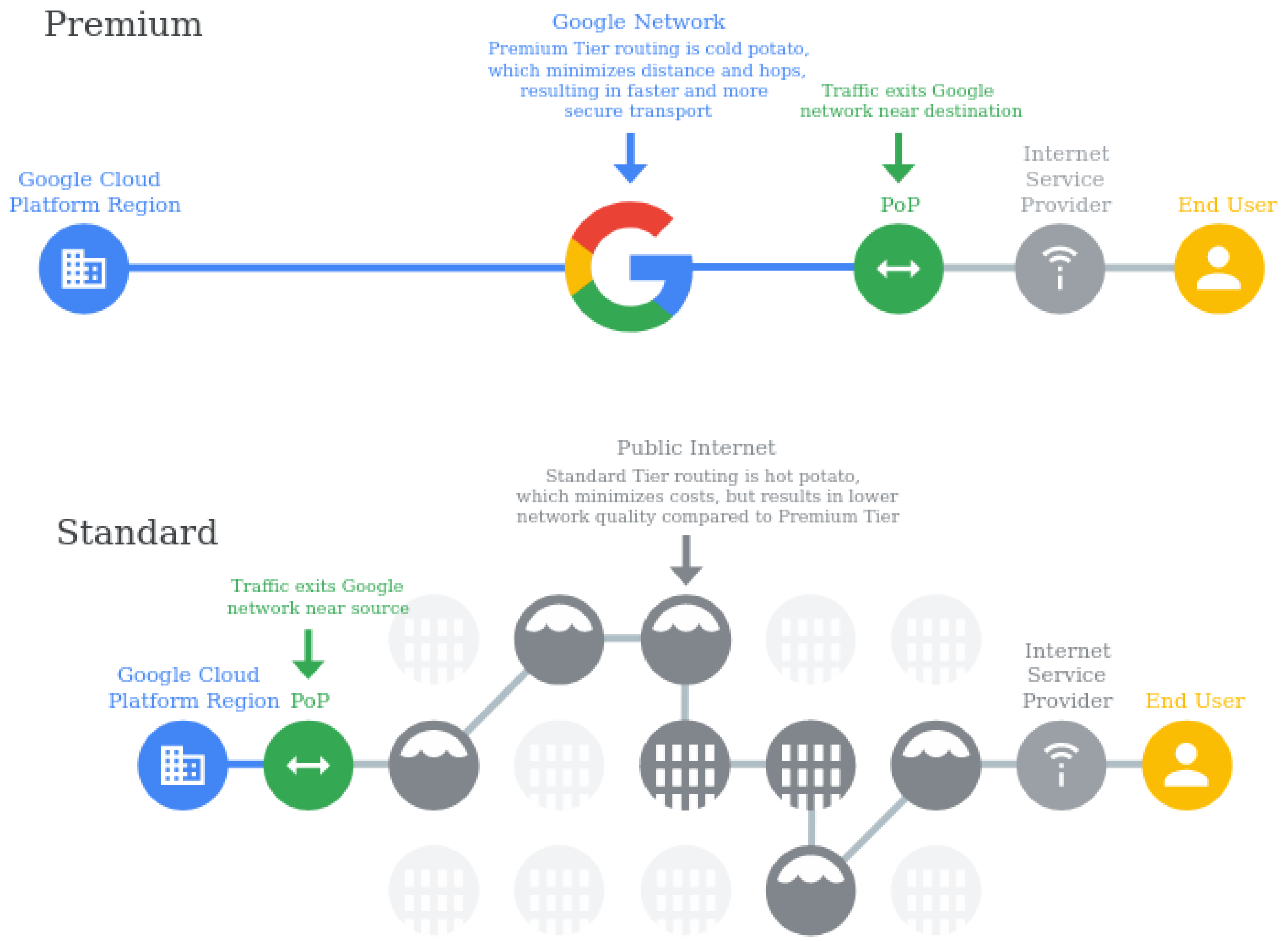


Figure 3: A custom-built switch and its topology.

Image: S. Jain et al. 2013. B4: Experience with a Globally-Deployed Software Defined WAN. SIGCOMM'13.

Network Traffic Billing

- Making everything configurable also gives the customers a choice in kind of network access.
- For instance, Google offers two network service tiers:
  - *Premium Tier*: using Google's backbone
  - *Standard Tier*: routed over public Internet.



Premium
Google Network
Premium Tier routing is cold potato, which minimizes distance and hops, resulting in faster and more secure transport
Traffic exits Google network near destination
Google Cloud Platform Region
PoP
Internet Service Provider
End User
Standard
Public Internet
Standard Tier routing is hot potato, which minimizes costs, but results in lower network quality compared to Premium Tier
Traffic exits Google network near source
Google Cloud Platform Region
PoP
Internet Service Provider
End User

Images: https://cloud.google.com/network-tiers/docs/overview

Further networking enhancements

- Network performance in Cloud environments (which are distributed systems) is critical!
- Further enhancements are being researched, e.g.:
  - Virtualized NICs that enforce performance isolation and lead to predictable performance. This to counter negative influences on network performance by ‘neighboring’ VMs.

    P. Kumar, et.al. 2019. PicNIC: predictable virtualized NIC. In Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19). ACM. 351–366.
  - Offload host networking to hardware (FPGA) to relieve CPU cores in the VMs.

    D. Firestone, et.al. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In: NSDI18. USENIX association.

Putting it all together

- We have now considered virtualization of machines, storage and networking.
- How do we manage all of this?
  - And have tenants create resources on the fly through a control panel.
  - And support APIs to create such resources.
- An all encompassing software system is necessary: the Cloud OS or Cloud manager.

Putting it all together (2)

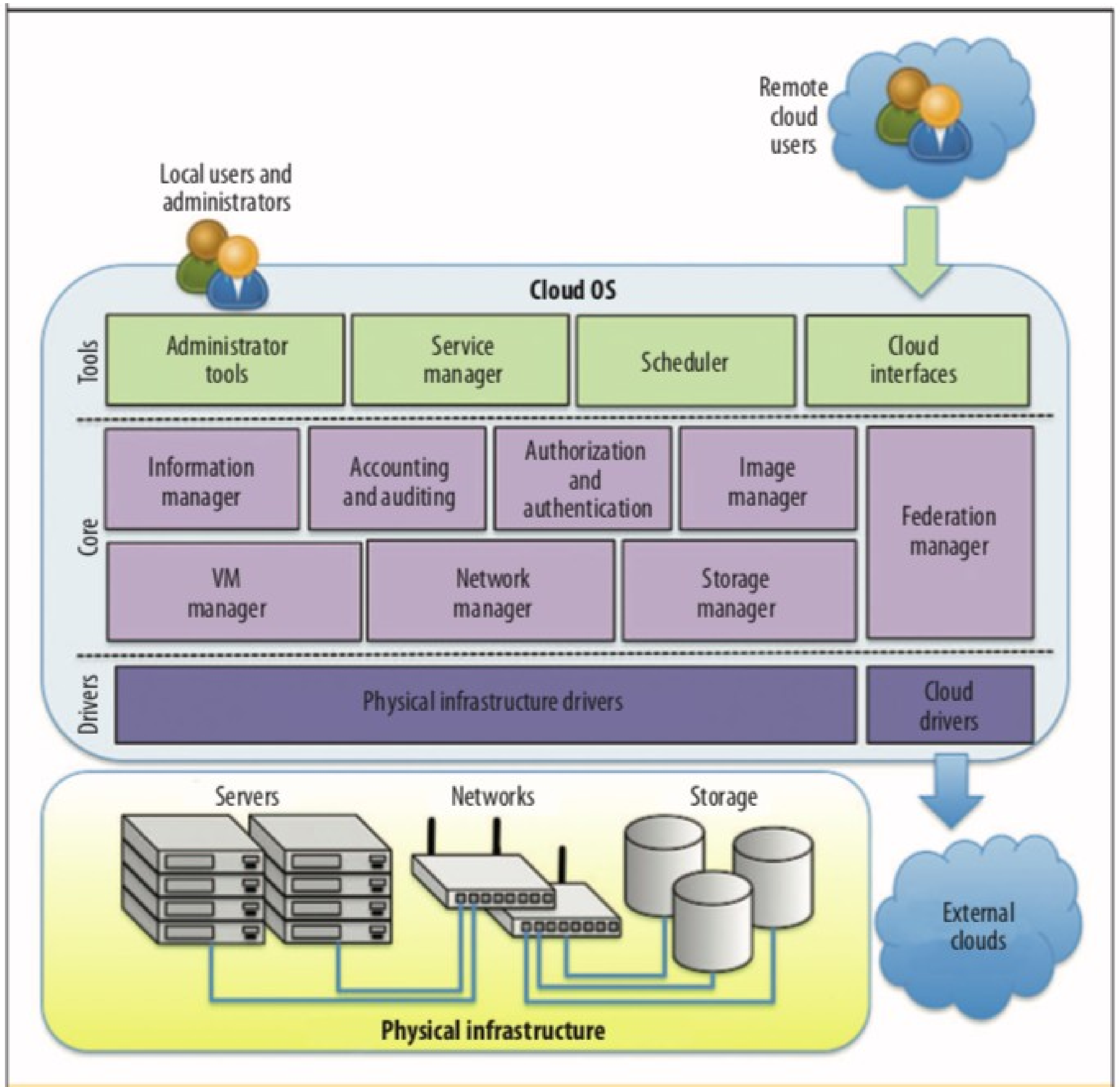


Figure 1. The cloud OS, the main component of an IaaS cloud architecture, is organized in three layers: drivers, core components, and high-level tools.

Image: R. Moreno-Vozmediano et al. 2012. IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures. Computer. IEEE Computer Society.

Eucalypus

- One of the first open source cloud management systems; developed at University of California.
- NC: Node Controller
- CC: Cluster Controller
- Walrus: Storage Controller
- CLC: Cloud Controller

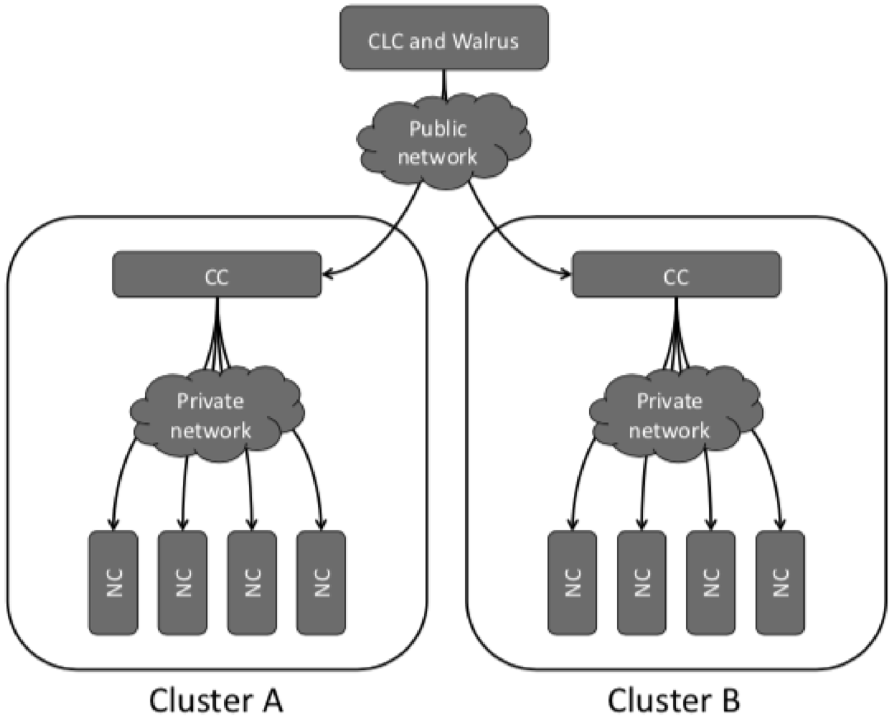
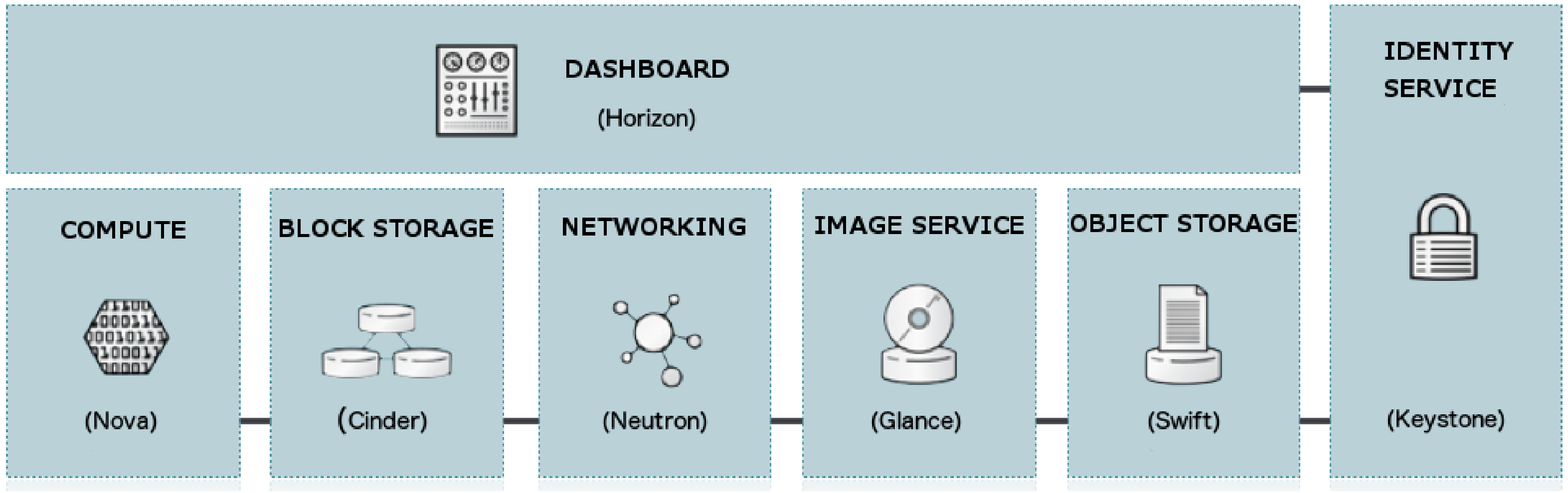


Figure 1. EUCALYPTUS *employs a hierarchical design to reflect underlying resource topologies.*

Image: D. Nurmi et al. 2009. The Eucalyptus Open-source Cloud-computing System. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid.

OpenStack

- One of the leading open source cloud platforms.
- Manages several large deployments



Source: https://docs.openstack.org/security-guide/_images/marketecture-diagram.png

OpenStack: Example Design

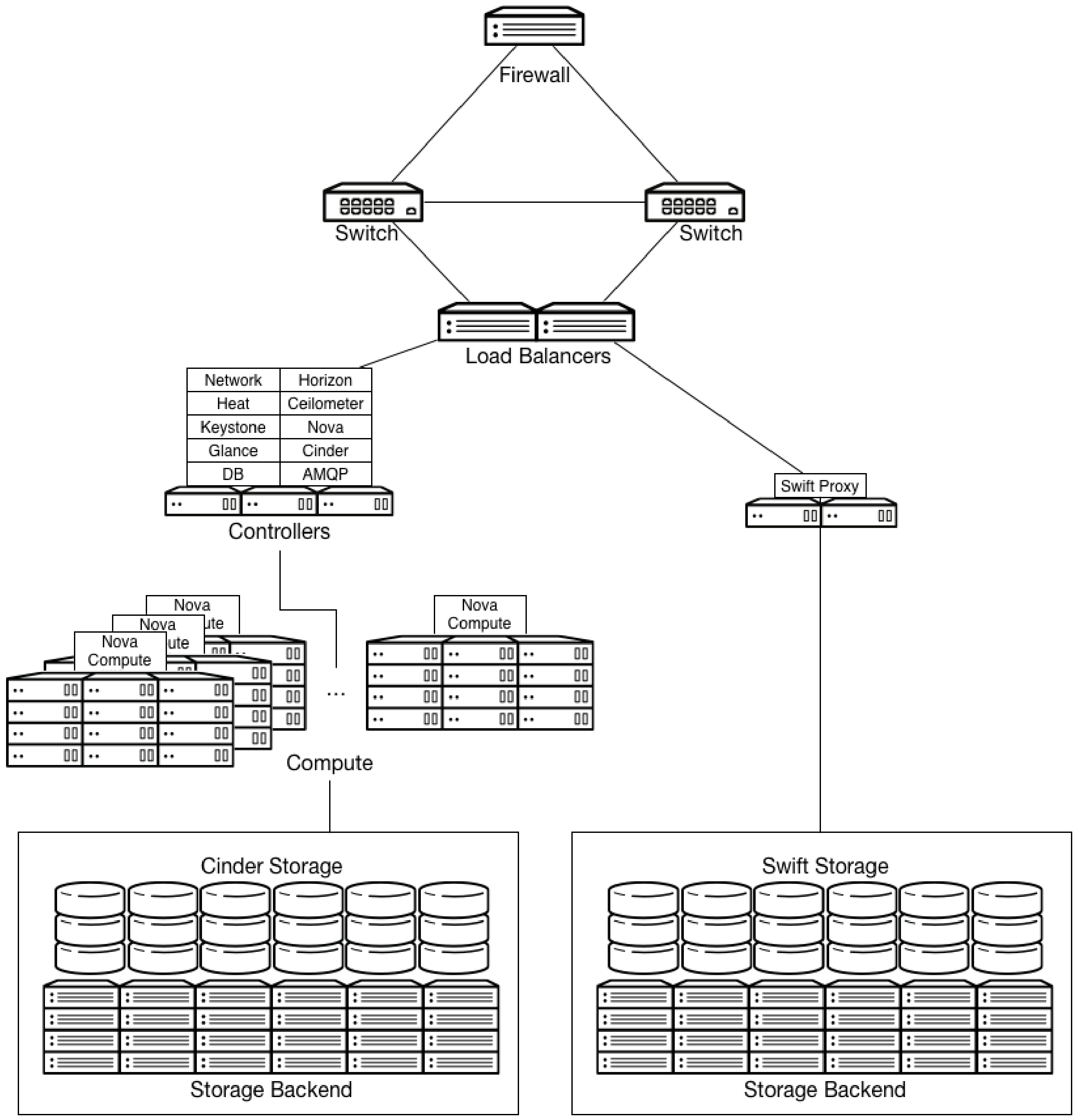


Image source: https://docs.openstack.org/arch-design/use-cases/use-case-general-compute.html

Putting it all together: Containers

- Eucalyptus and OpenStack are “classic” approaches that initially focus on managing Virtual Machines.
- However, containers have seen a very sharp increase in use in the last few years.
- As we have seen, we can use Docker to create containers.
  - But just like we need something to manage virtual machines, we need the same for containers.
  - Recall that Docker only creates and manages containers locally, on a single machine.

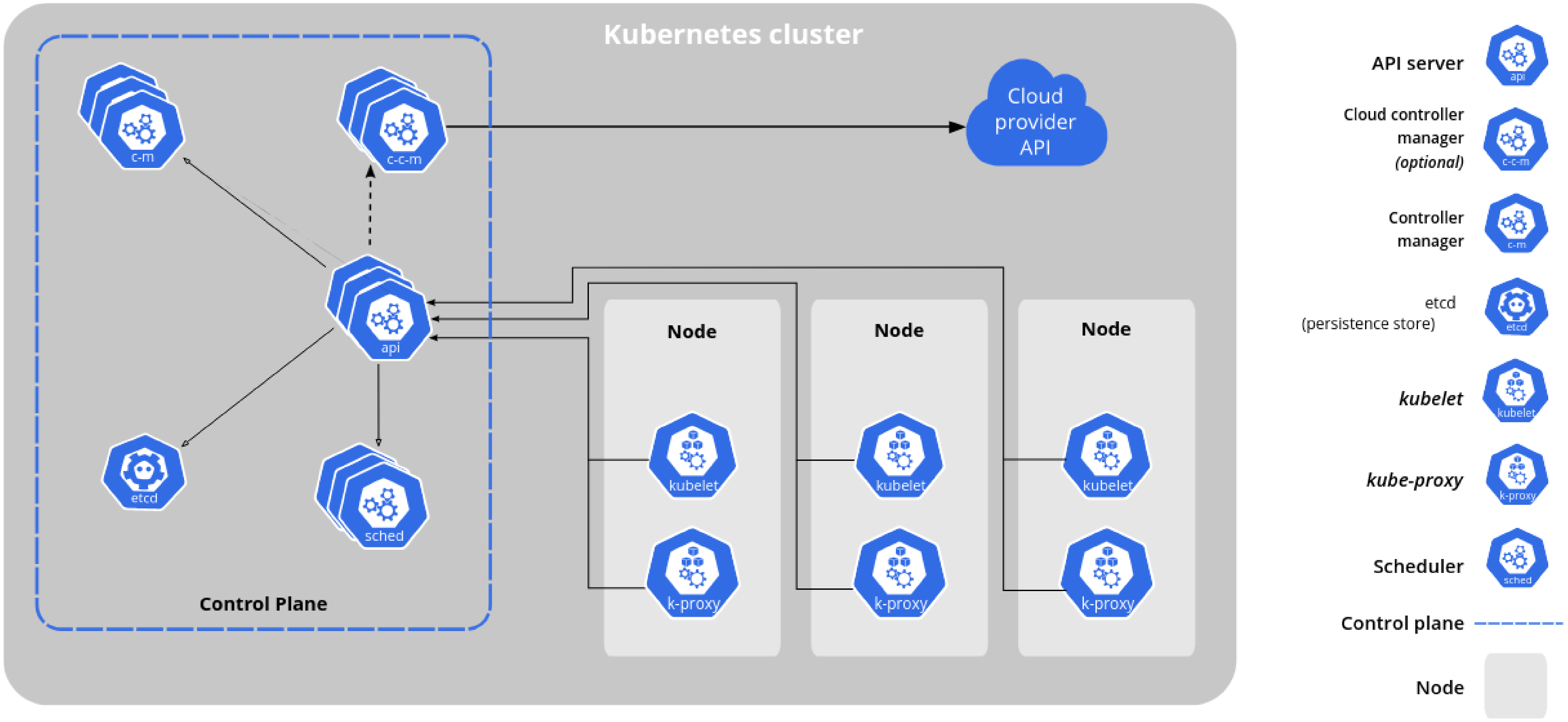
Clusters of container hosts

- In order to manage clusters of container hosts, we need additional software.
  - Container orchestration.
- One of the first projects to accomplish this was “Docker Swarm”.
  - Manager & worker nodes.
  - The deployment entity is a service, which consists of tasks. A task is a wrapper around a container.
  - Managers make sure tasks are started on worker nodes.

Clusters of container hosts (2)

- However, lately everybody is talking about Kubernetes.
  - Initially developed by Google, open source.
- More extensive than Docker swarm
  - Workloads & Pods, Pods wrap containers.
  - Can handle load balancing for exposed ports to groups of pods.
  - Resource management: fit containers onto nodes using bin packing.
  - Supports different container runtimes: Docker, containerd, CRI-O.
  - Docker support to be deprecated.

Kubernetes Overview



Source: https://kubernetes.io/docs/concepts/overview/components/

Kubernetes derivatives

- Google Kubernetes Engine (GKE)
- Amazon Elastic Kubernetes Service (EKS)
- Red Hat OpenShift
  - Built on Red Hat CoreOS

Marketing trend

- Software Defined Computing / Software Defined Infrastructure / Software Defined Data Center
  - Software Defined Everything (SDx)
- All resources are virtualized:
  - compute (SDC)
  - storage (SDS)
  - network (SDN)
- All infrastructure is offered as a service; tenants can build their own data center.

Next steps ...

- We now have a cloud as in IaaS architecture
  - These are the building blocks for PaaS and SaaS
- How to put this to use?
- How to create elastic applications?

Universiteit
Leiden