# Virtualization Technologies – Revised Version

## (ENCS 691K – Chapter 3)

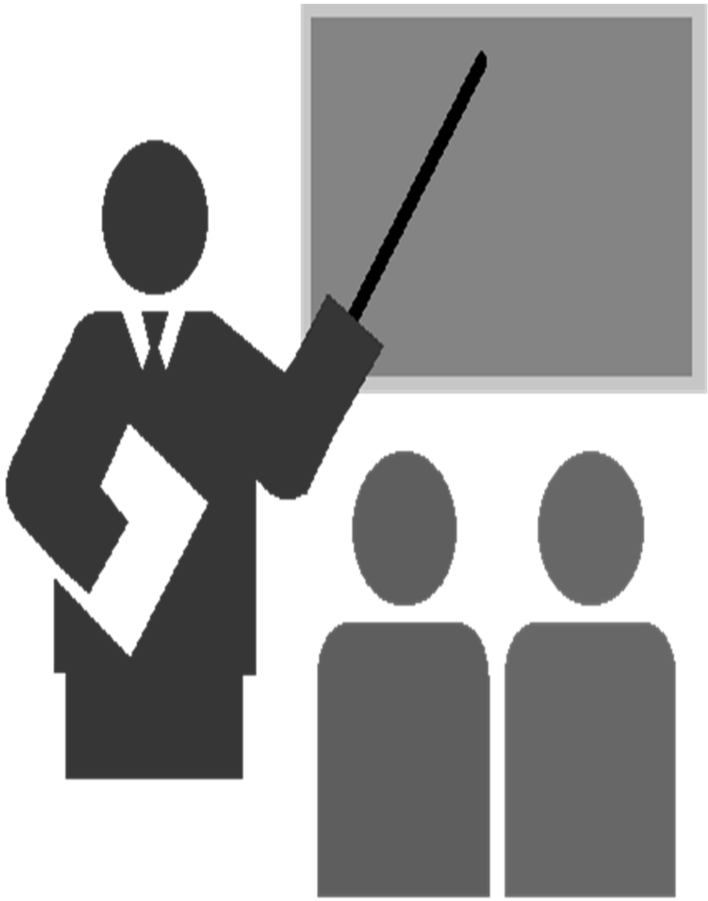
**Note: The changes made to the original version are strictly limited to the references.**

**Roch Glitho, PhD**

**Associate Professor and Canada Research Chair**

**My URL - http://users.encs.concordia.ca/~glitho/**

# The Key Technologies on Which Cloud Computing Relies



- Web Services
- Virtualization

# References


1. . M. Pearce et al., Virtualization: Issues, Security, Threats, and Solutions, ACM Computing Survey,  February 2013

2. . A. Khan et al., Network Vitrtualization: A Hypervisor for the Internet?, IEEE Communications Magazine, January 2012

3. P. Barham et al., XEN and the Art of Virtualization, SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, Pages 164-177

4. . N.M Chowdhury and r. Boutaba, Network Virtualization: State of the Art and Research Challenges, IEEE Communications Magazine, July 2009

5.. J. Carapinha et al., Network Virtualization – A View from the Bottom, VISA '09 Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, Pages 73-80

# References (Network Virtualization)

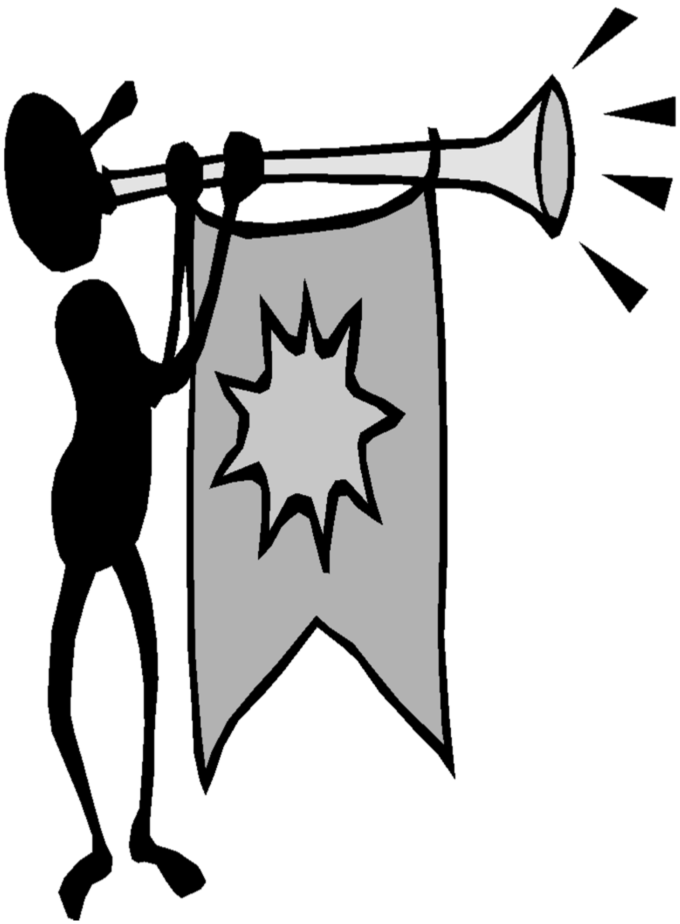
6. G. Schaffrat et al., Network Virtualization Architecture: Proposal and Initial Prototype, Proceeding VISA '09 Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures, Pages 63-72

7. J. Kurose and K. Ross, Computer Networking: A Top Down Approach, Pearson, 6th Edition, 2013

8. Venkateswanan, Virtual Private Networks, IEEE Potentials, Issue 20, no1


# Virtualization

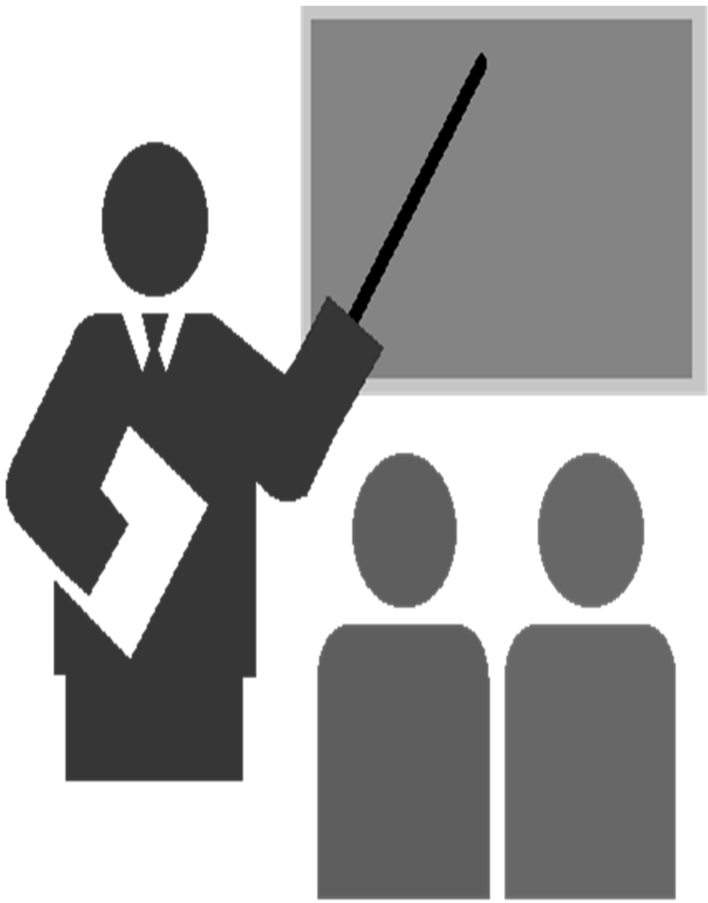
# Outline



1. Systems virtualization

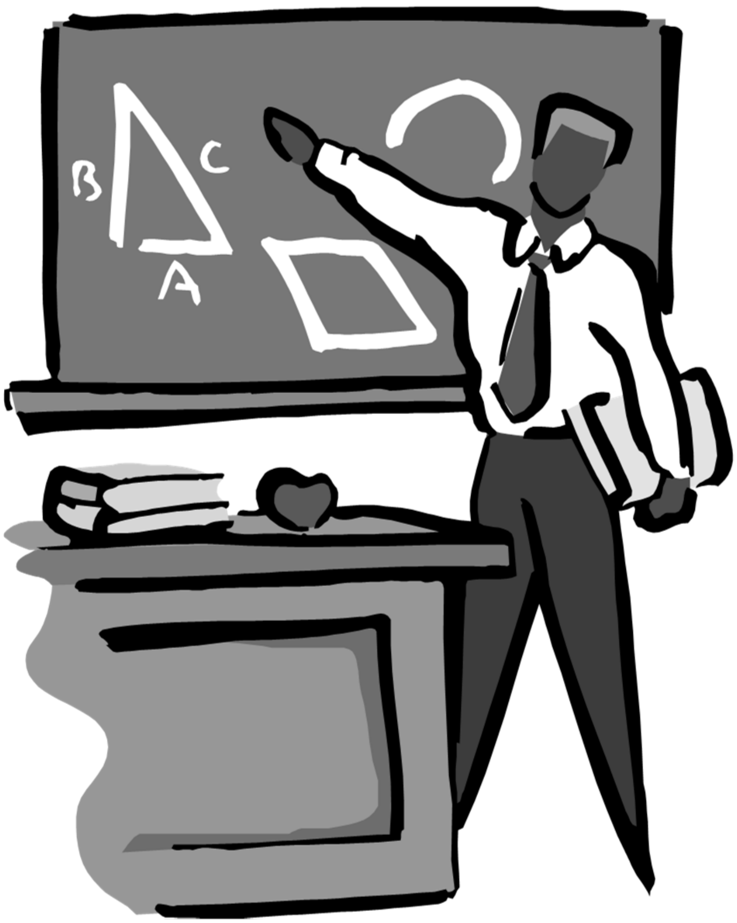
2. Network virtualization

# On Systems Virtualization



- Key concepts
- Type I (bare metal) vs. Type 2 (hosted)
- Full virtualization vs. para-virtualization

# Basic concepts




1. On operating systems

2. Virtual machine, virtual machine monitor/hypervisor

4. Examples of benefits

# Operating systems

## Some of the motivations

- Only one single tread of CPU can run at a time on any single core consumer machine
- Machine language is tedious

# Operating systems

**Operating systems bring a level of abstraction on which multiple processes can run at a time – Deal among other things with:**

- Multiplexing
- Hardware management issues

However only one operating system can run on a bare single core consumer machine

# virtual machines and hypervisors

- Systems virtualization dates back to the 60s

- IBM experimentation with “time sharing systems”
  - Need for virtual machines to test how applications / users can time share a real machine

# virtual machines and hypervisors

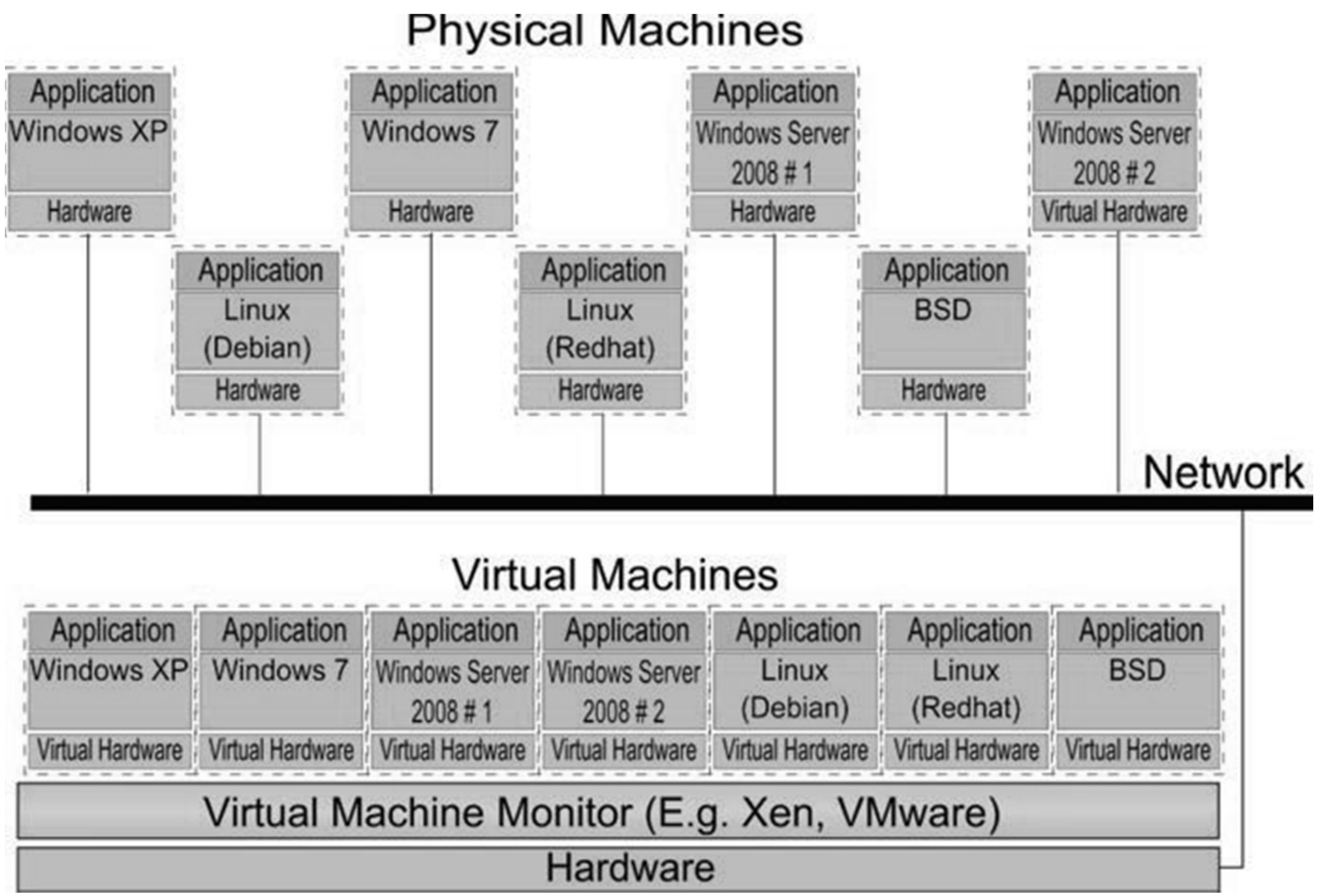
Virtual machine (VM) (sometimes called virtual hardware)

- Software that provides same inputs / outputs and behaviour expected from hardware (i.e. real machine) and that supports operations such as:
  - Create
  - Delete
  - Migrate
  - Increase resources

Virtual machine monitor (also called hypervisor)

- Software environment that enables operations on virtual machines (e.g. XEN, VMWare) and ensures isolation

# virtual machines, hypervisors



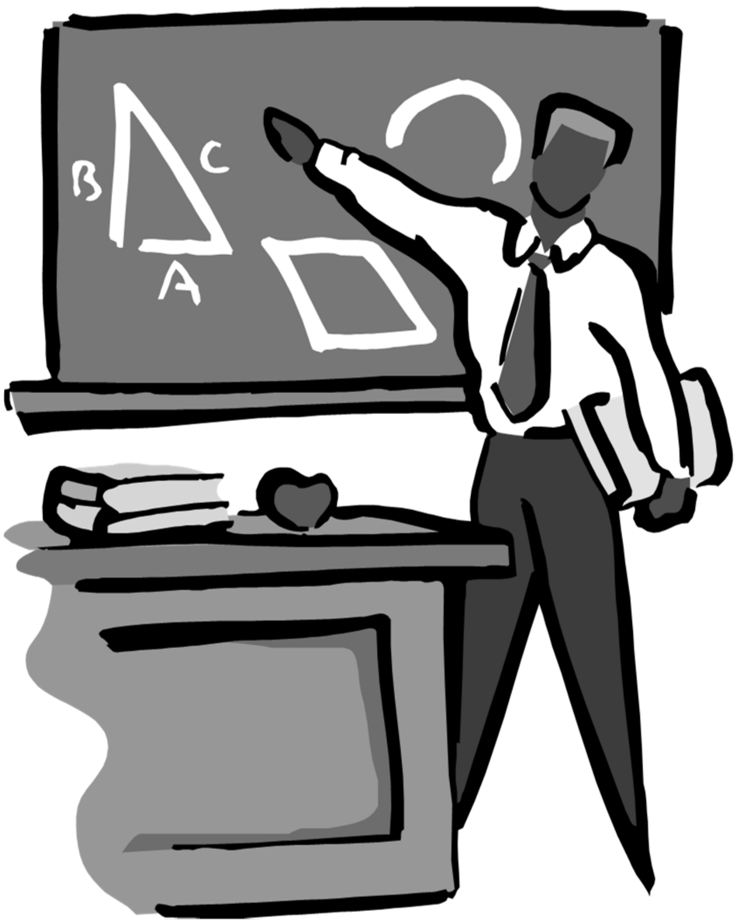

From reference [1] – Note: There is a small error in the figure

# Examples of Benefits

All benefits are due to the possibility to manipulate virtual machine (e.g. create, delete, increase resources, migrate), e.g.

- Co-existence of operating systems
- Operating systems research
- Software testing and run-time debugging
- Optimization of hardware utilization
- Job migration

# Advanced concepts




1. Bare metal vs. hosted hypervisor

2. Full virtualization vs. Para-virtualization

# Type I vs Type II Hypervisor

Types of hypervisor

- Type I – bare metal
  - Installed on bare hardware
  - Examples
    - Citrix XEN server
    - VMWARE ESX/ESXI

# Type I vs Type II Hypervisor

Types of hypervisor

- Type 2 – hosted
  - Runs on top of host operating system
  - Examples:
    - VMWare workstation
    - VirtualBox

# Type I vs Type II Hypervisor

Type I - Bare metal

- Hypervisor installed on bare hardware
  - Advantages (compared to type II)
    - Performance (No additional software layer to go through)
    - Security (No possible attack through host operating system)
  - Drawbacks (compared to type II)
    - Host operating system needs to be “ported” on top of hypervisor
    - Complexity depends on the type of virtualization (Full virtualization vs. para-virtualization)

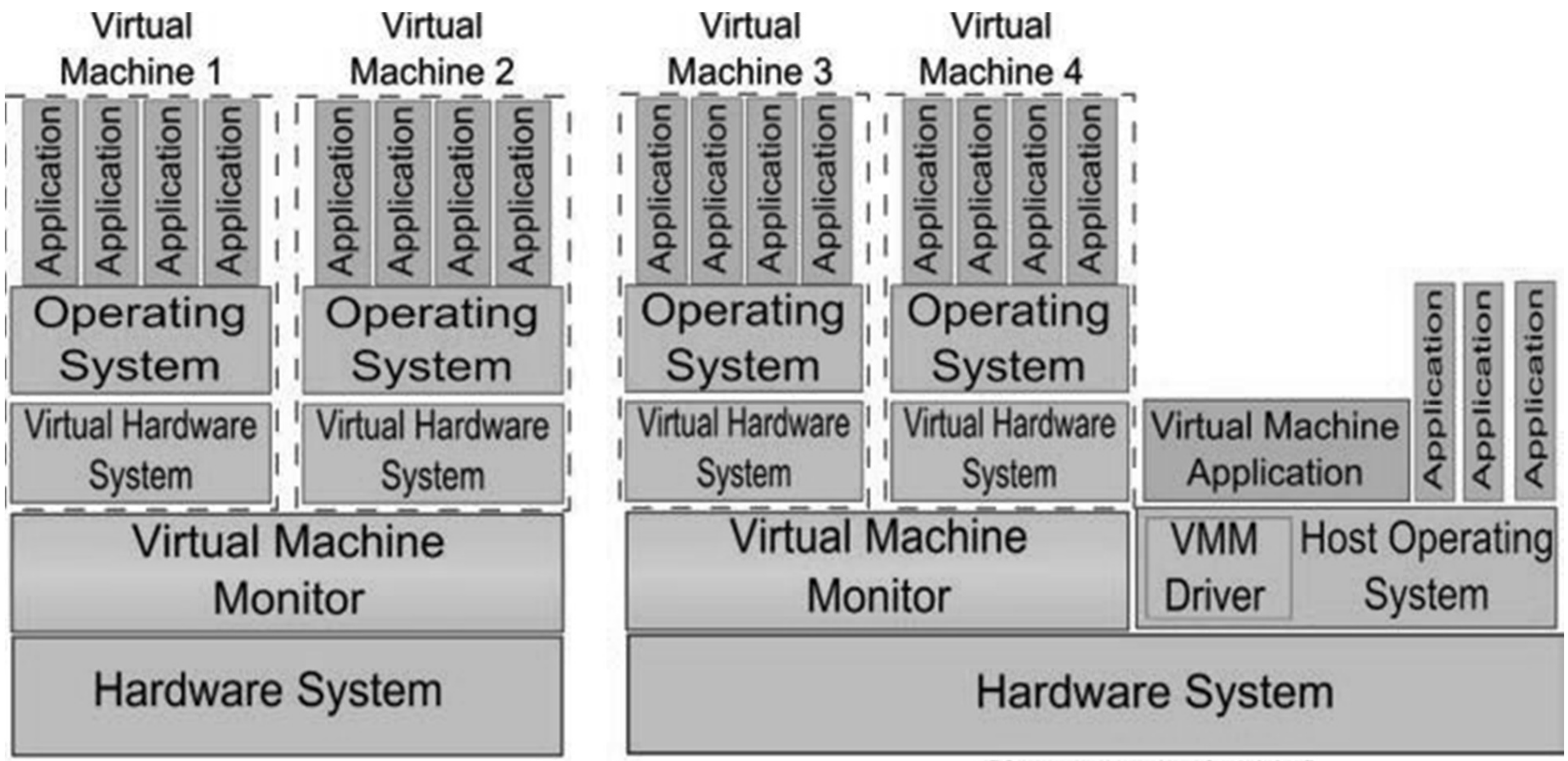
# Type I vs Type II Hypervisor

Type II - Hosted

- Hypervisor installed on top of host operating system
  - Drawbacks (compared to type I)
    - Performance (need to go through host operating system)
    - Security (i.e. Possibility to attack through host operating system)
  - Advantages (compared to type I)
    - Host operating system is re-used as it is (No need to port it)
    - No change required to applications running on top of host operating system

# Type I vs Type II Hypervisor (Summary)

Types of hypervisor/virtual machine monitor (From ref. 1)



[Sugerman et al. 2001]

# Full virtualization vs. Para-virtualization

## More on operating systems fundamentals

- Privileged vs. non privileged instruction
  - Privileged
    - If called in user mode, the CPU needs to trap it and switch control to supervisory software (e.g. hypervisor) for its execution

# Full virtualization vs. Para-virtualization

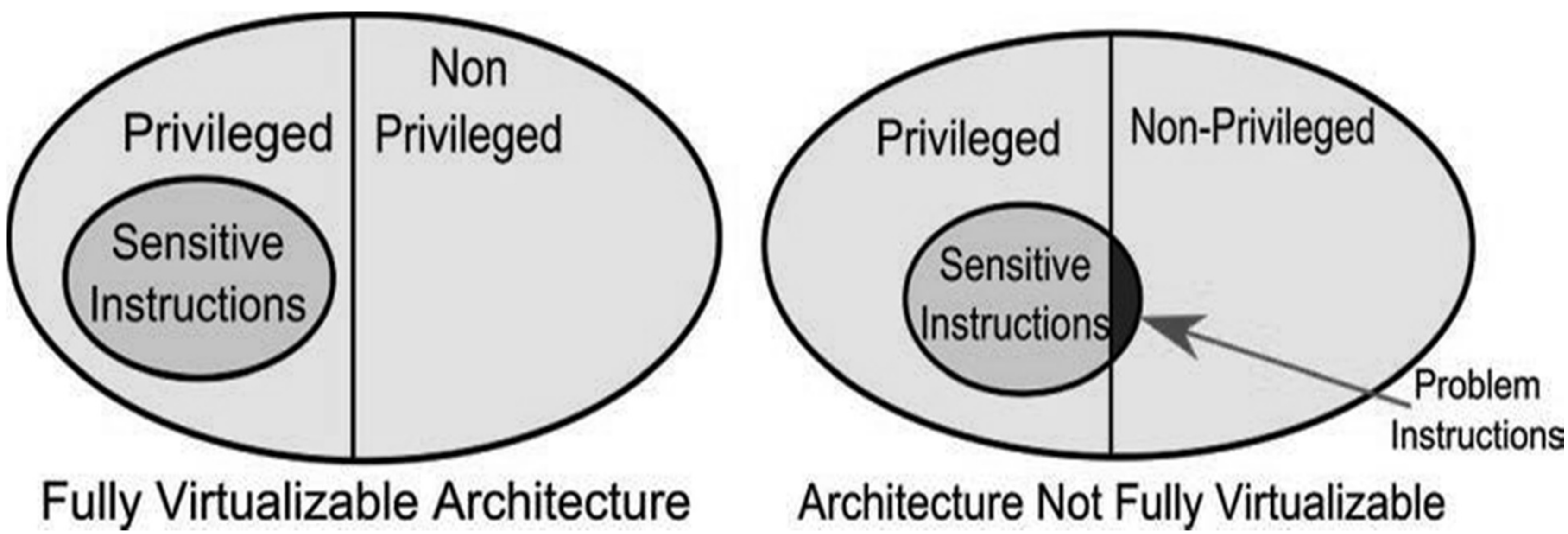
## More on operating systems fundamentals

- Sensitive vs. non sensitive instruction
  - Sensitive
    - Has the capacity to interfere with supervisor software functioning (e.g. Hypervisor)
      - Write hypervisor memory vs. read hypervisor memory

# Full virtualization vs. Para-virtualization

## Could all CPU architectures be fully virtualized ?

- Could be fully virtualized only if the set of sensitive instructions is a subset of the privileged instructions




**From reference [1]**

# Full virtualization vs. Para-virtualization

## Could all CPU architectures be fully virtualized ?

- The case of Intel x86 CPU architectures
  - Cannot be fully virtualized
    - "Certain instructions must be handled by the VMM for correct virtualization, but these with insufficient privilege fail silently rather than causing a convenient trap" – Reference [3]

# Full virtualization vs. Para-virtualization

## Definitions

### Full virtualization

- Hypervisor enables virtual machines identical to real machine
  - Problematic for architectures such as Intel x86

# Full virtualization vs. Para-virtualization

## Definitions

### Para-virtualization

- Hypervisor enables virtual machine that are similar but not identical to real machine
  - A solution to the problem of CPU architectures that cannot be virtualized
    - Prevents user programs from executing sensitive instructions
  - Note:
    - Para-virtualization is not the only solution to the problem

# Full virtualization vs. Para-virtualization

## Full virtualization

- Advantages
  - Possibility to host guest operating systems with no change since virtual machines are identical to real machines
- Disadvantages
  - Not always feasible (e.g. Intel x86)
    - There are work around (e.g. binary translation)
  - Some guest operating systems might need to see both virtual resources and real resources for real time applications

# Full virtualization vs. Para-virtualization

**Para - virtualization**

- Advantages
  - Feasible for all CPU architectures
  - Performance – Compared to:
    - Full virtualization
    - Other approaches to architectures that could not be virtualized (e.g. binary translation)
- Disadvantages
  - Need to modify guest operating systems

# Full virtualization vs. Para-virtualization

**Para - virtualization**

- Alternatives to para-virtualization
  - Binary translation (e.g. VMWare ESX server)
    - Leads to full virtualization
    - No need to re-write "statically" guest operating systems
      - i.e. guest OS can be installed without change
    - Interpretation of guest code (OS + application)
      - "Rewrites" dynamically guest code and insert traps when necessary

# Full virtualization vs. Para-virtualization

**Para - virtualization**

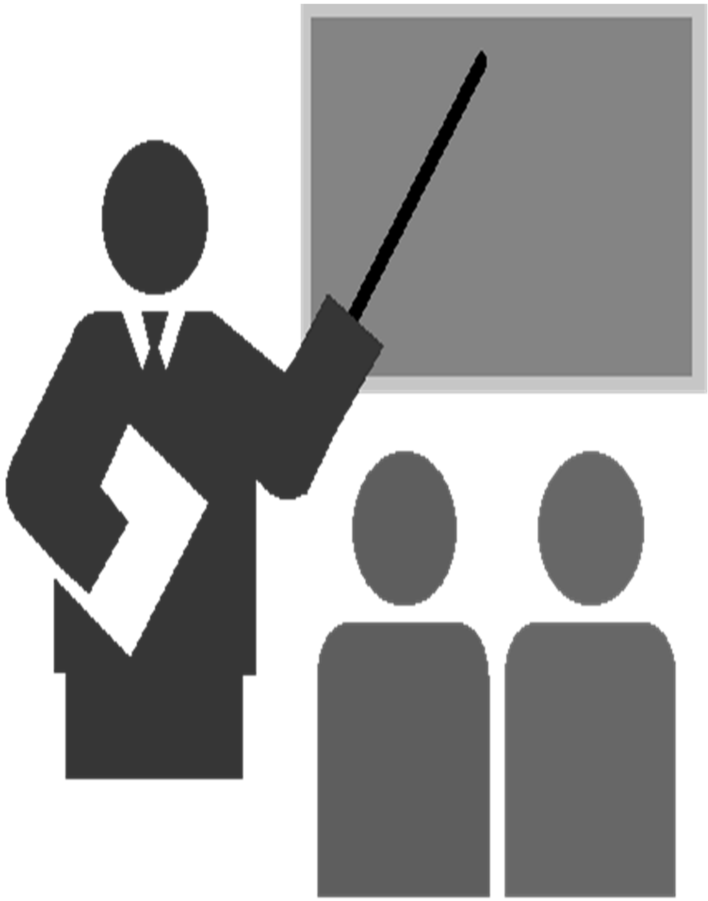
- Alternatives to para-virtualization
  - Binary translation
    - Disadvantages / penalties
      - Performance
      - However, optimization is possible, e.g.
        - » Adaptive translation (i.e. optimize the code being translated)

# Full virtualization vs. Para-virtualization

## Para – virtualization

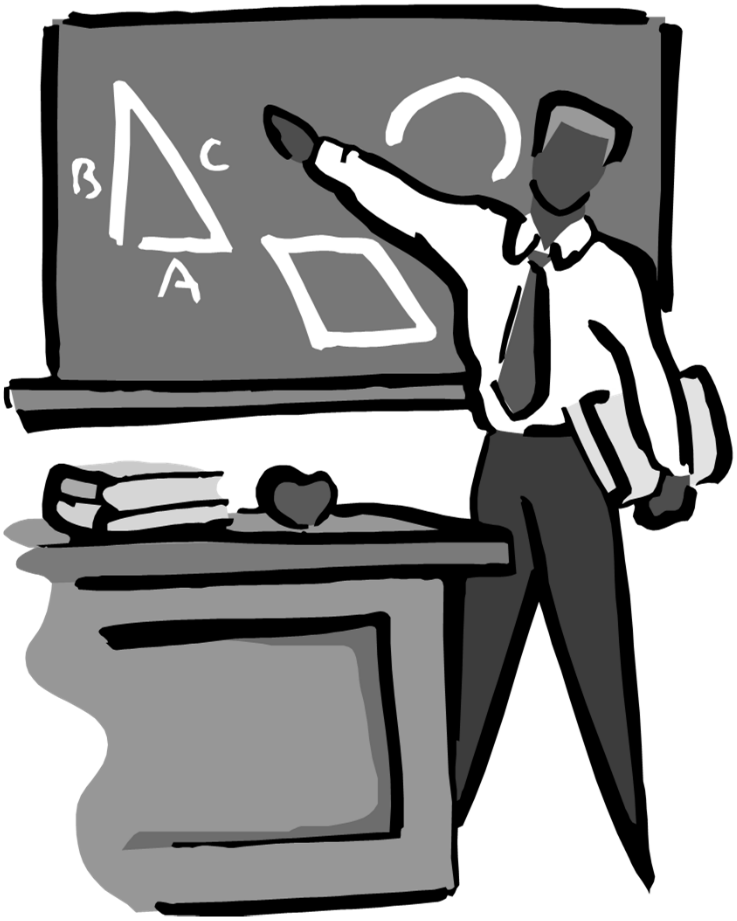
- A detailed case study on para-virtualization
  - XEN (Reference 3)

# On Network Virtualization



- Motivations and basic components
- Prior to network virtualization
- A case study

# On Network virtualization




1. Motivations

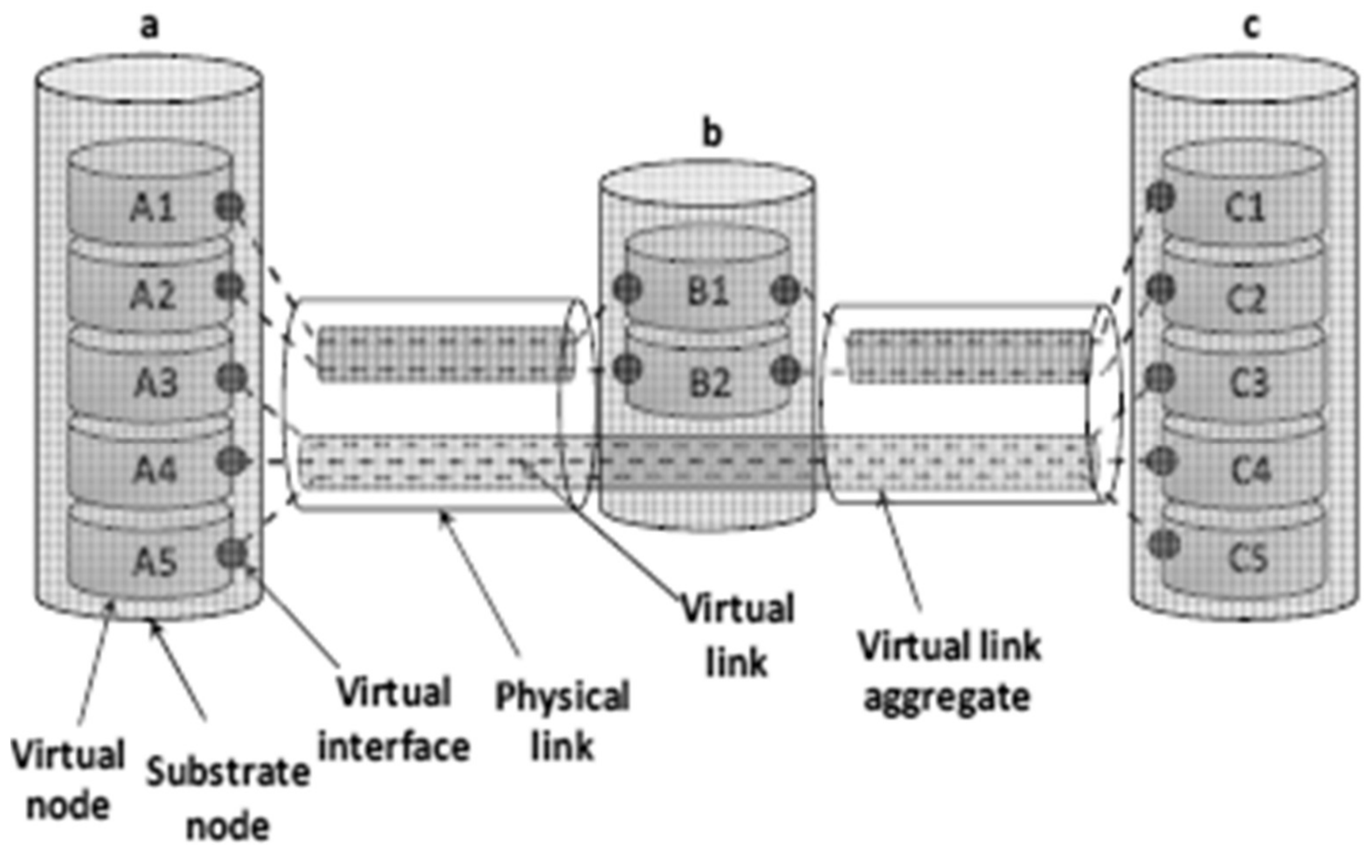
2. Basic components

# Motivations

Bring the benefits of systems virtualization to the networking world, e.g.

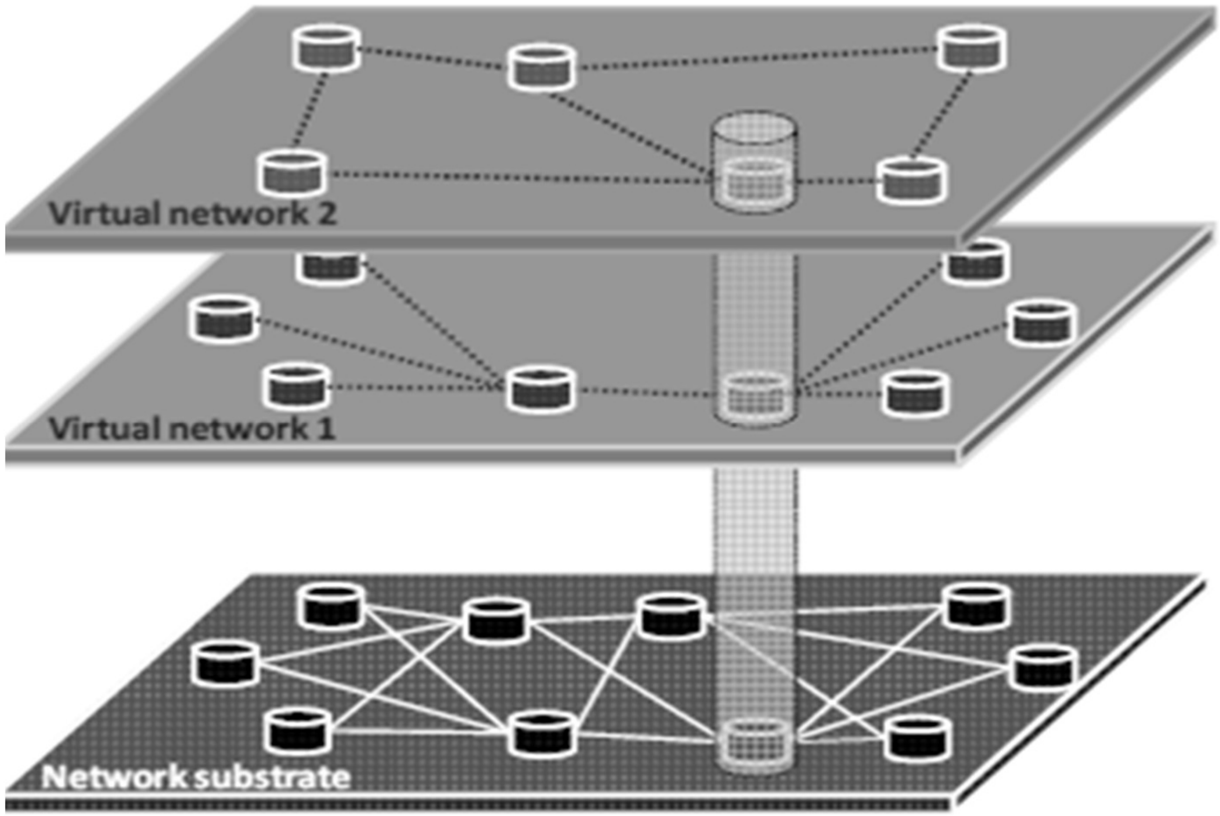
- Co-existence of virtual networks on top of a same real network
  - Note: Virtual Private Networks (VPNs) do not rely on virtualization and have several limitations
    - Different technologies and protocol stacks cannot be used for instance
- Networking research
- Optimization of networking resources utilization
  - Nodes
  - Links

# Basic components



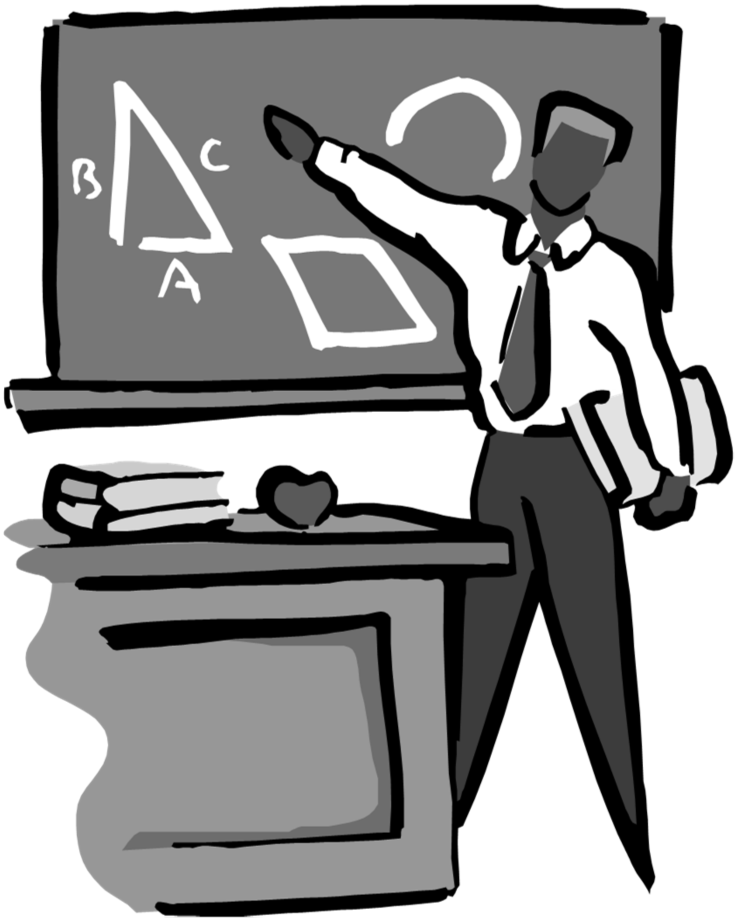

From reference 5

# Basic components




From reference 5

# On Network virtualization




1. Prior to network virtualization

2. A Case study

# Prior to Network Virtualization

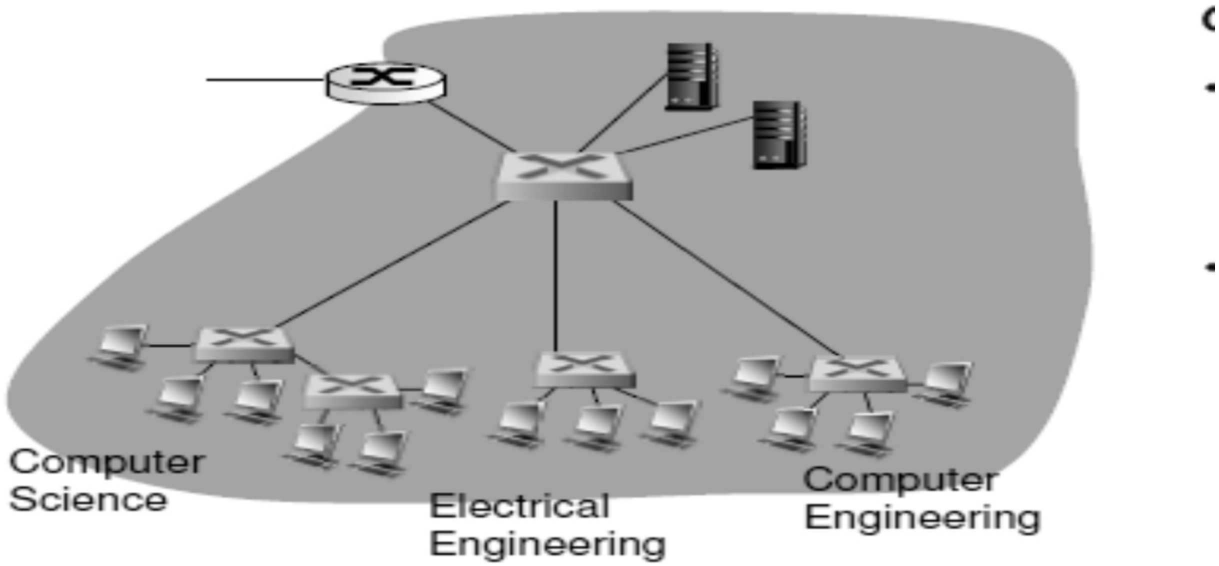
## Virtual Local Area Networks (VLANs)

- Possibility to define several VLANs over a same physical LAN infrastructure
  - Each VLAN has its broadcast domain and has an id.
- However
    - Each physical node is part of one and only VLAN
      - No efficient resource usage

# Prior to Network Virtualization

## Virtual Local Area Networks (VLANs)

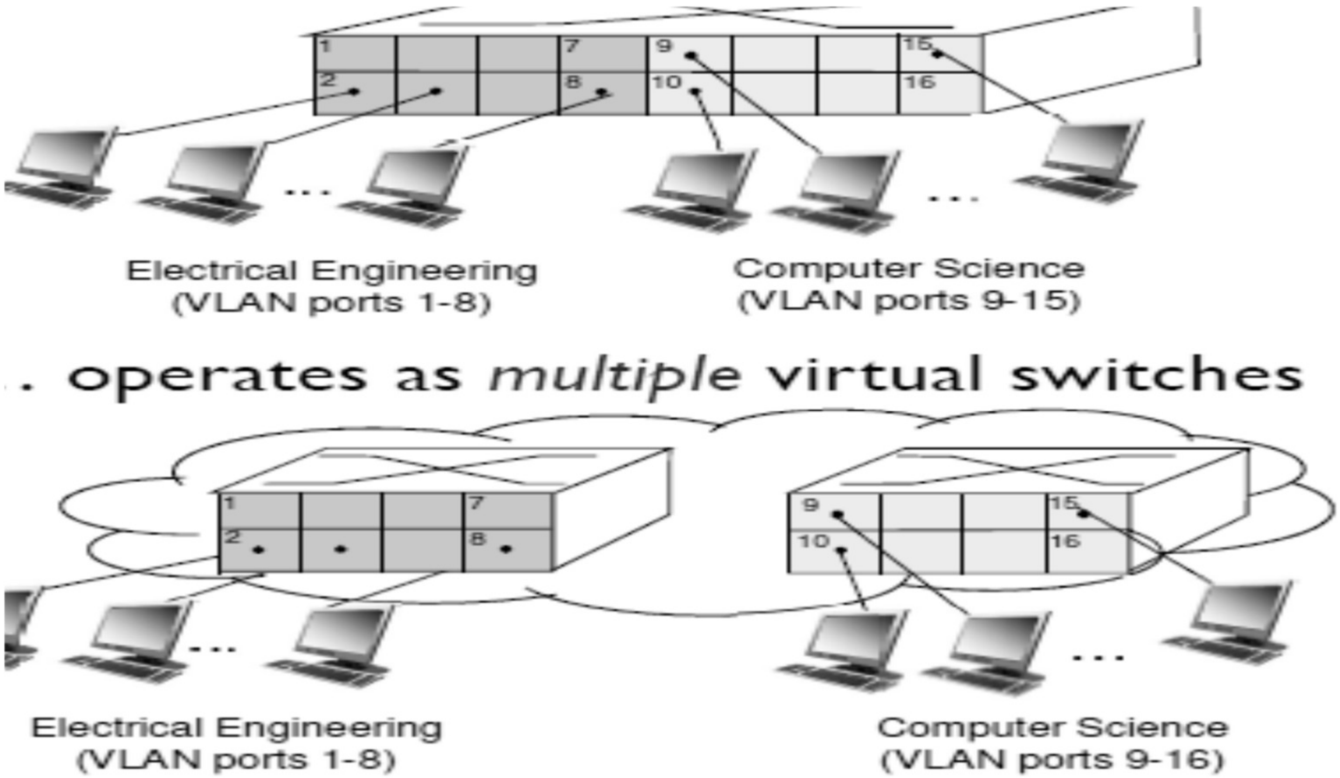
- A LAN (Reference 7)



# Prior to Network Virtualization

## Virtual Local Area Networks (VLANs)

- A VLAN (Reference 7)



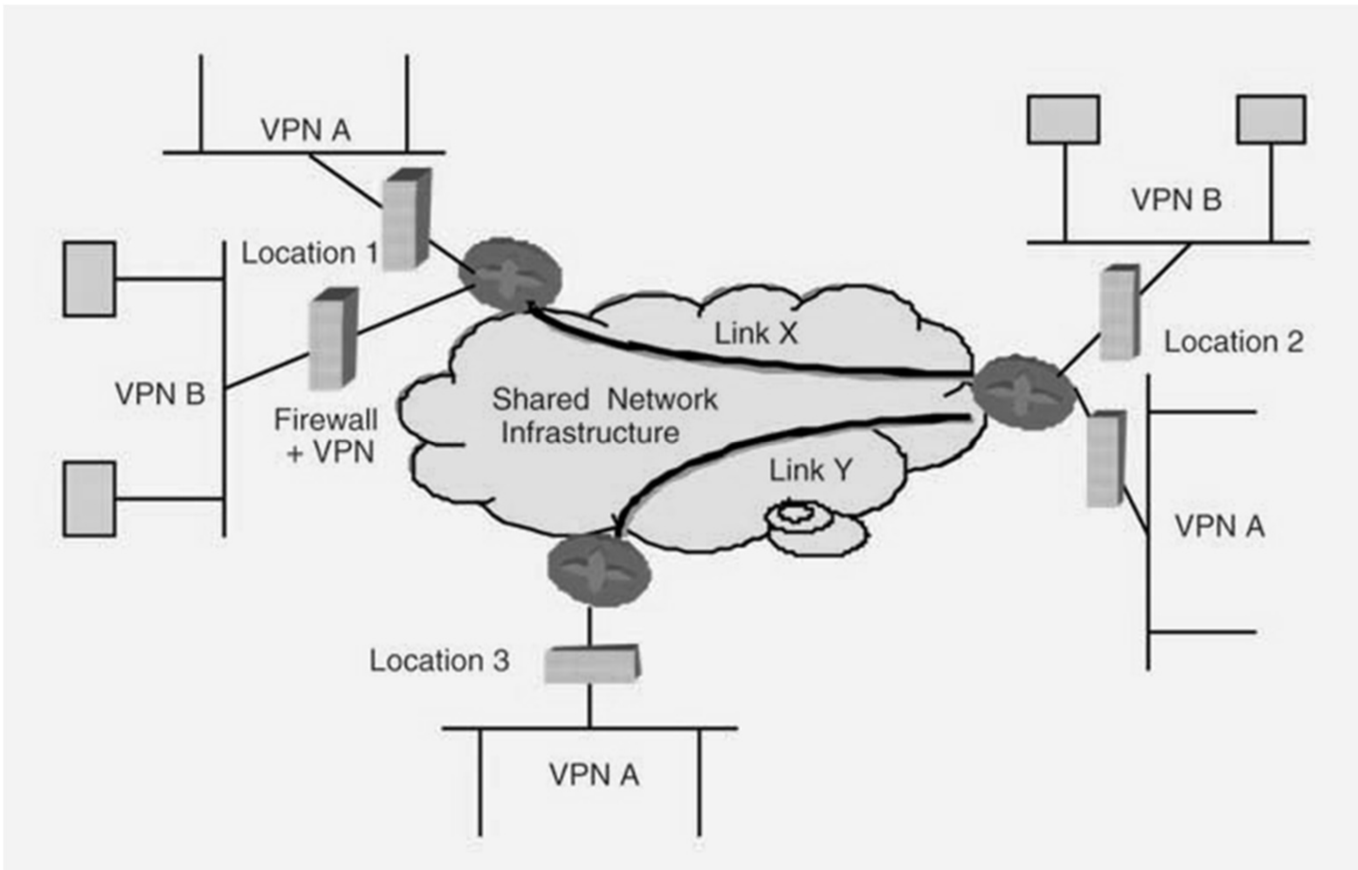
# Prior to Network Virtualization

## Virtual Private Networks

- Possibility to build virtual networks using a shared infrastructure (usually Internet, but might be a dedicated networks)
  - Site interconnection
  - Extranets
- But:
  - No real insolation between the different networks traffic over the shared infrastructure

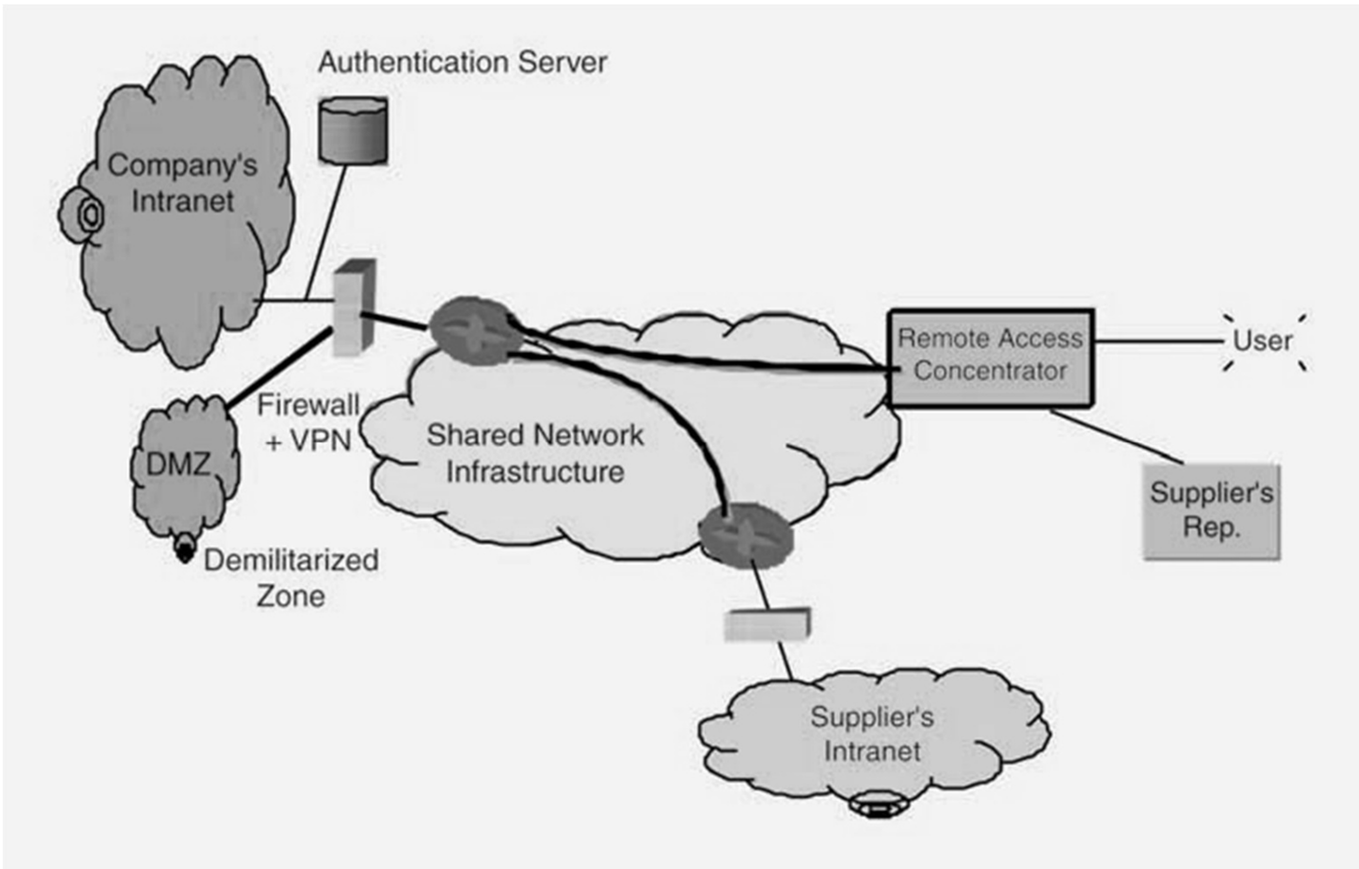
# Prior to Network Virtualization

## Virtual Private Networks – Reference 8 (LAN Interconnection)



# Prior to Network Virtualization

## Virtual Private Networks – Reference 5 (LAN Interconnection)



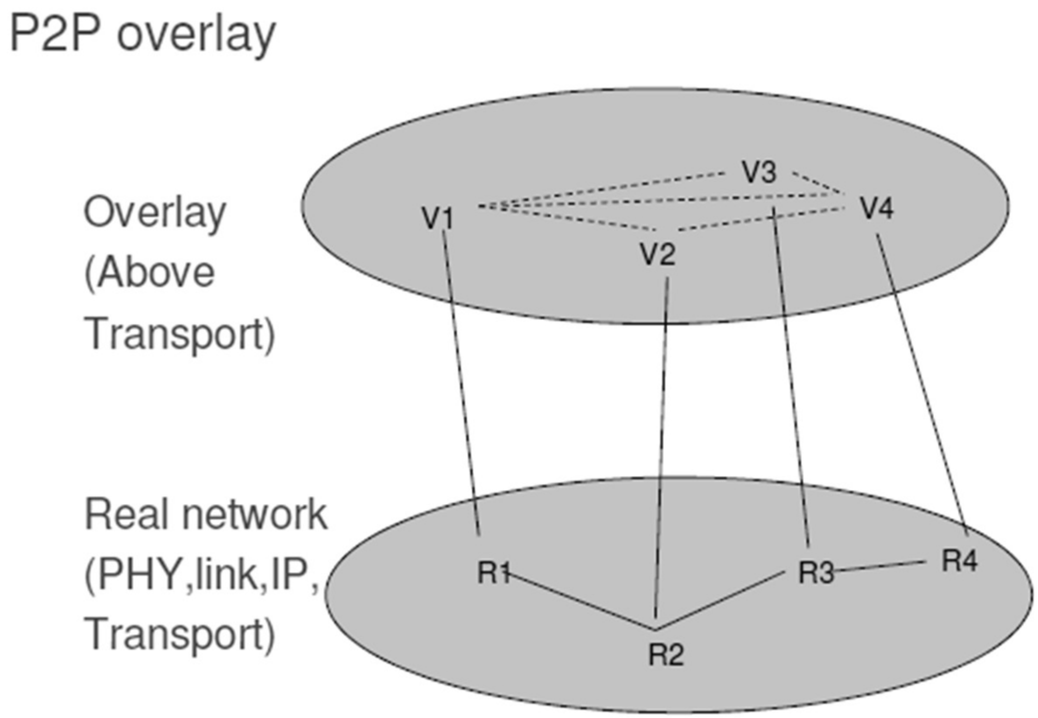
# Prior to Network Virtualization

**Overlays**

- Logical networks built on top of real networks (e.g. skype)
- A same physical node might be part of several overlays
- But:
  - Overlays might interact in a harmful way
  - Used mainly at application layer and does not enable experimentation of lower layer protocols

# Prior to Network Virtualization

## Overlays



# Prior to Network Virtualization

**Overlays**

P2P overlay

- Characteristics
  - own topology that may be different from the topology of the real network
  - Own protocols that may be different from the protocols used in the real network
  - May come with an application embedded in it (e.g. Skype) or as an infrastructure that can be used by other applications (e.g. CHORD)
  - APIs, toolkits are provided when the application is not embedded in the overlay

# A Case Study on Network Virtualization (Reference 6)

Business model of current Internet:

- Internet Service Providers (ISPs) (e.g. Bell, Rogers)
- Service Providers (eg. Google, Akamai)

# A Case Study on Network Virtualization
# Reference 6

New business model (4 roles):

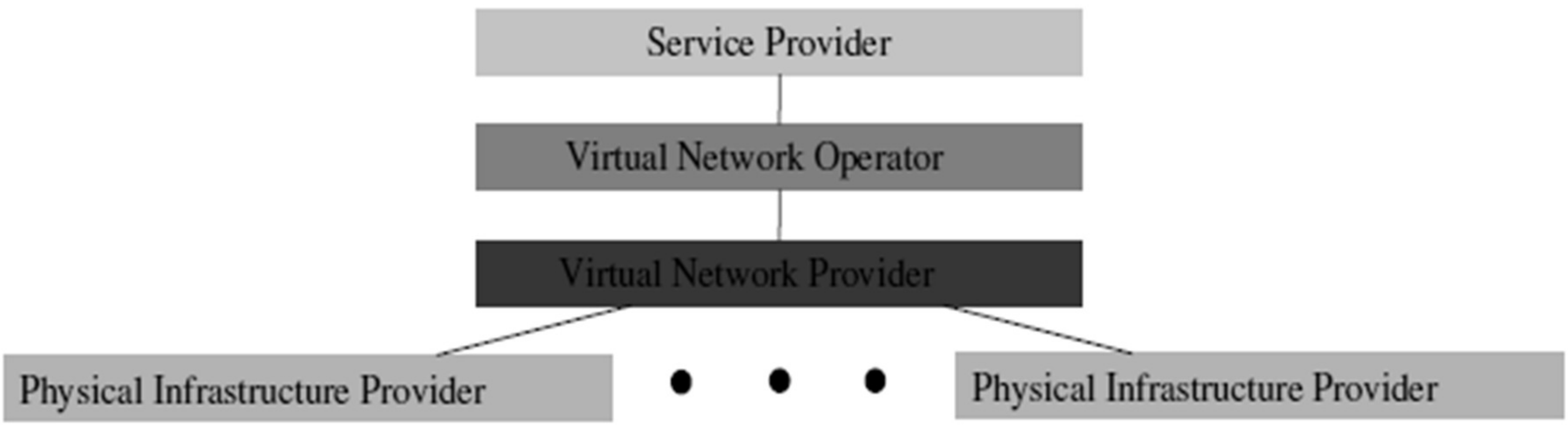
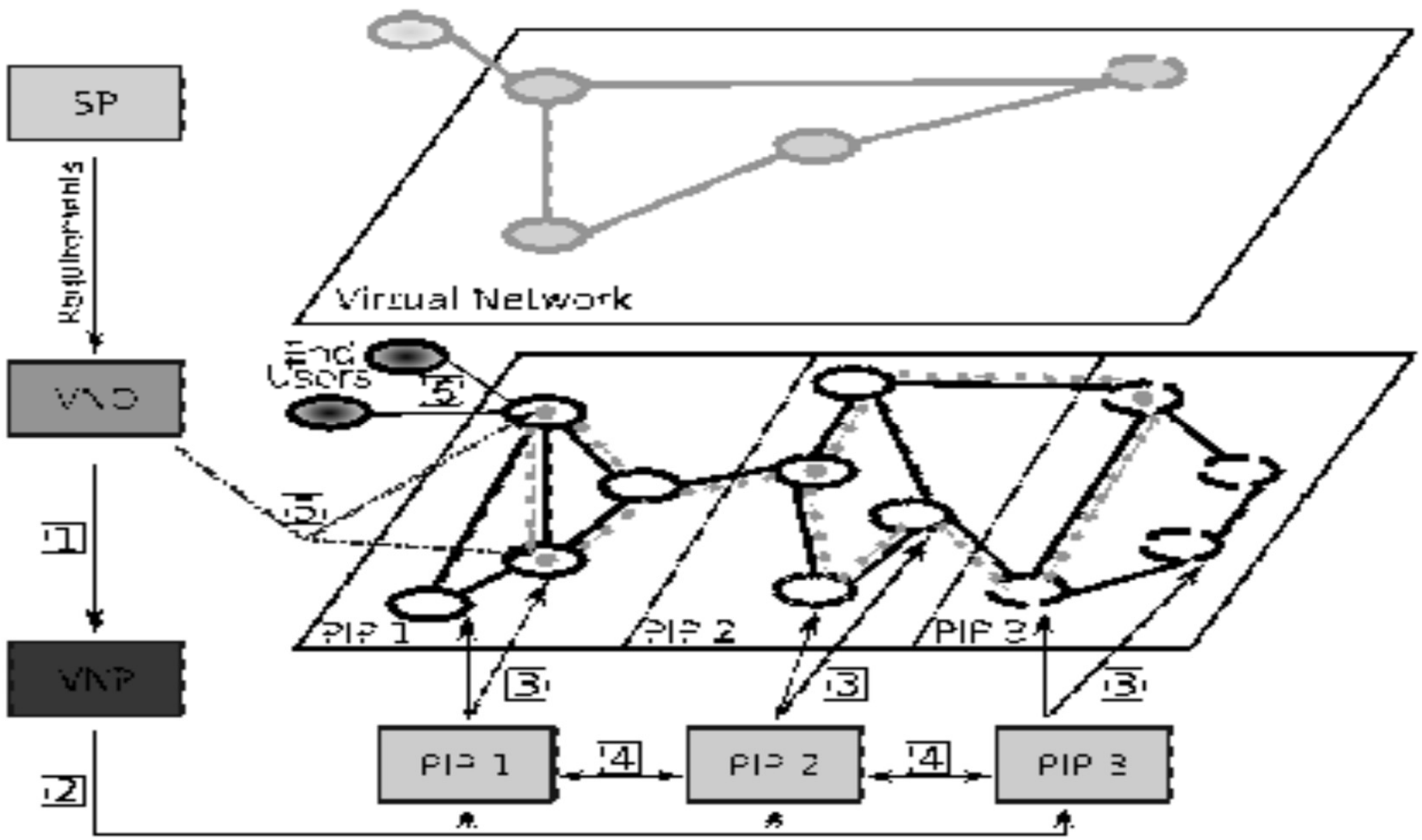



Figure 1: VNet Management and Business Roles

# A Case  Study on Network Virtualization
# Reference 6

New business model (6 interfaces):



# A Case Study on Network Virtualization
# Reference 6

Simplified scenario

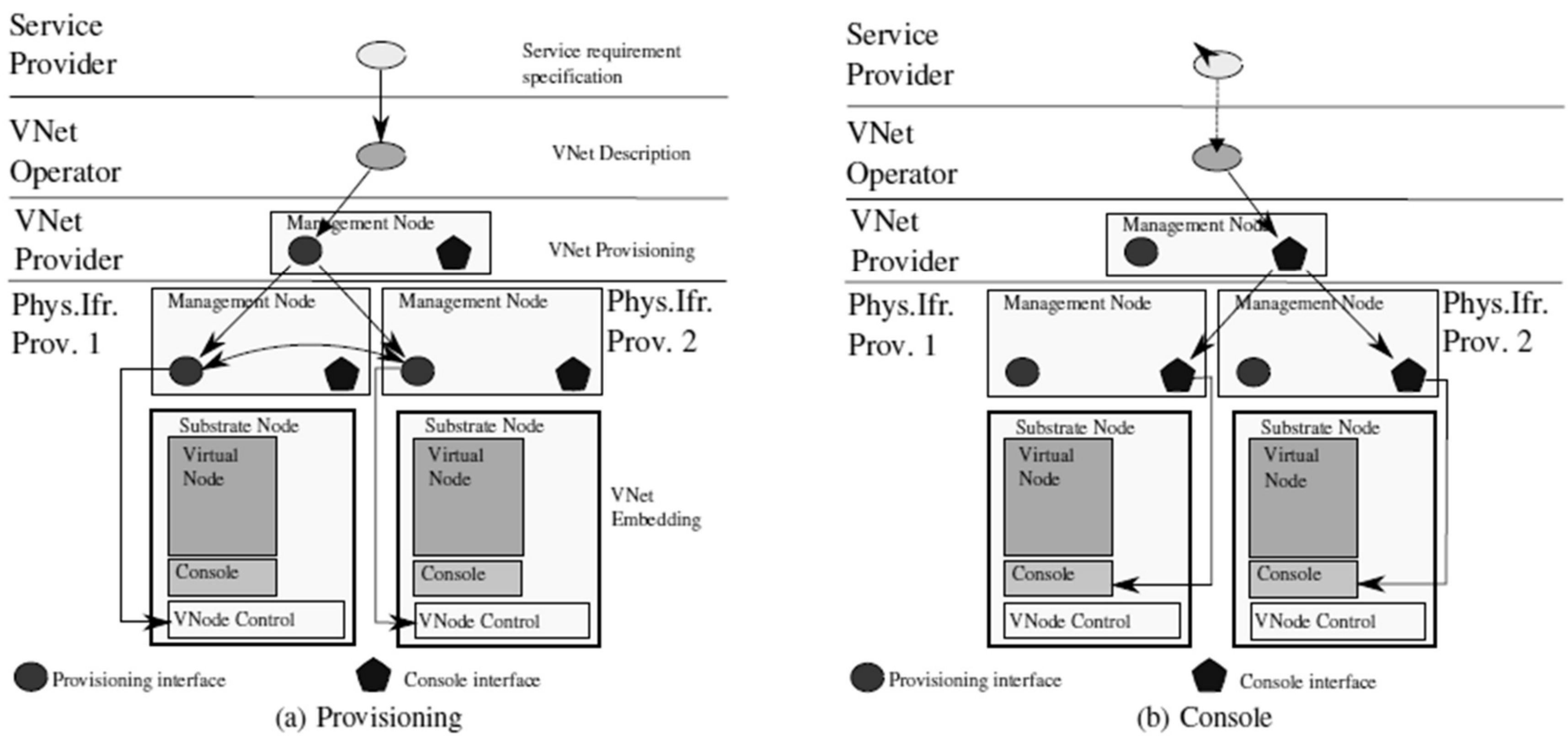



Figure 3: VNet provisioning (a) and console architecture (b).

# A Case Study on Network Virtualization (Reference 6)

Prototype

- Node level virtualization
  - XEN
- VNET description
  - XML

# A Case Study on Network Virtualization (Reference 6)

Topology used for Vnet instantiation measurements (end to end from Vnet request by service provider till full provisioning of VNET

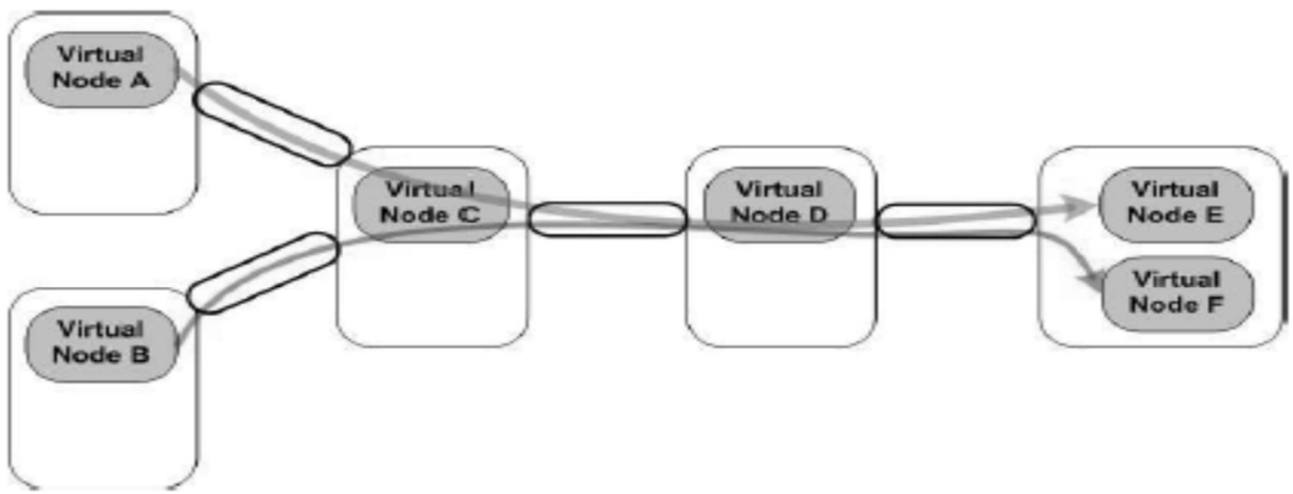



Figure 6: Experimental topology.

# The End