Beyond the Clouds, the DISCOVERY Initiative

Localization is a key element to deliver
efficient as well as sustainable Utility Computing Solutions

MINES TELECOM
INSTITUT Mines-Télécom

Inria lina

Adrien Lèbre / Ascola Project Team
August, 2013

MINES Nantes

Context

xxx Computing

- Meta / Cluster / Grid / Desktop / "Hive" / Cloud / Sky ...
- A common objective: provide computing resources (both hardware and software) in a flexible, transparent, secure, reliable, ... way

⇒ xxx as Utility Computing (UC)

- Challenges

  Data Sharing

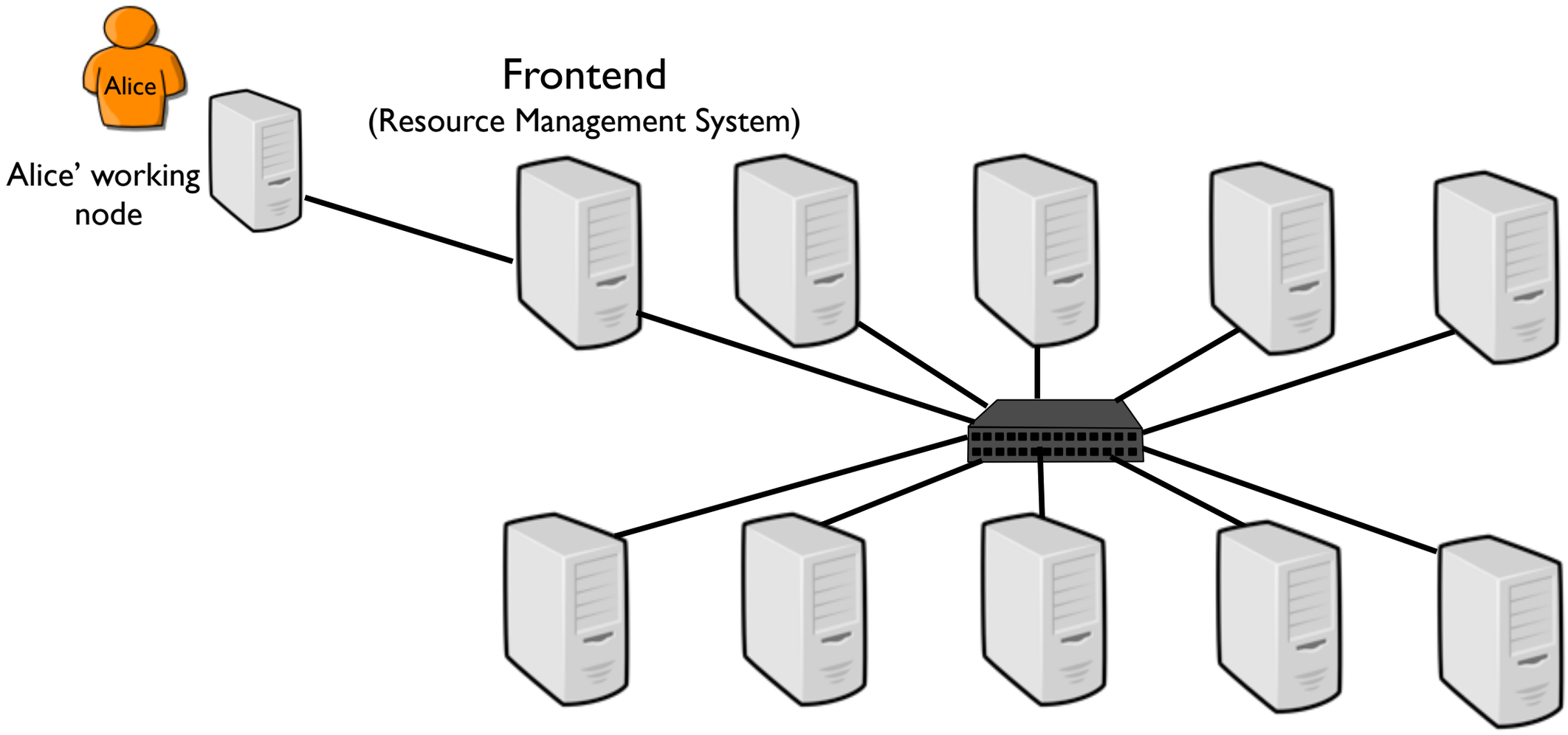
  Software/Hardware heterogeneity

  Security (Isolation between applications, ...)

  Reliability / Resiliency ...

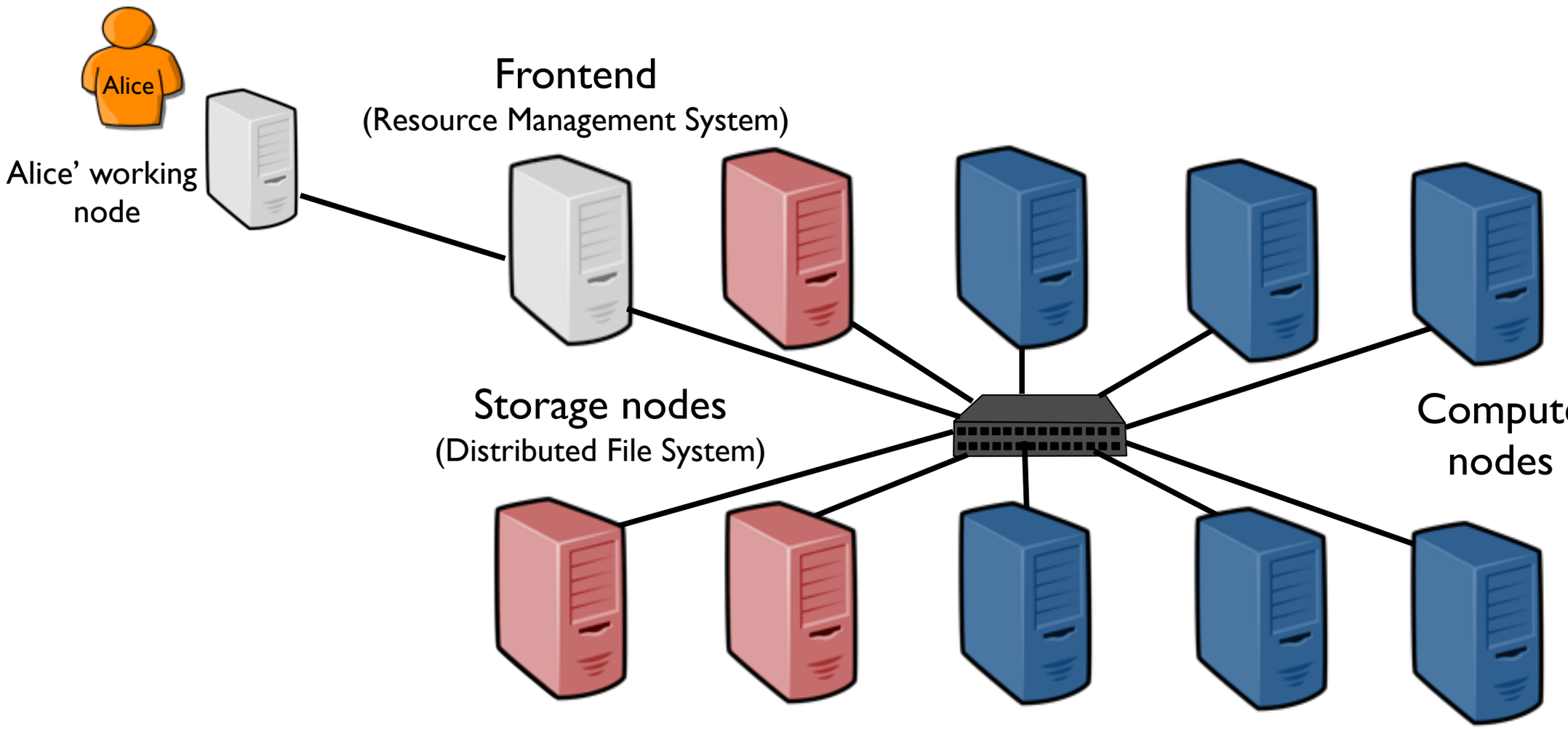
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**



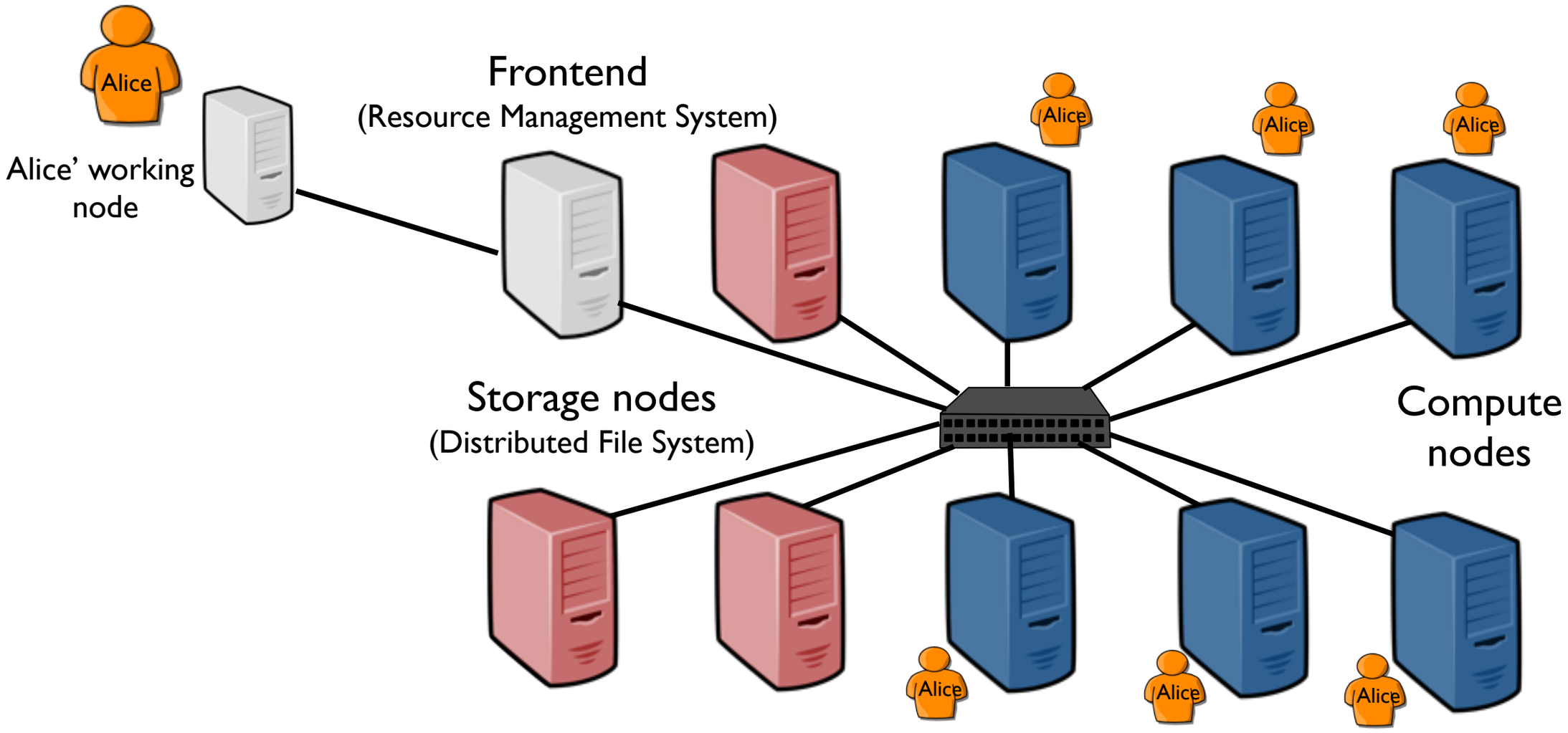
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**



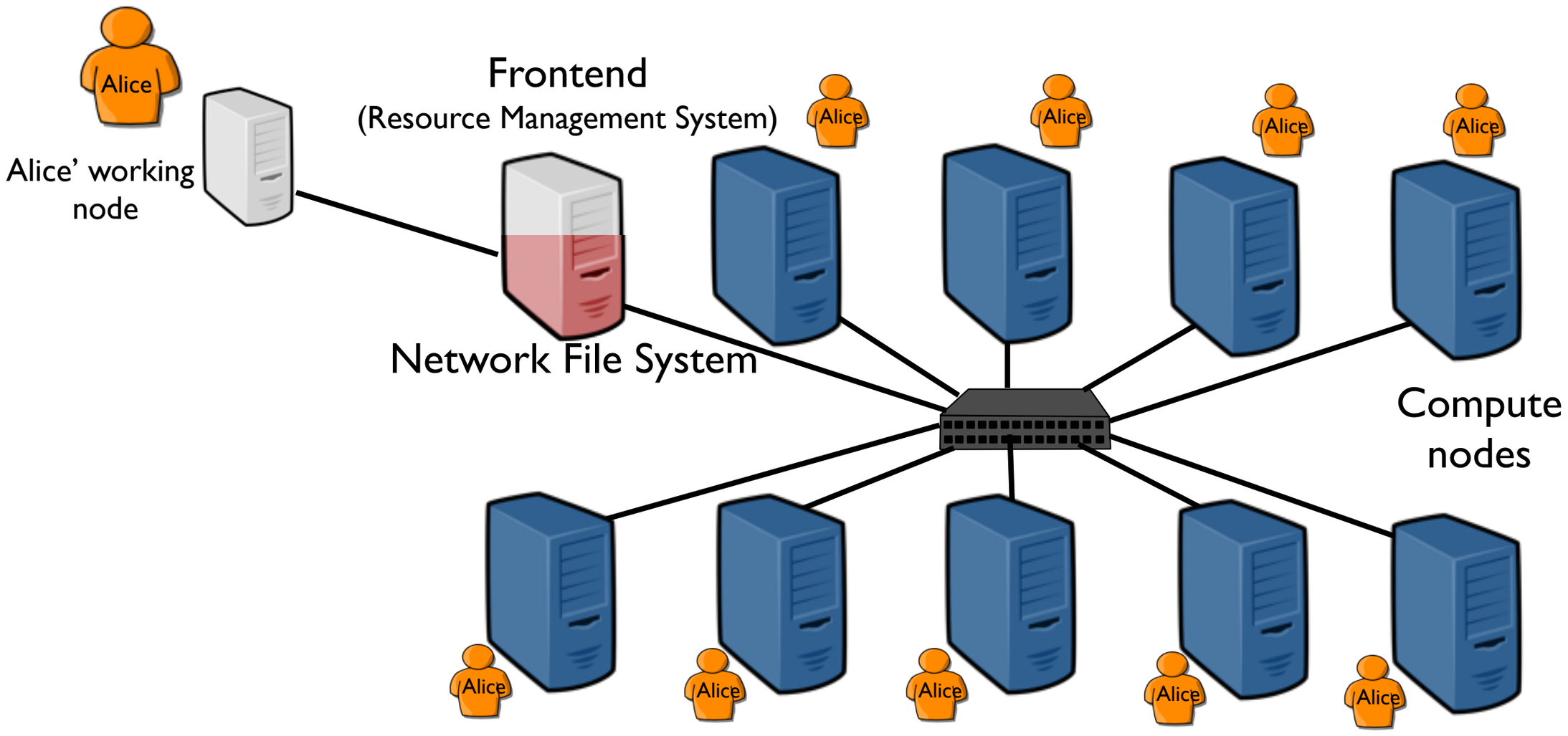
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**



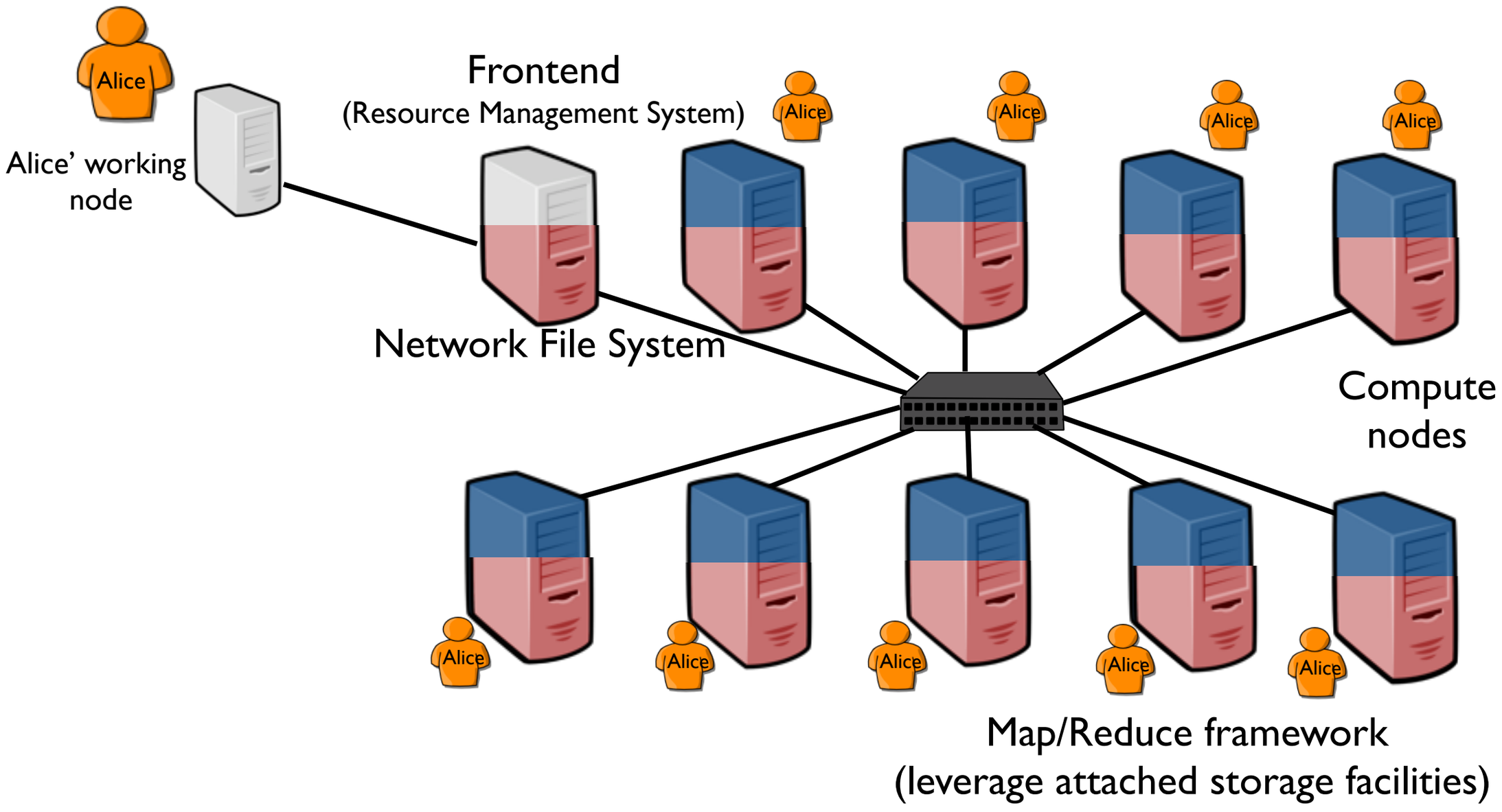
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**



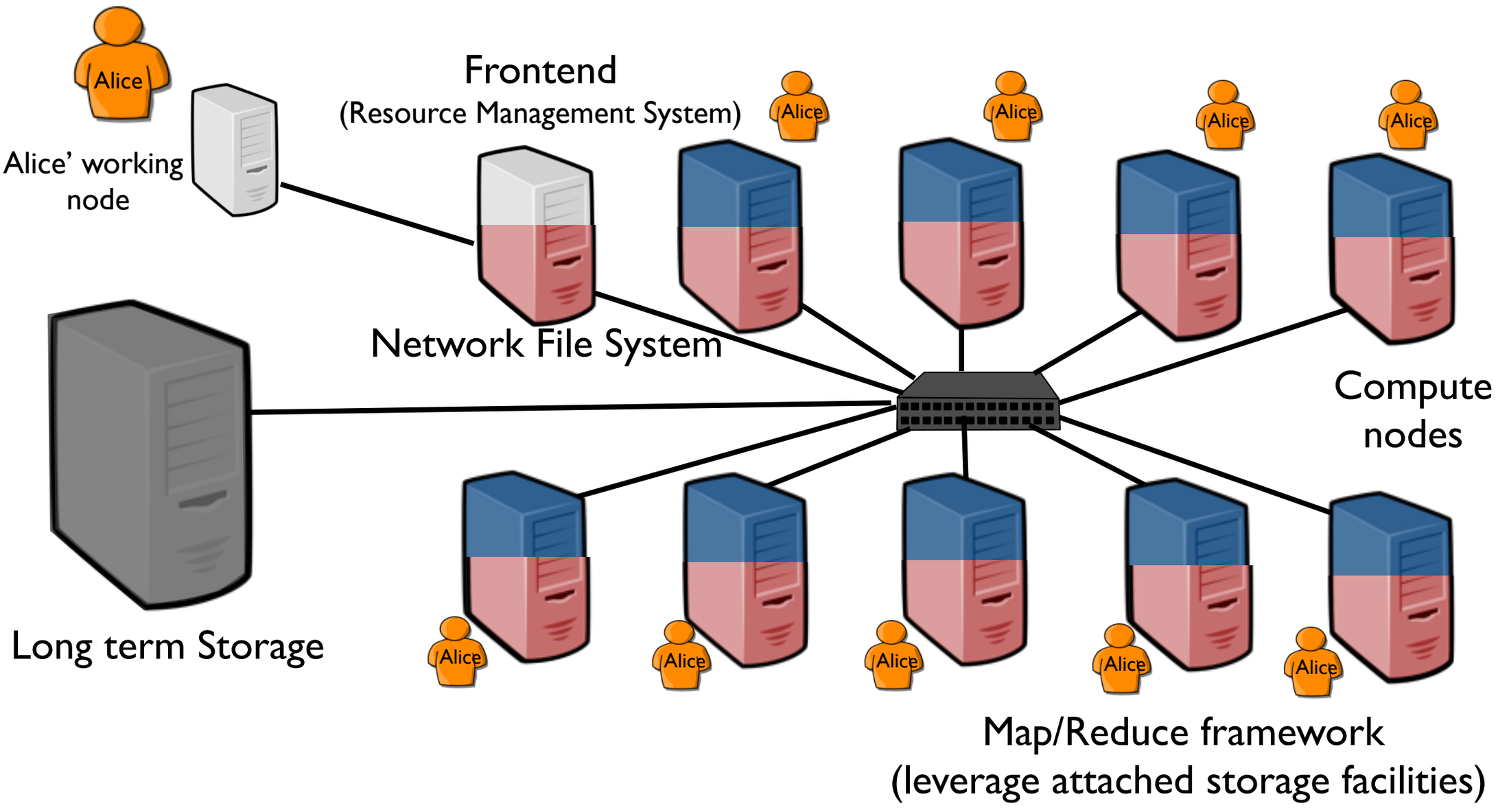
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**



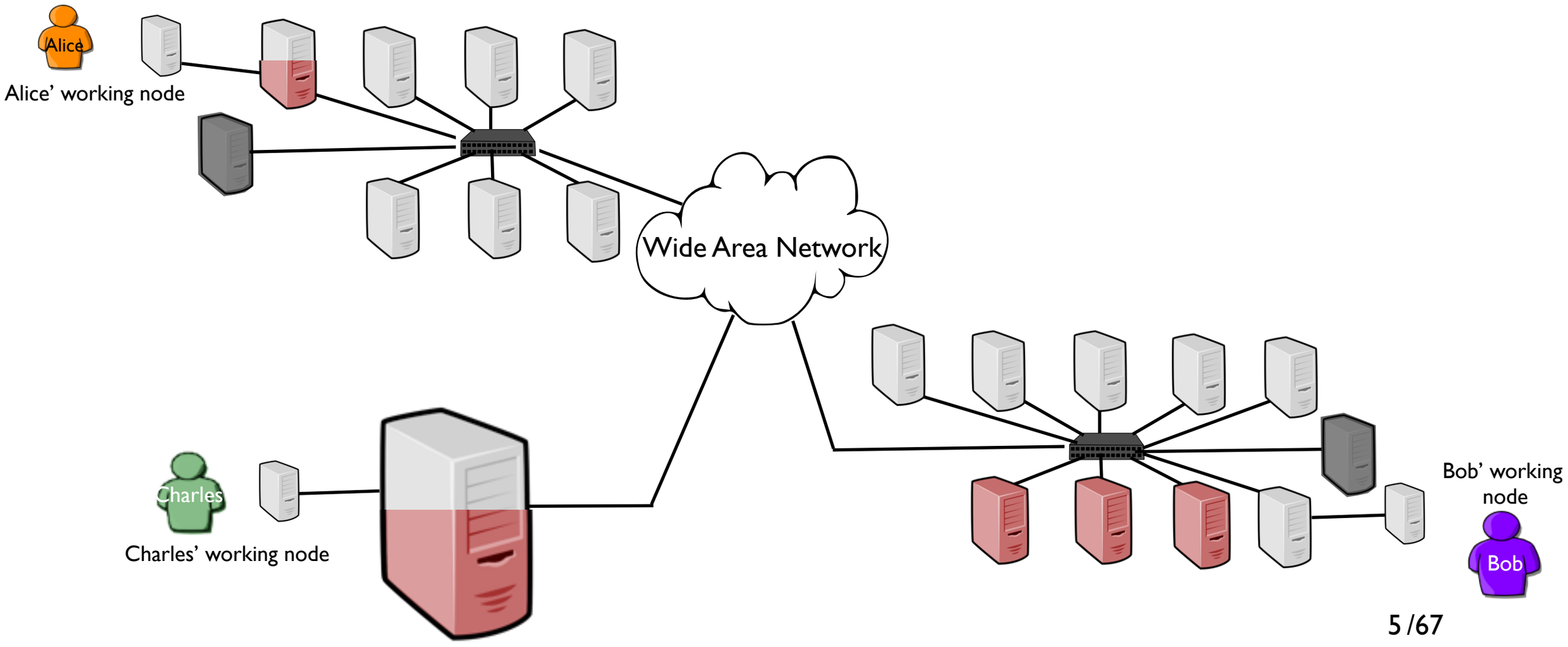
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**



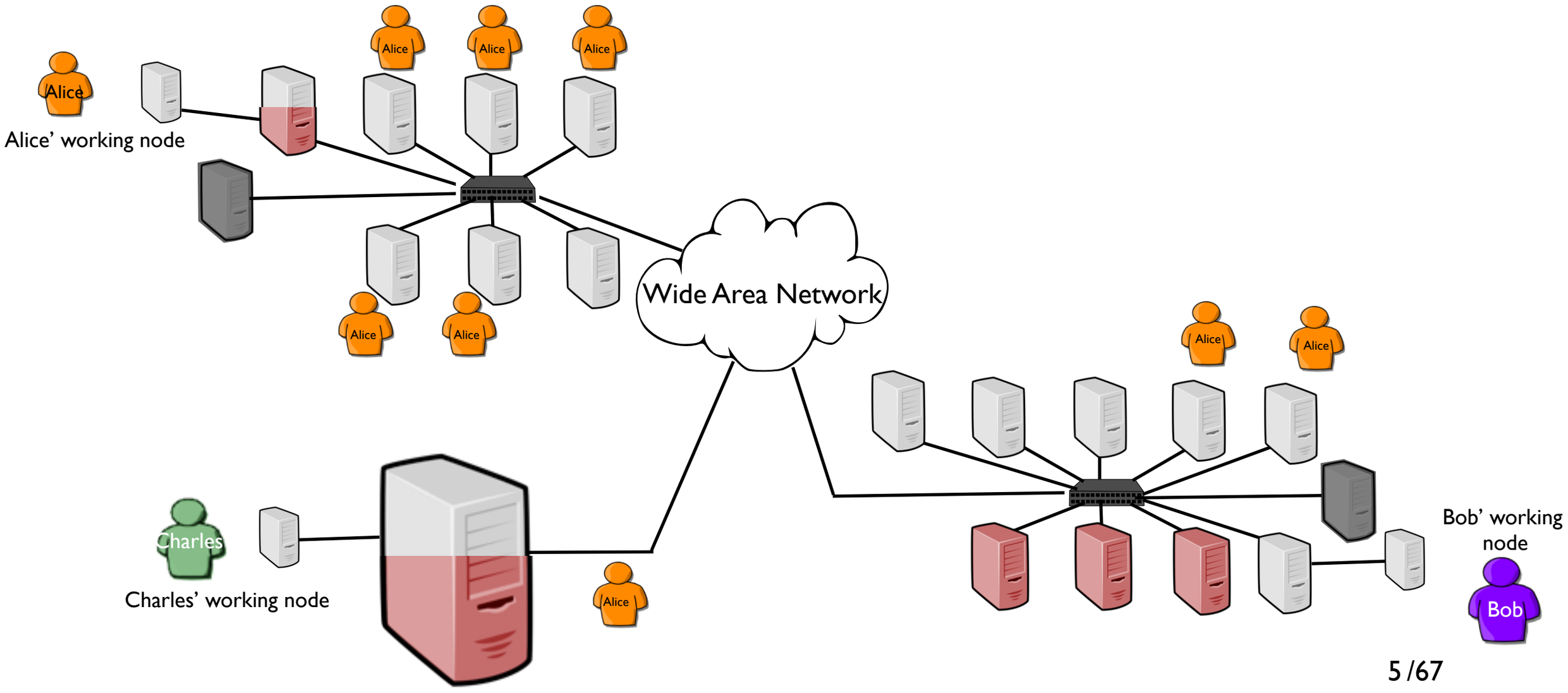
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



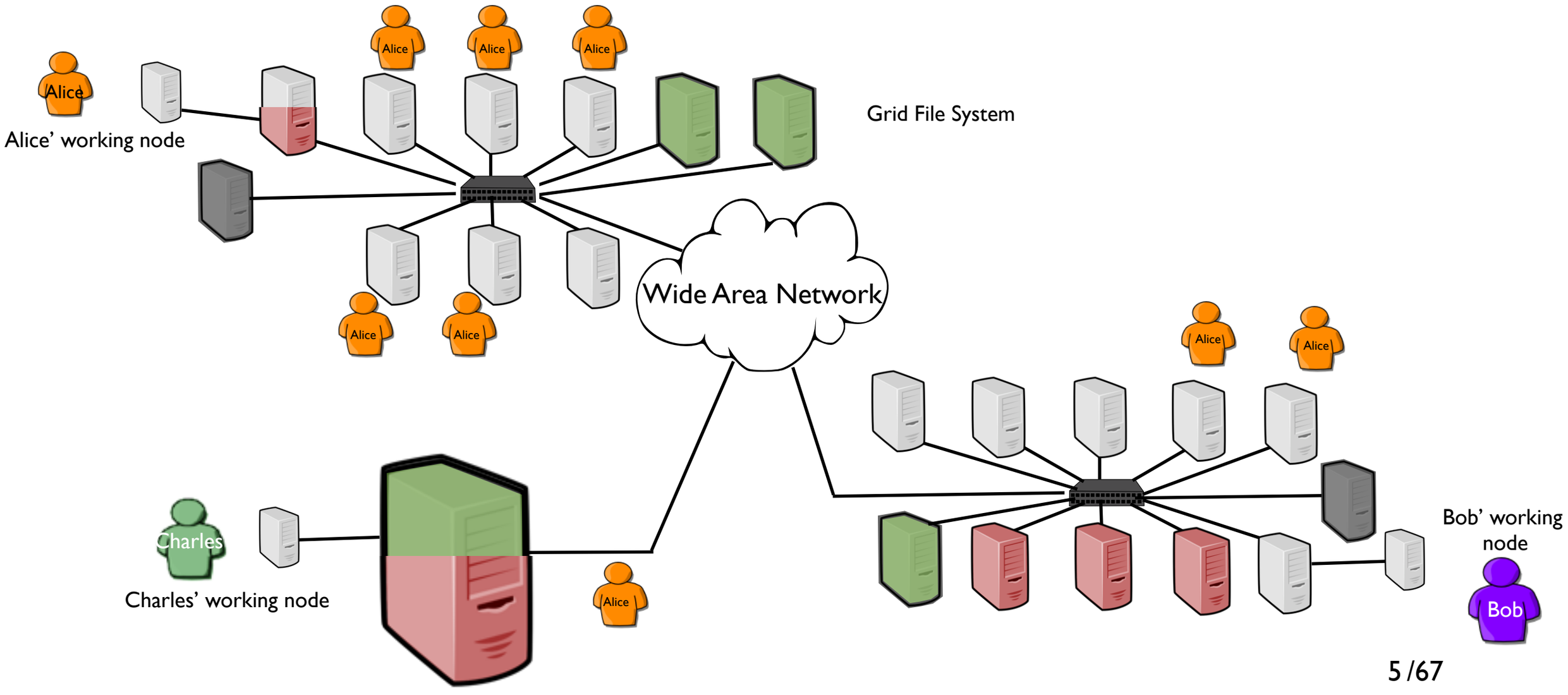
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



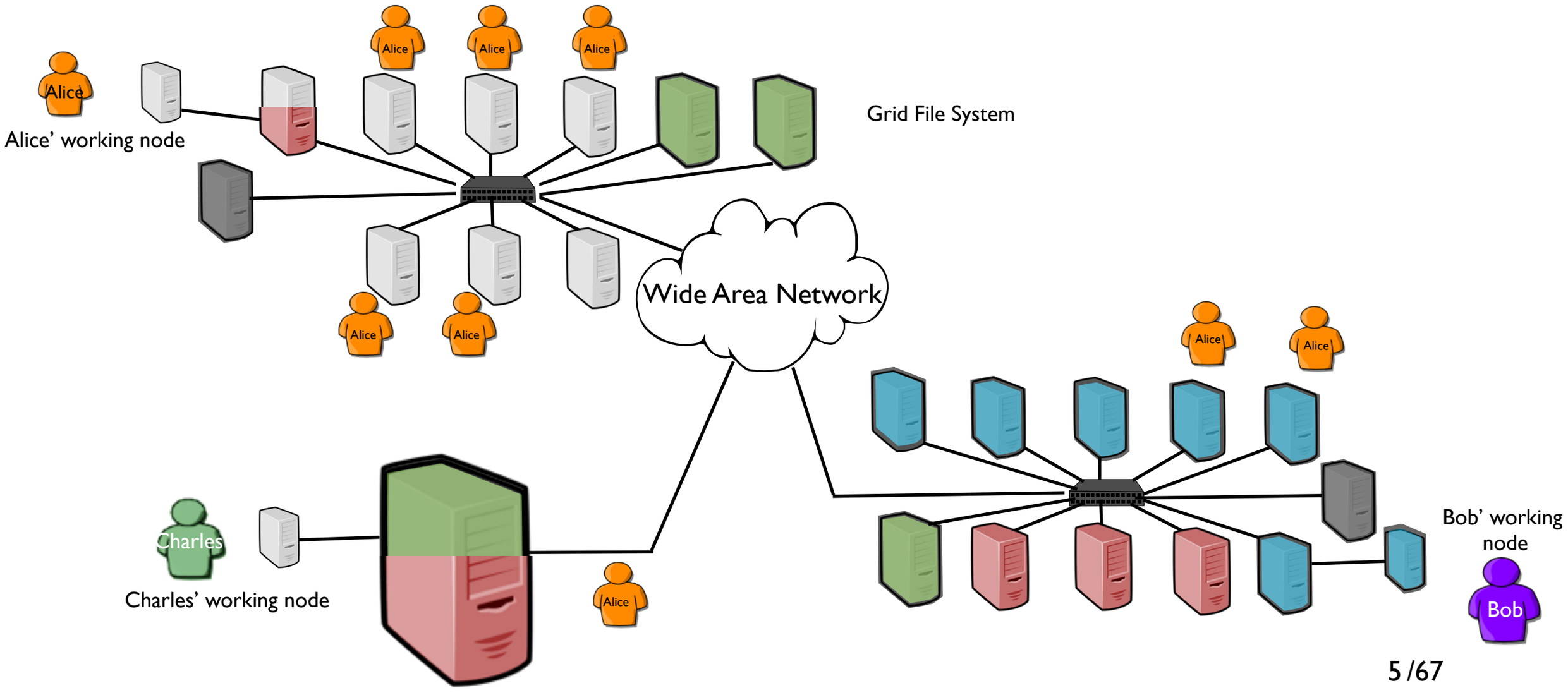
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



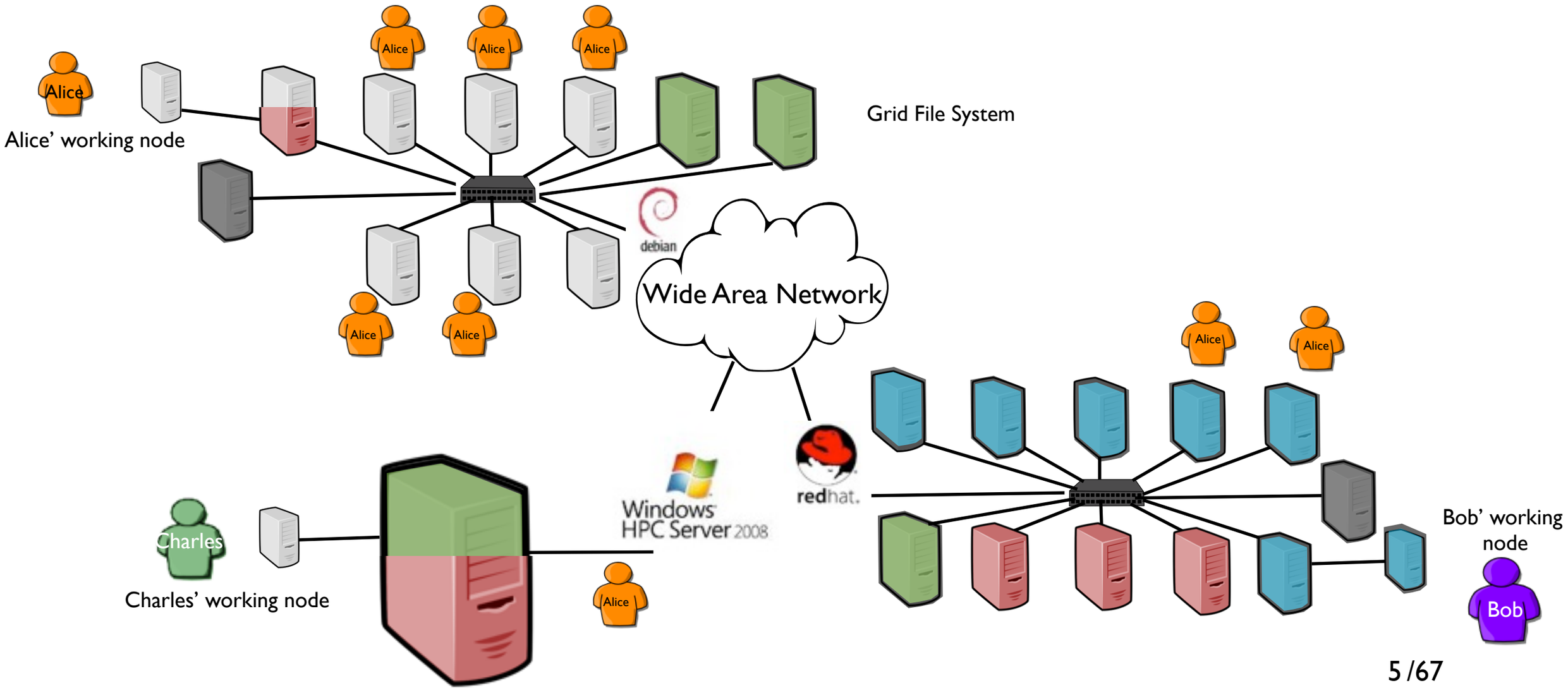
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



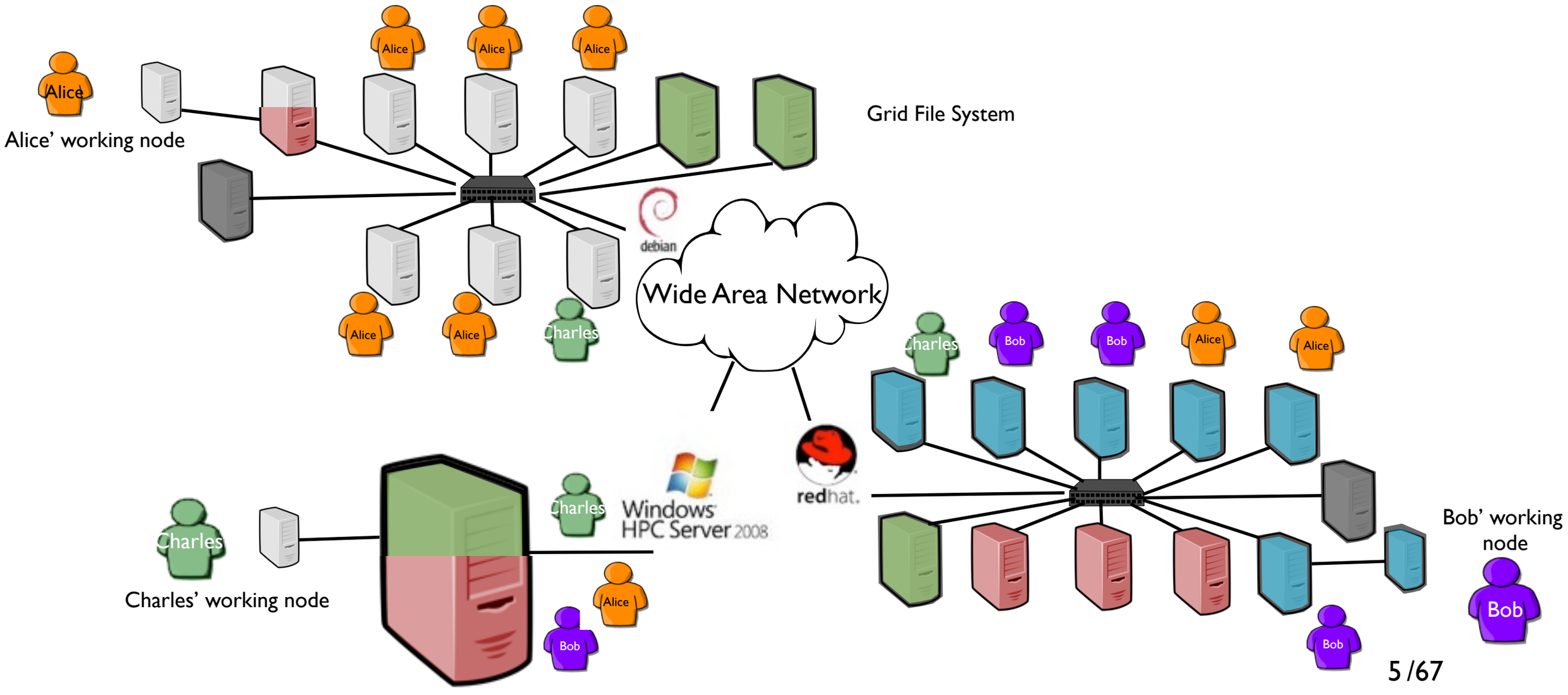
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



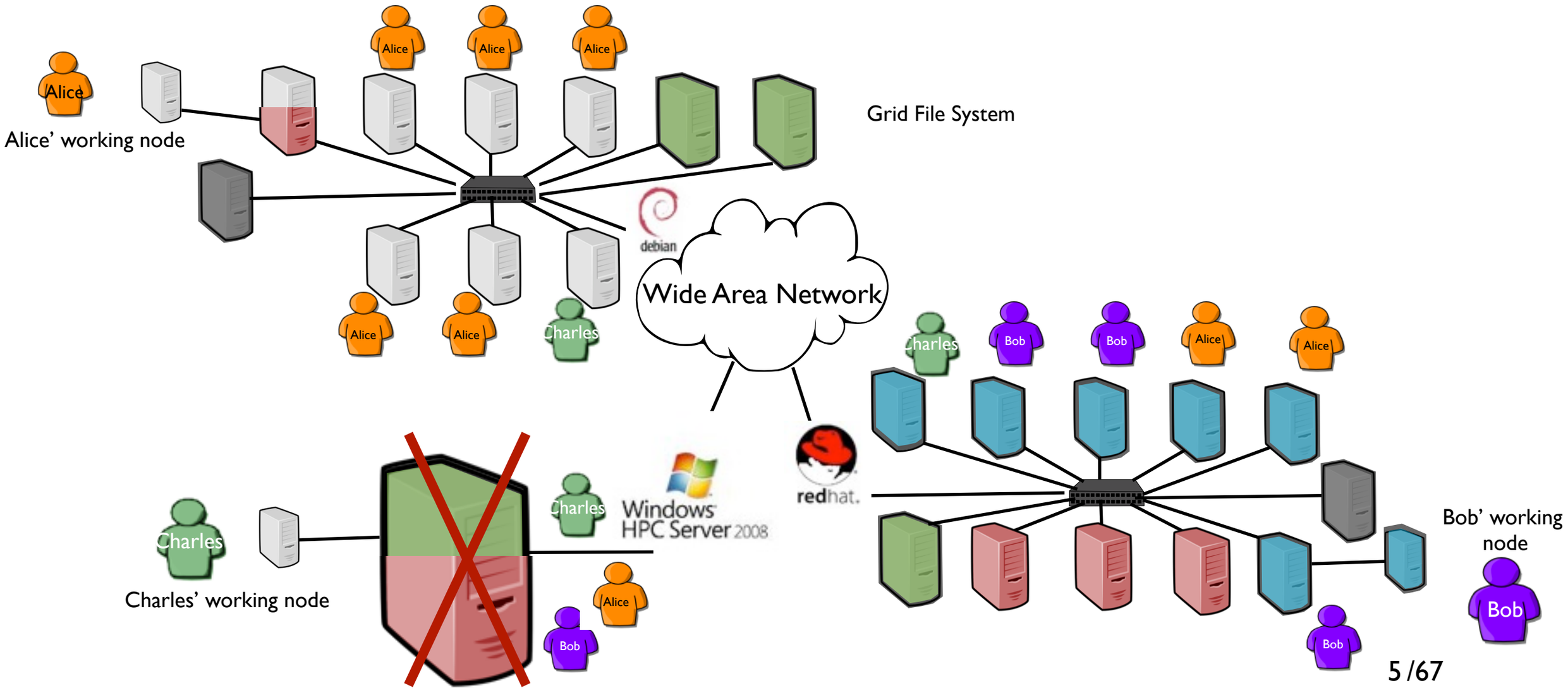
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



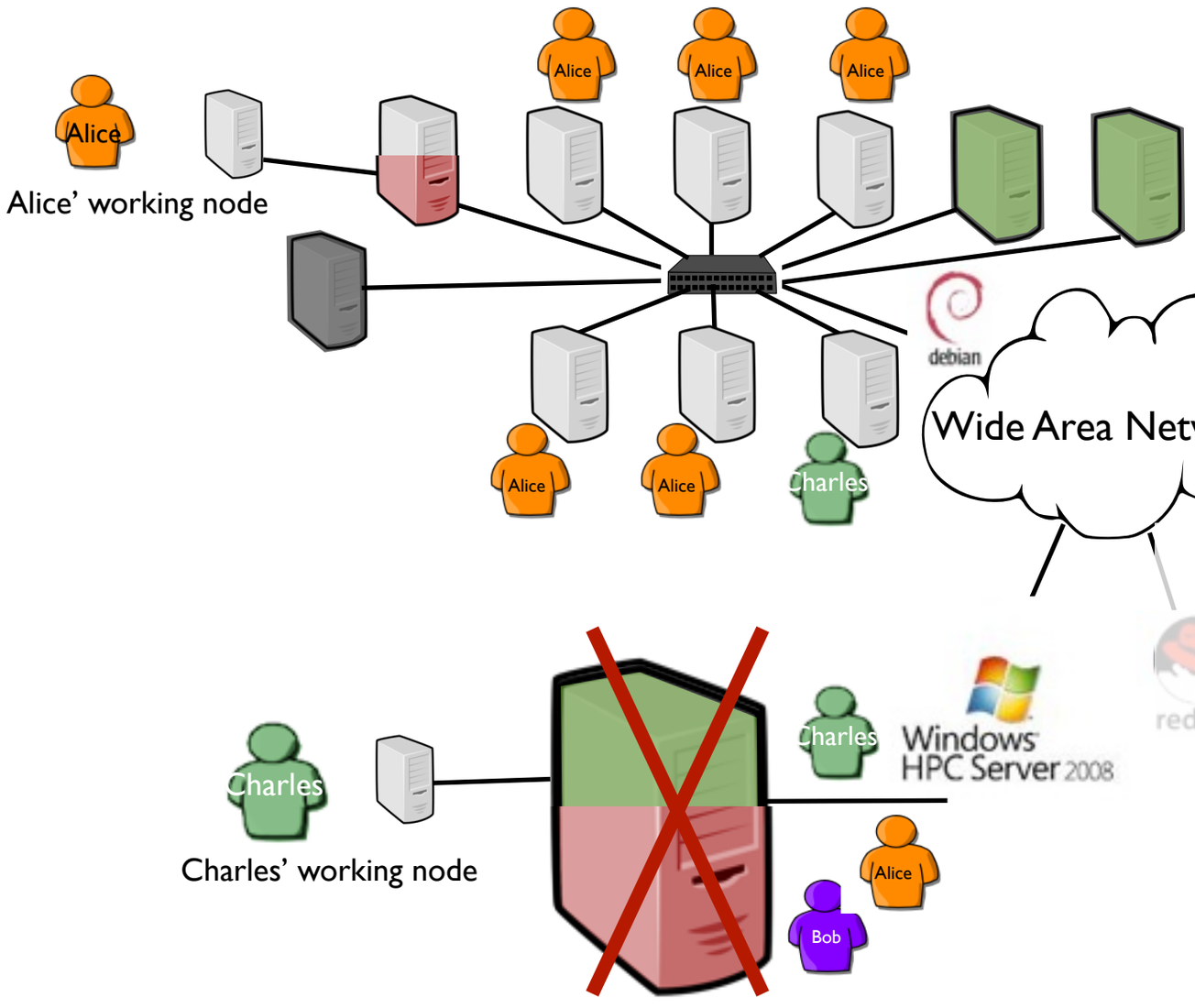
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



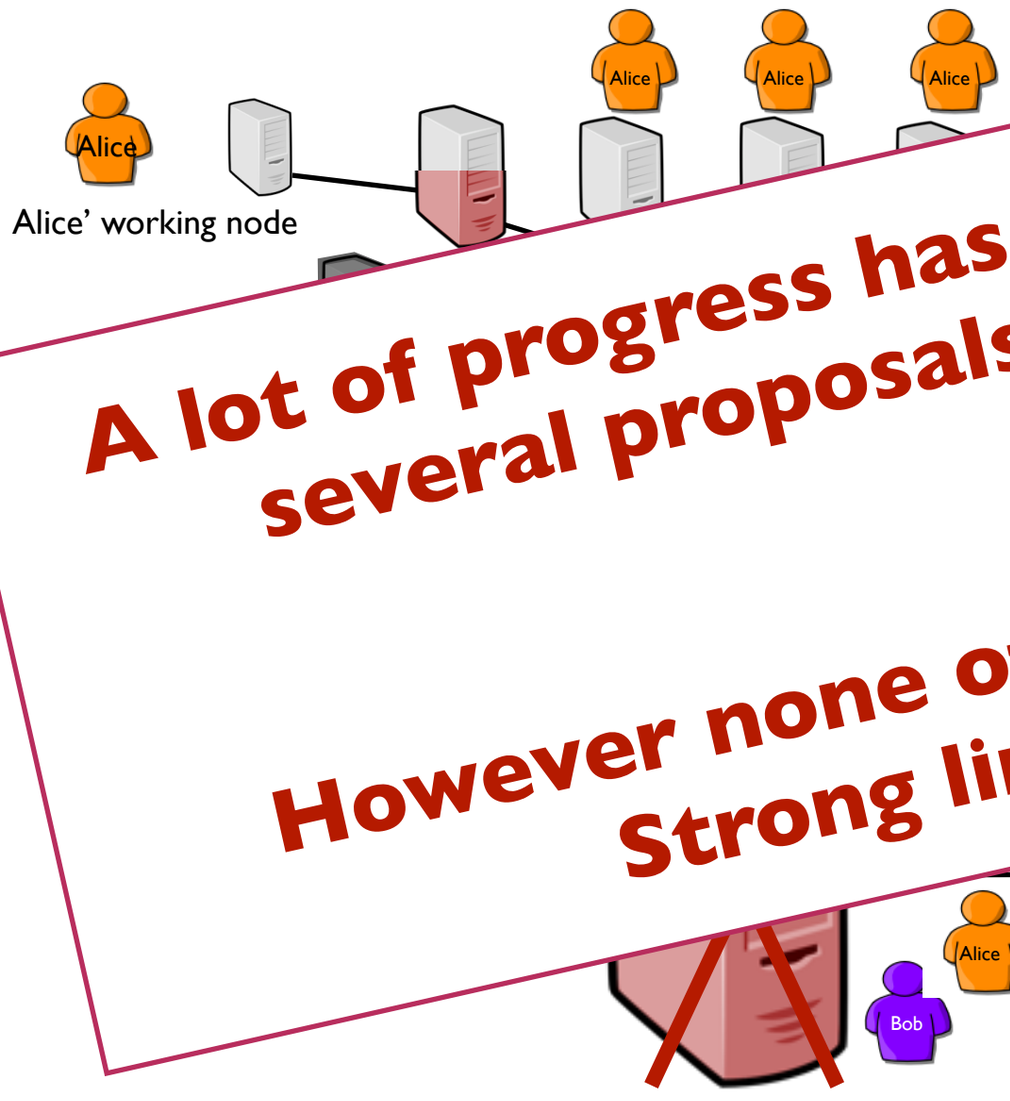
Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**



Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**

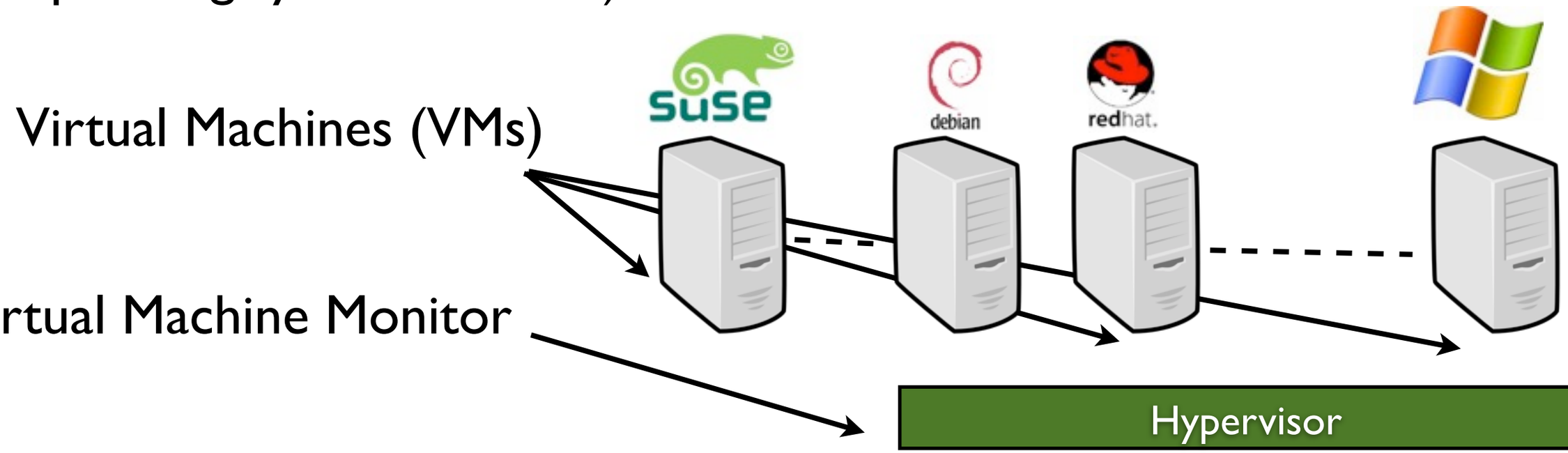


A lot of progress has been done since the 90's and several proposals partially addressed these concerns.

However none of them is mature enough and Strong limitations still persist !

Here Comes *System Virtualization*

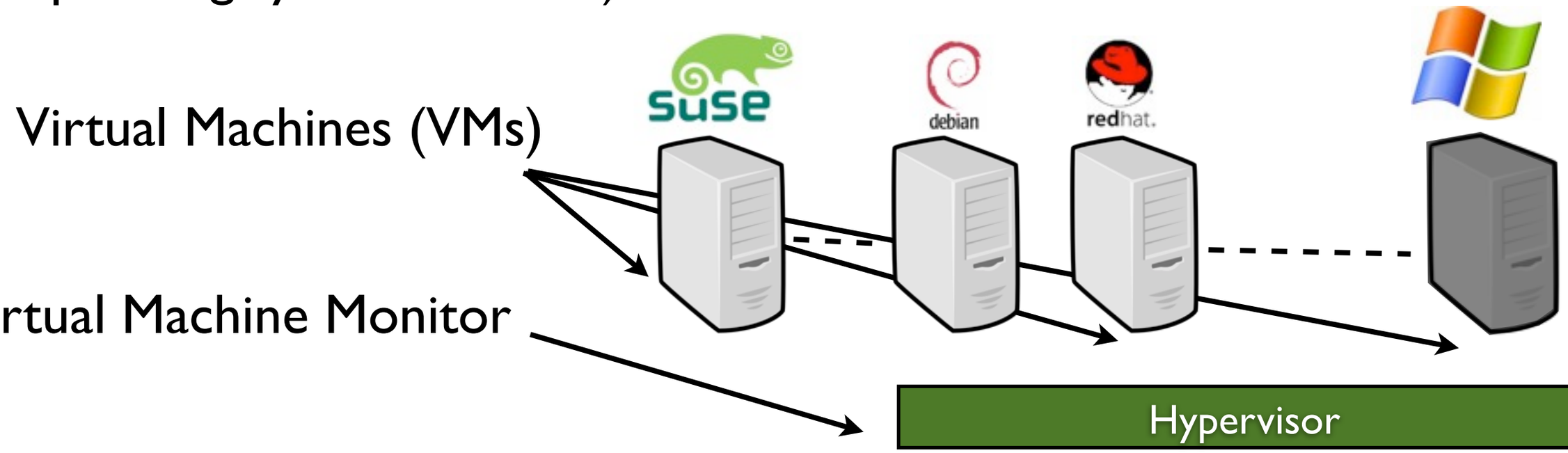
- One to multiple OSes on a physical node thanks to a hypervisor (an operating system of OSes)



"A ***virtual machine*** (VM) provides a faithful implementation of a physical processor's hardware running in a protected and isolated environment.
Virtual machines are created by a software layer called the ***virtual machine monitor*** (VMM) that runs as a privileged task on a physical processor."

Here Comes *System Virtualization*

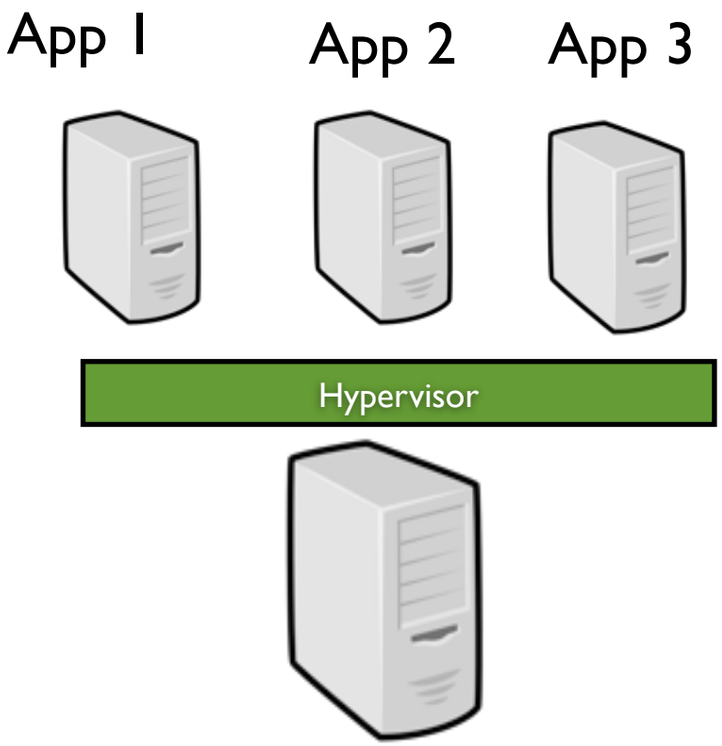
- One to multiple OSes on a physical node thanks to a hypervisor (an operating system of OSes)



"A ***virtual machine*** (VM) provides a faithful implementation of a physical processor's hardware running in a protected and isolated environment.
Virtual machines are created by a software layer called the ***virtual machine monitor*** (VMM) that runs as a privileged task on a physical processor."

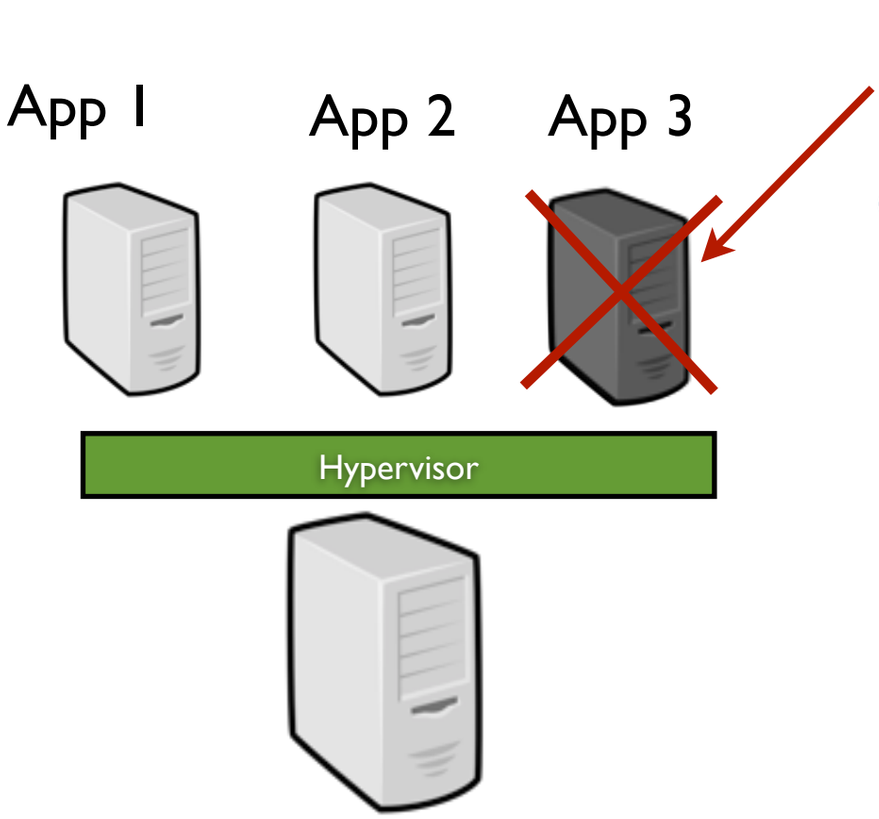
Physical Machine (PM)

VM Capabilities



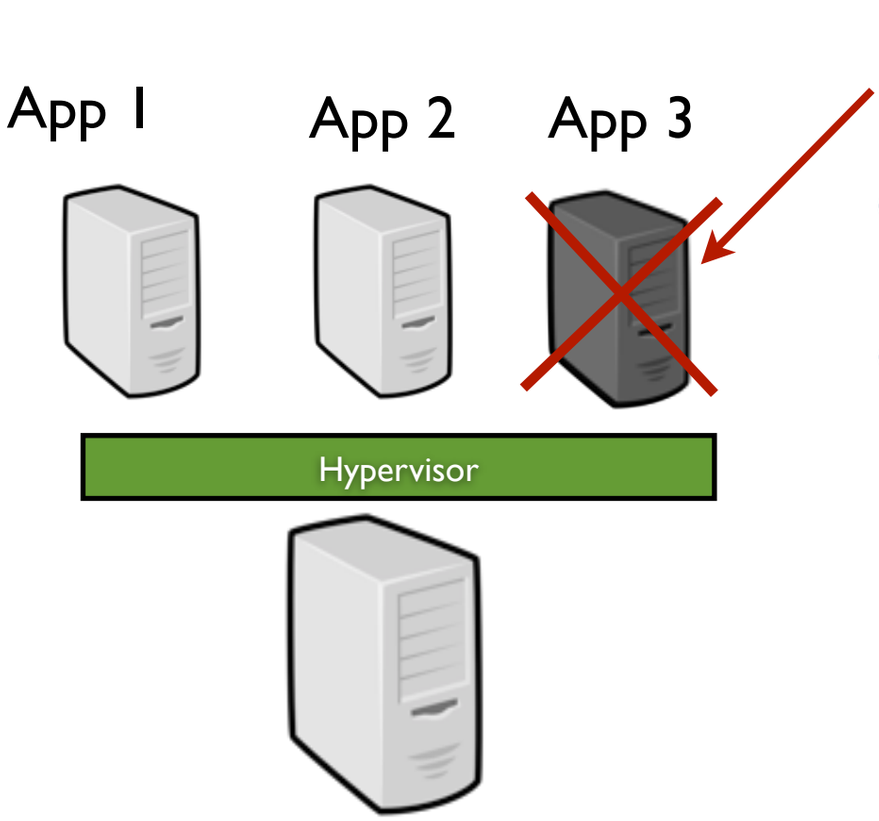
- Isolation (“security” between each VM)

VM Capabilities



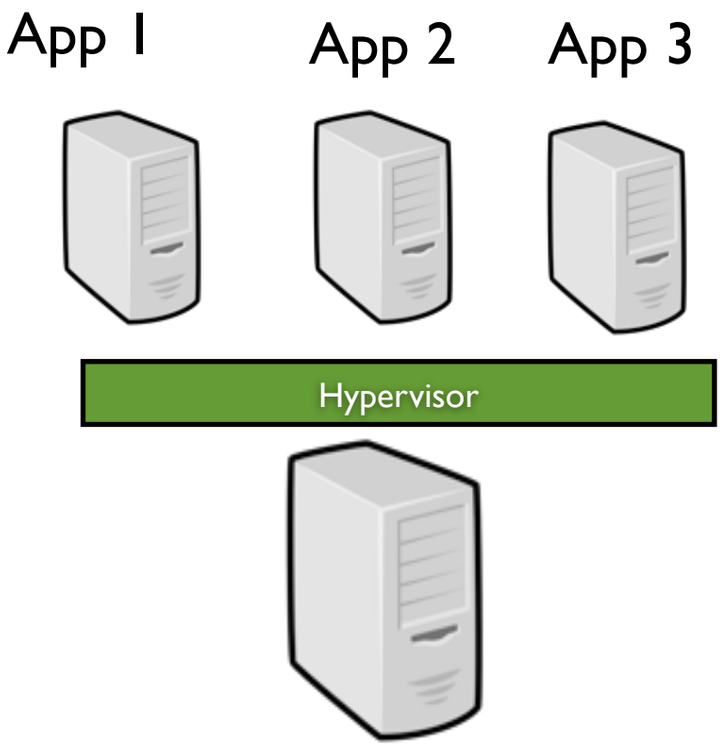
- Isolation (“security” between each VM)

VM Capabilities



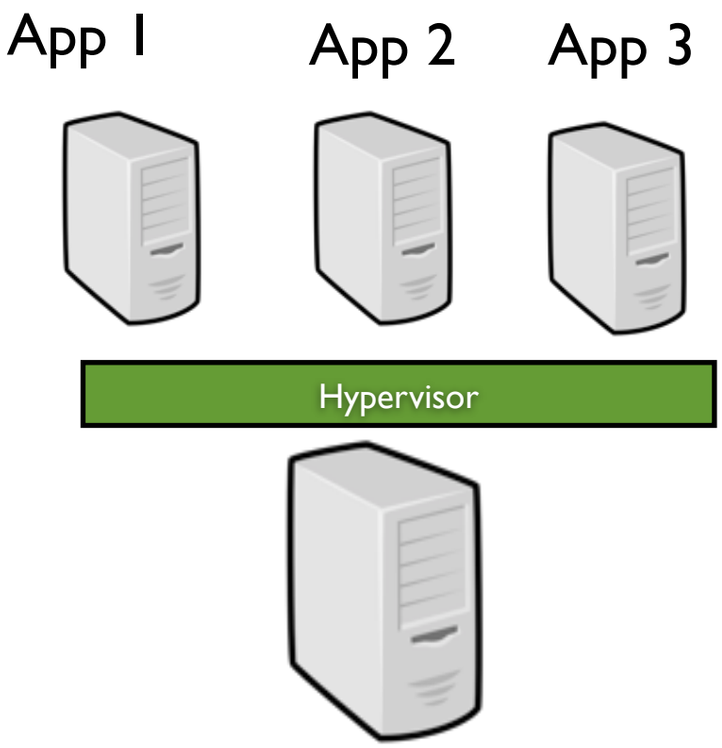
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

VM Capabilities



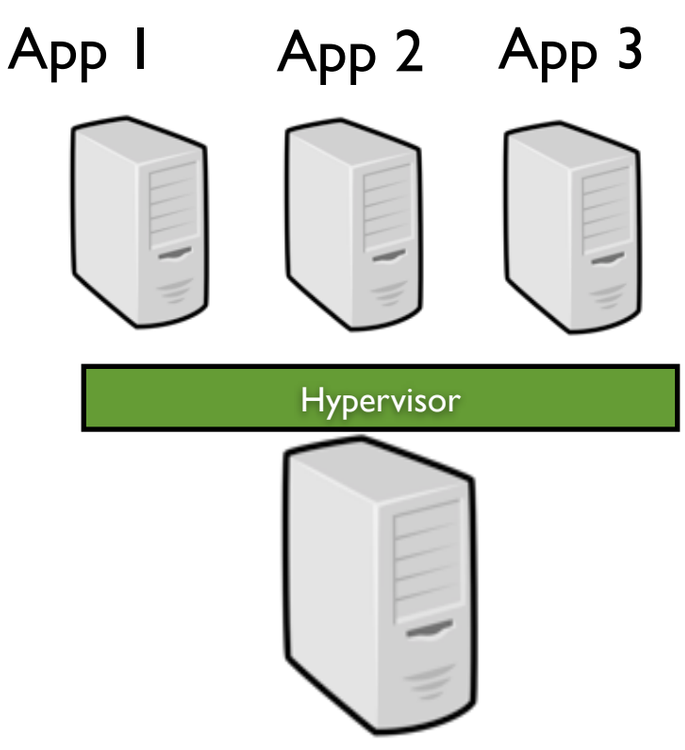
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

VM Capabilities

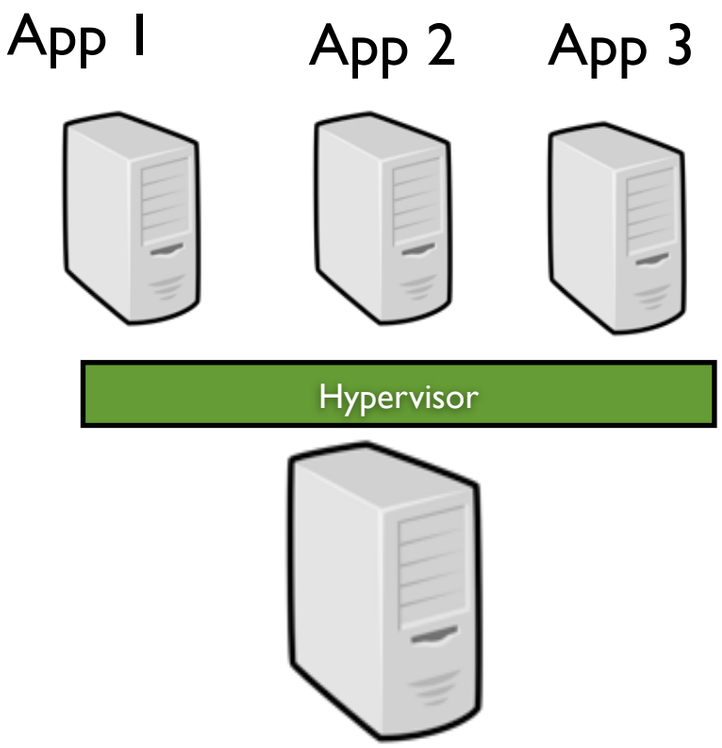


- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume

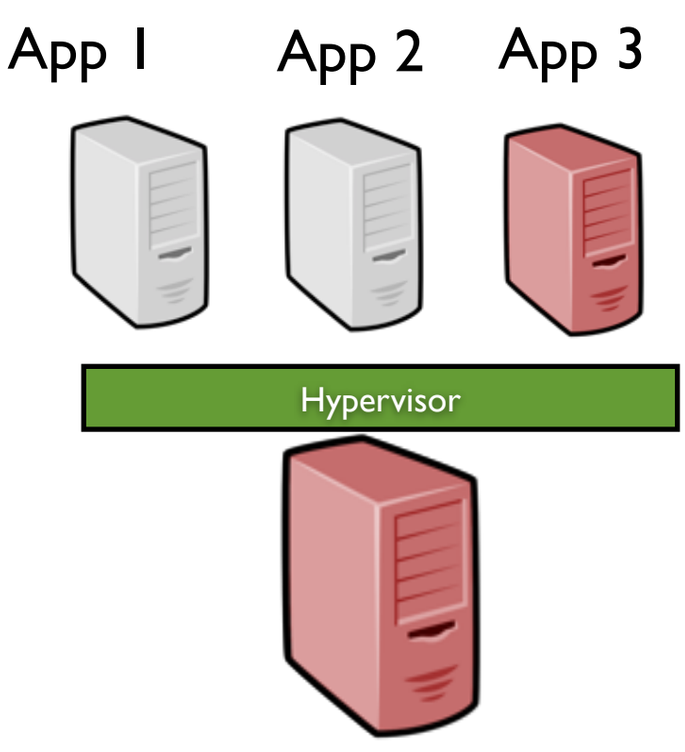


VM Capabilities

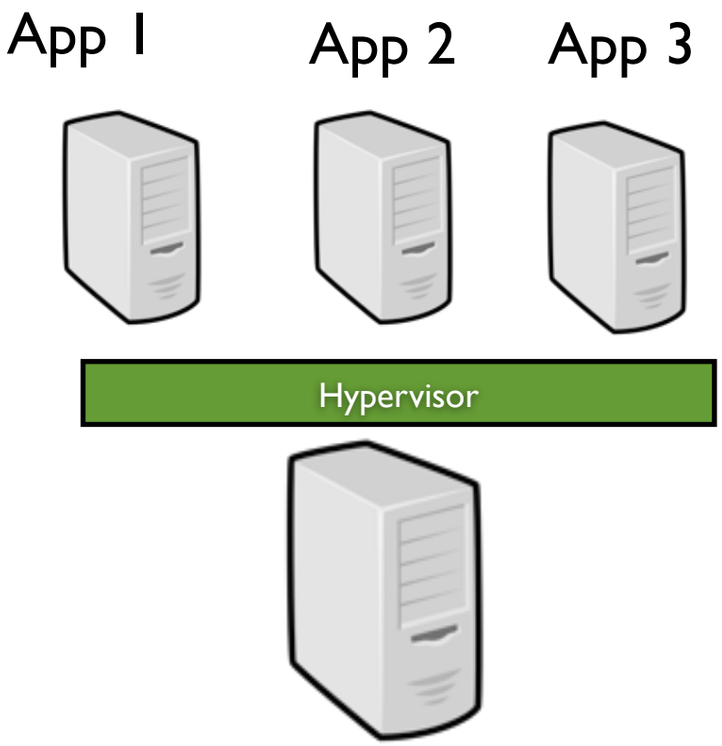


- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

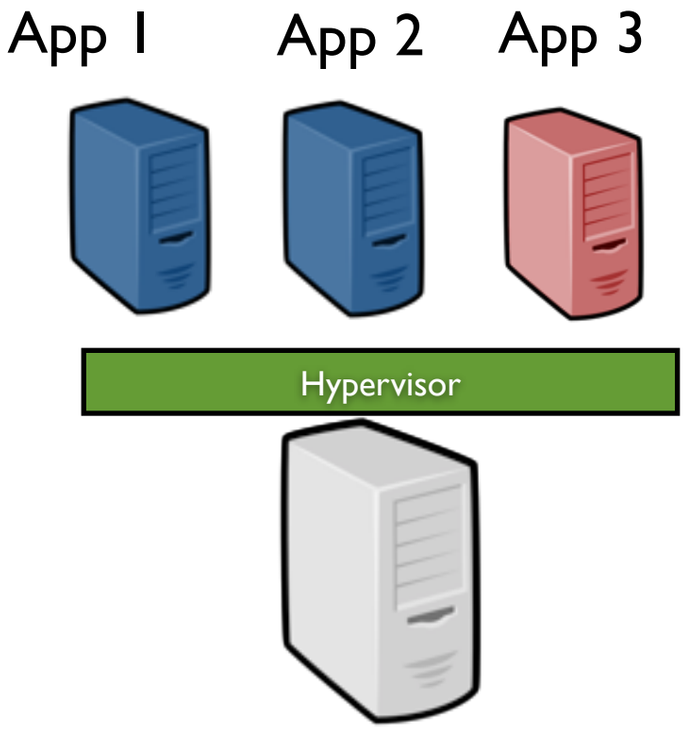
- Suspend/Resume



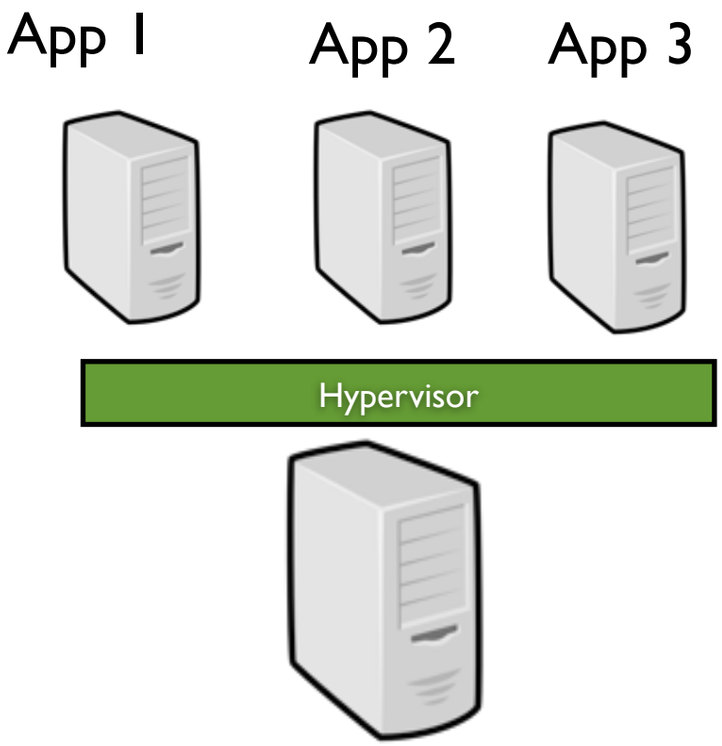
VM Capabilities



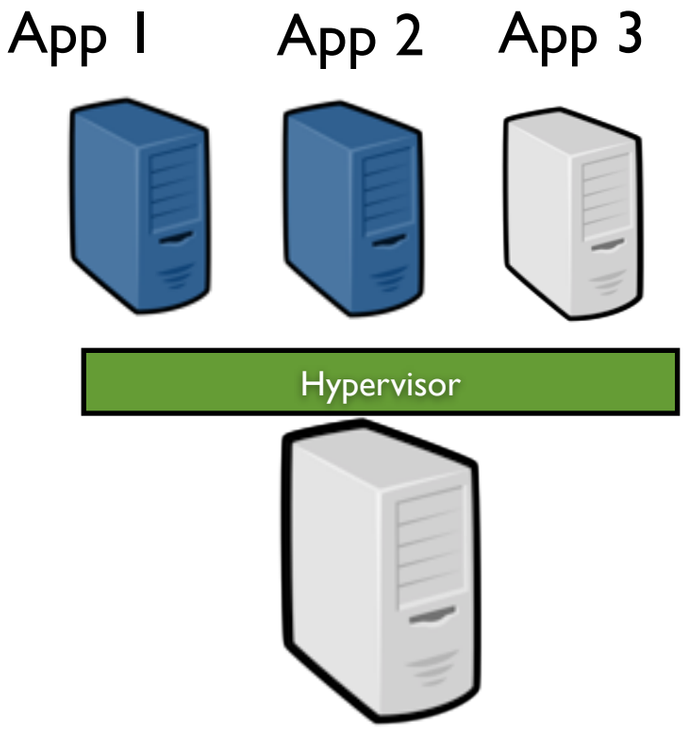
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)
- Suspend/Resume



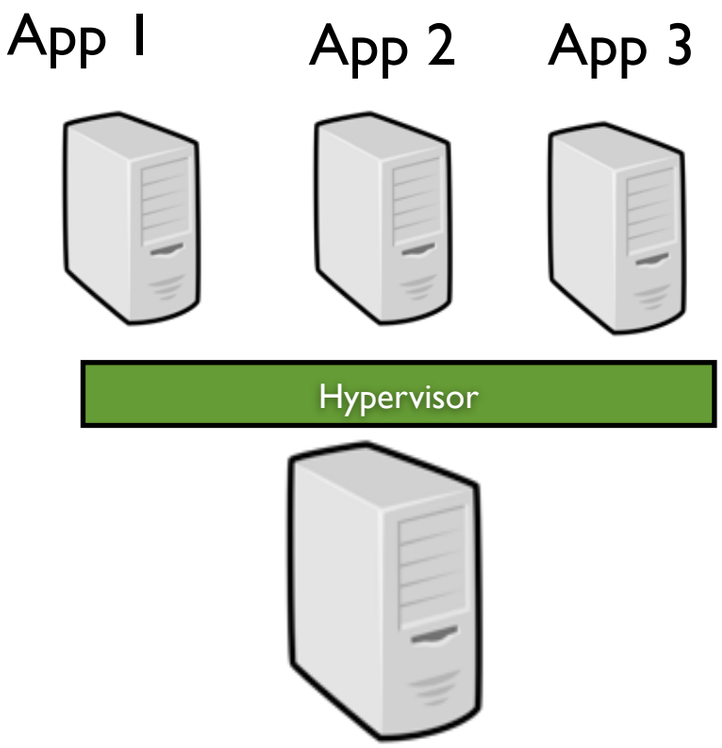
VM Capabilities



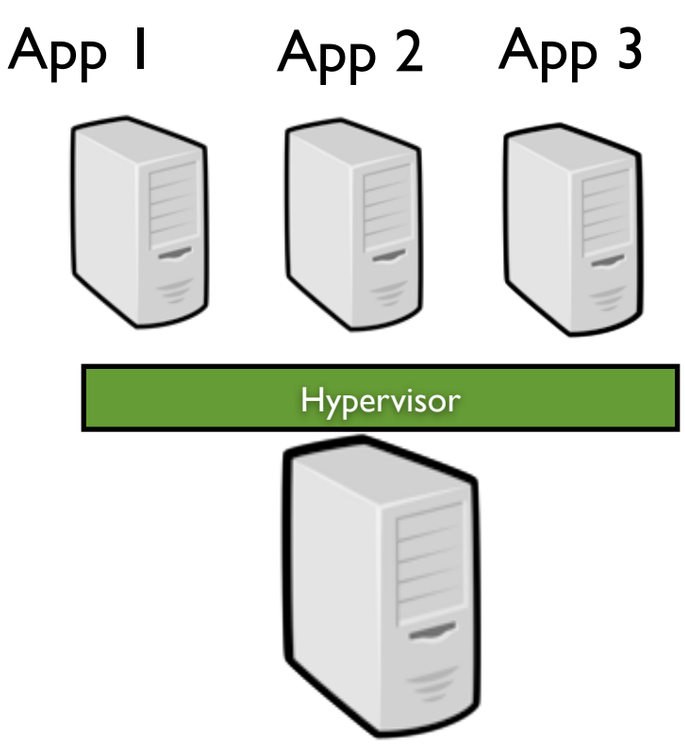
- Isolation (“security” between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)
- Suspend/Resume



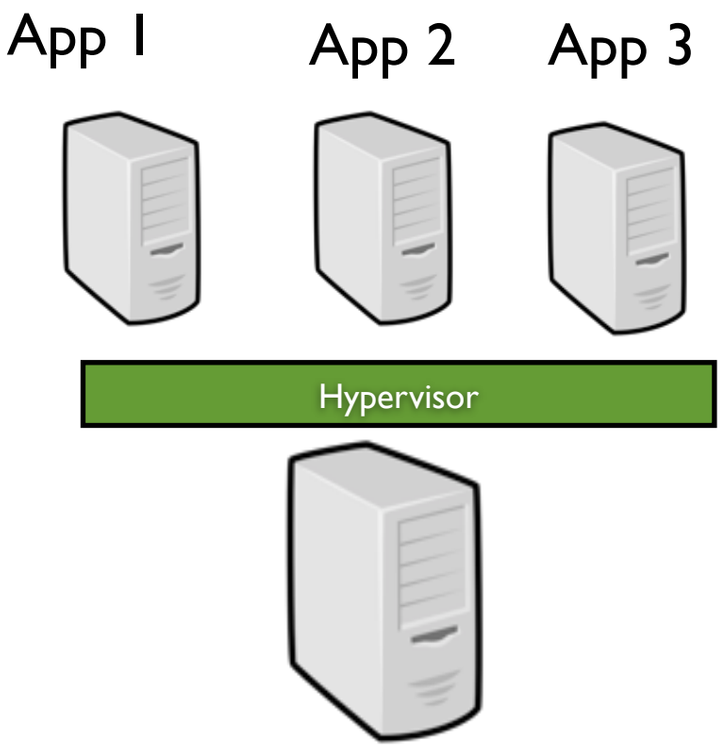
VM Capabilities



- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)
- Suspend/Resume

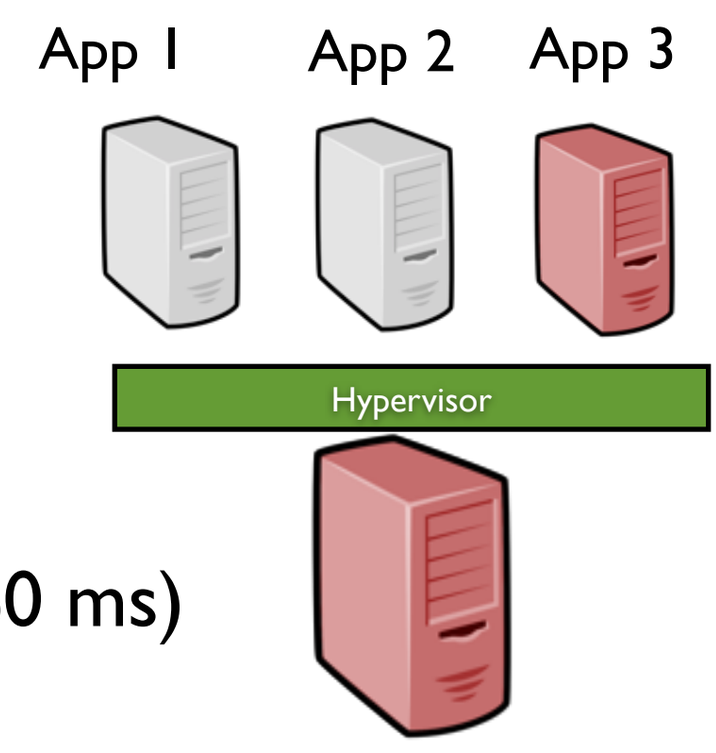


VM Capabilities

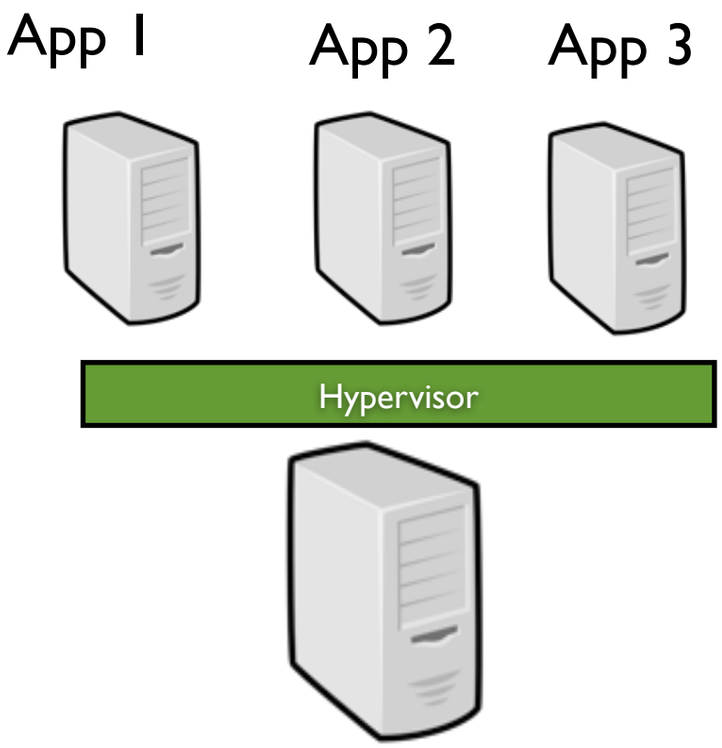


- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)
  Post/Pre Copy

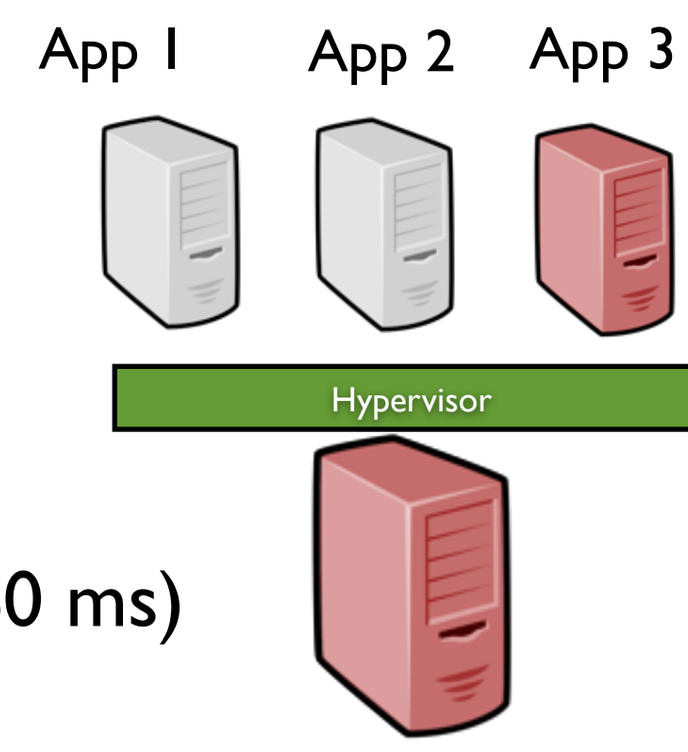


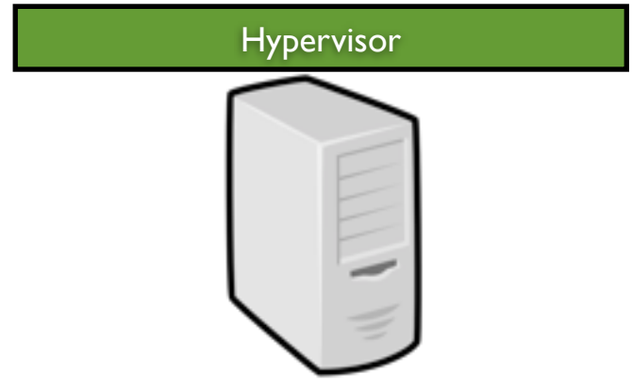
VM Capabilities



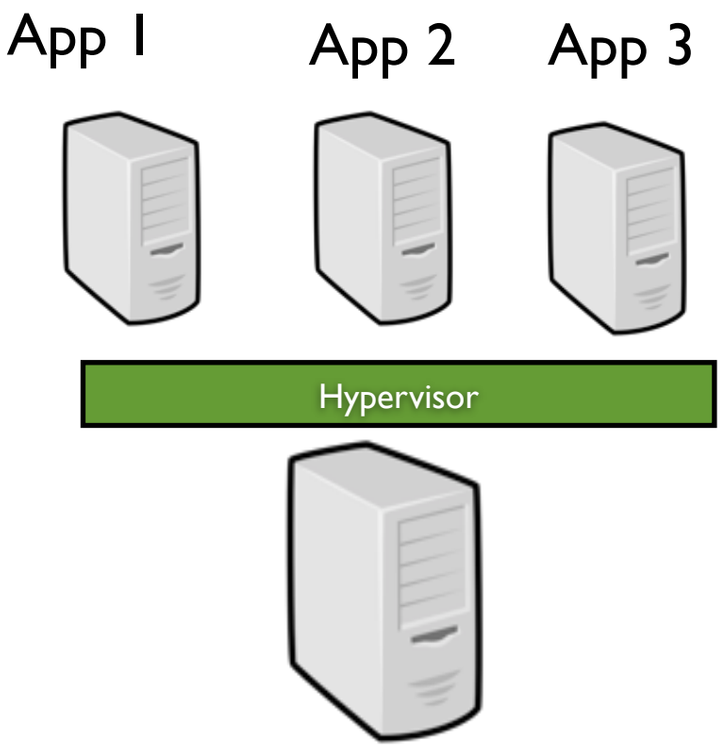
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)
  Post/Pre Copy



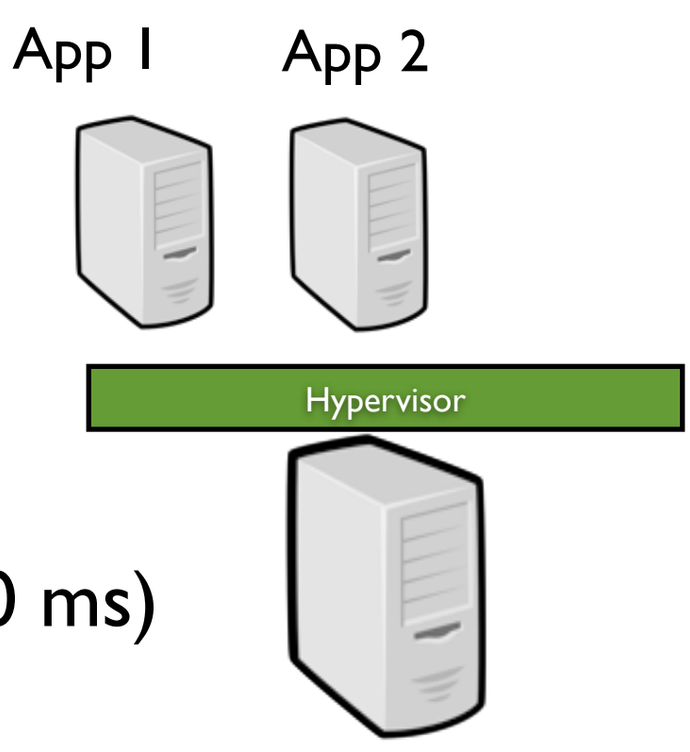


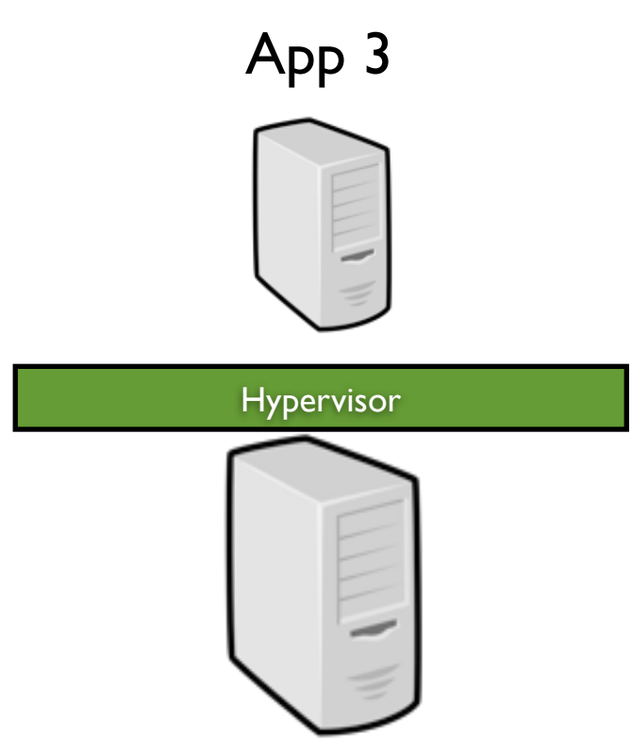
VM Capabilities



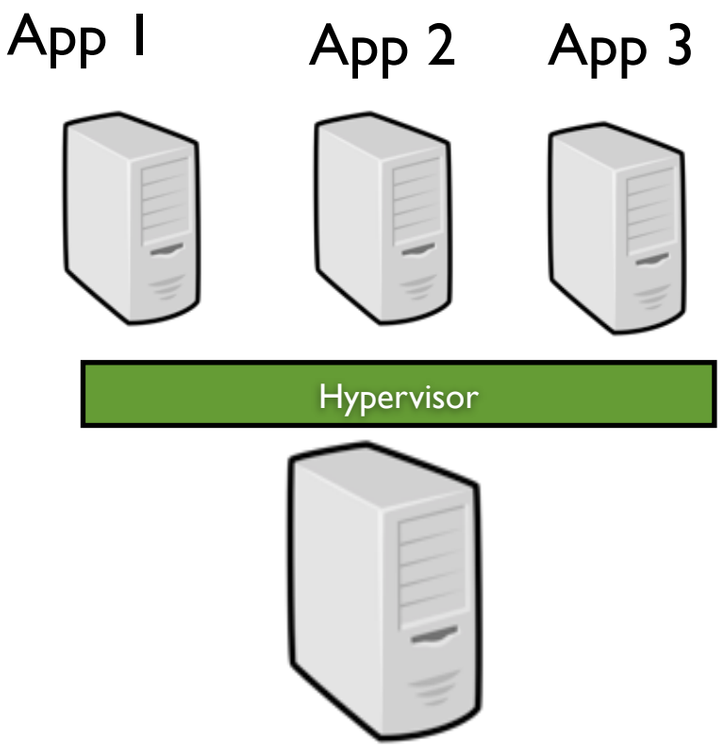
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)
  Post/Pre Copy



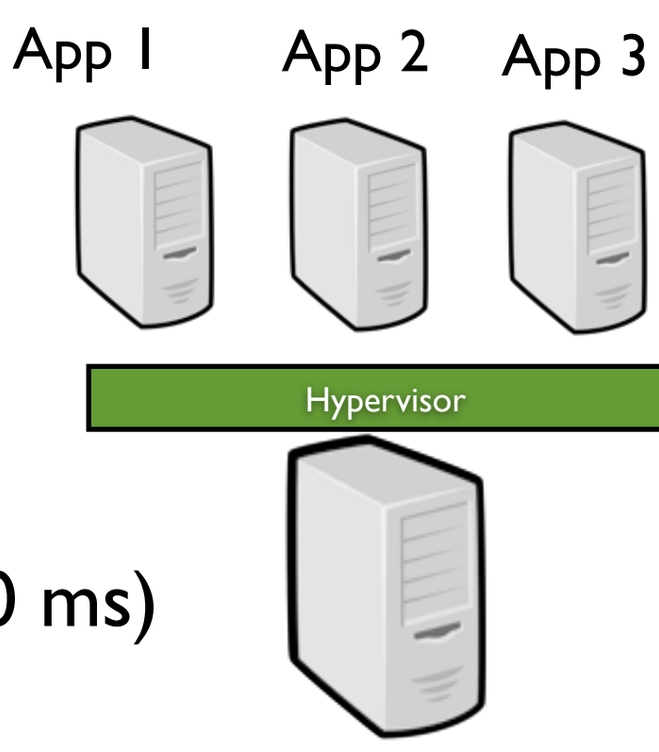


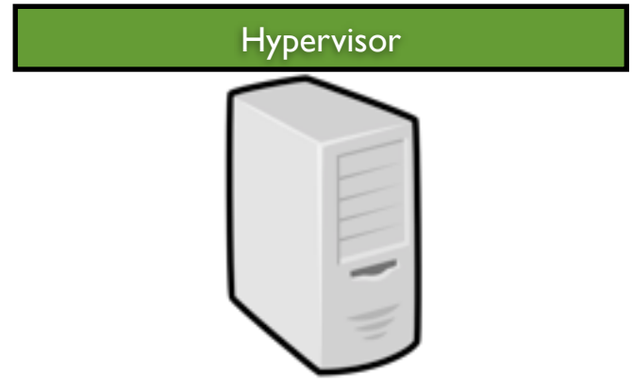
VM Capabilities



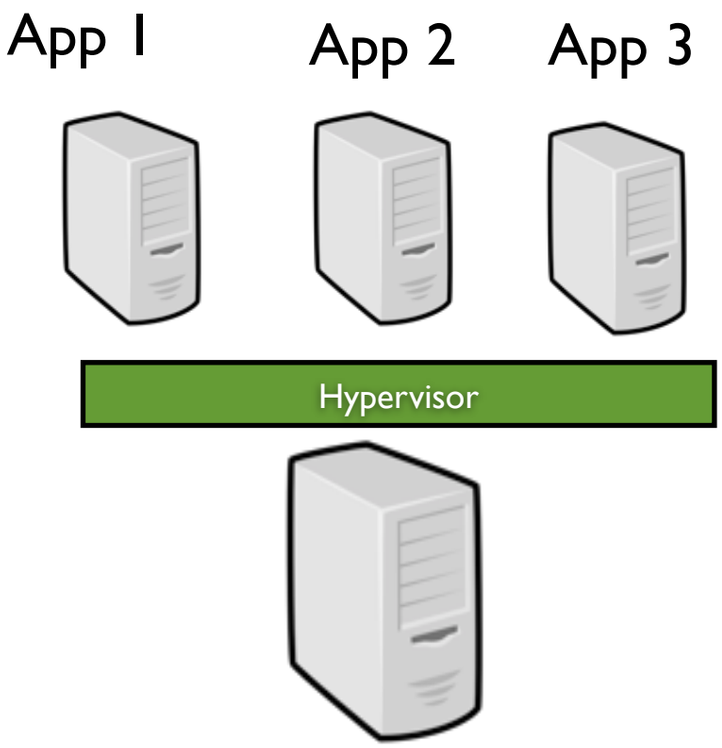
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)
  Post/Pre Copy



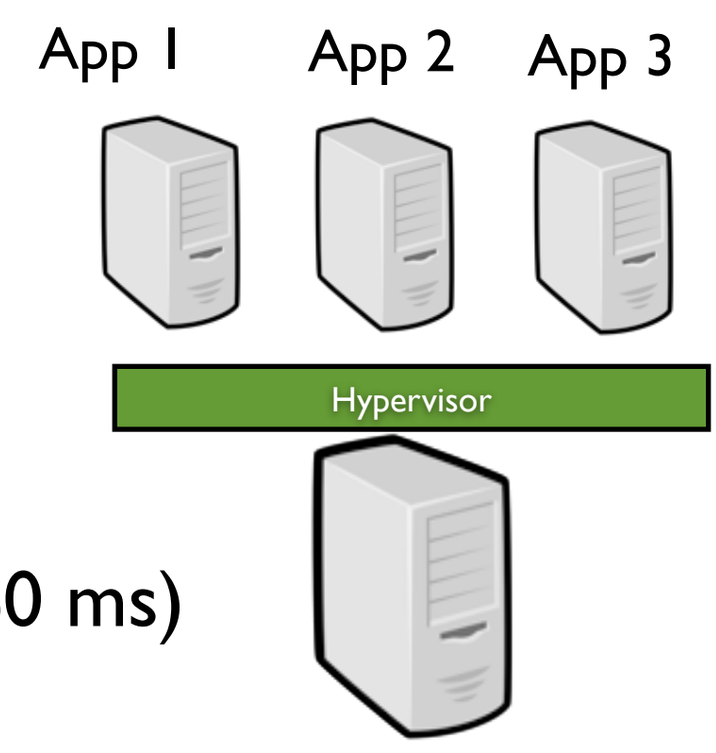


VM Capabilities



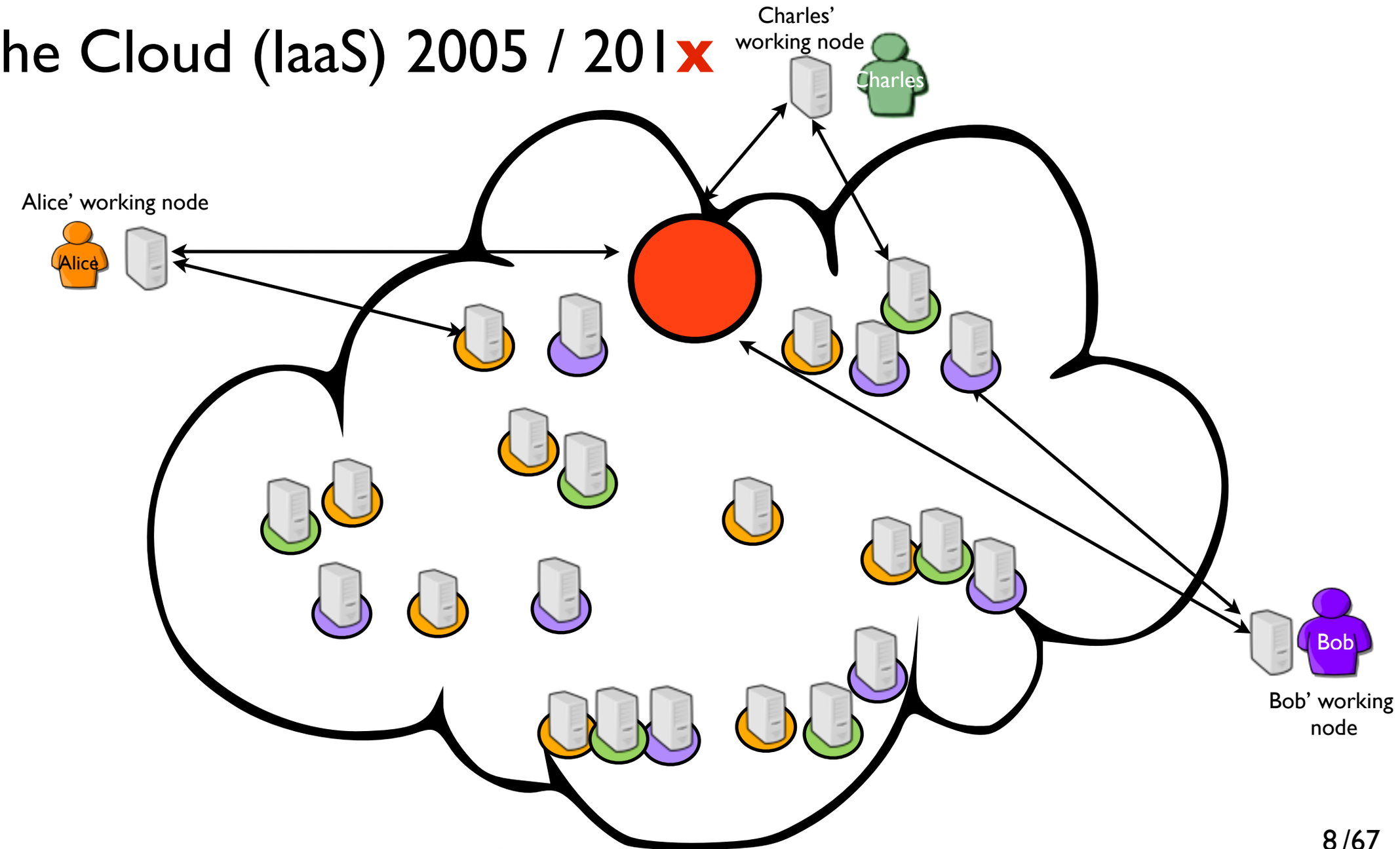
- Isolation ("security" between each VM)
- Snapshotting (a VM can be easily resumed from its latest consistent state)

- Suspend/Resume
- Live migration
  (negligible downtime ~ 60 ms)
  Post/Pre Copy



Utility Computing - Successive Generations

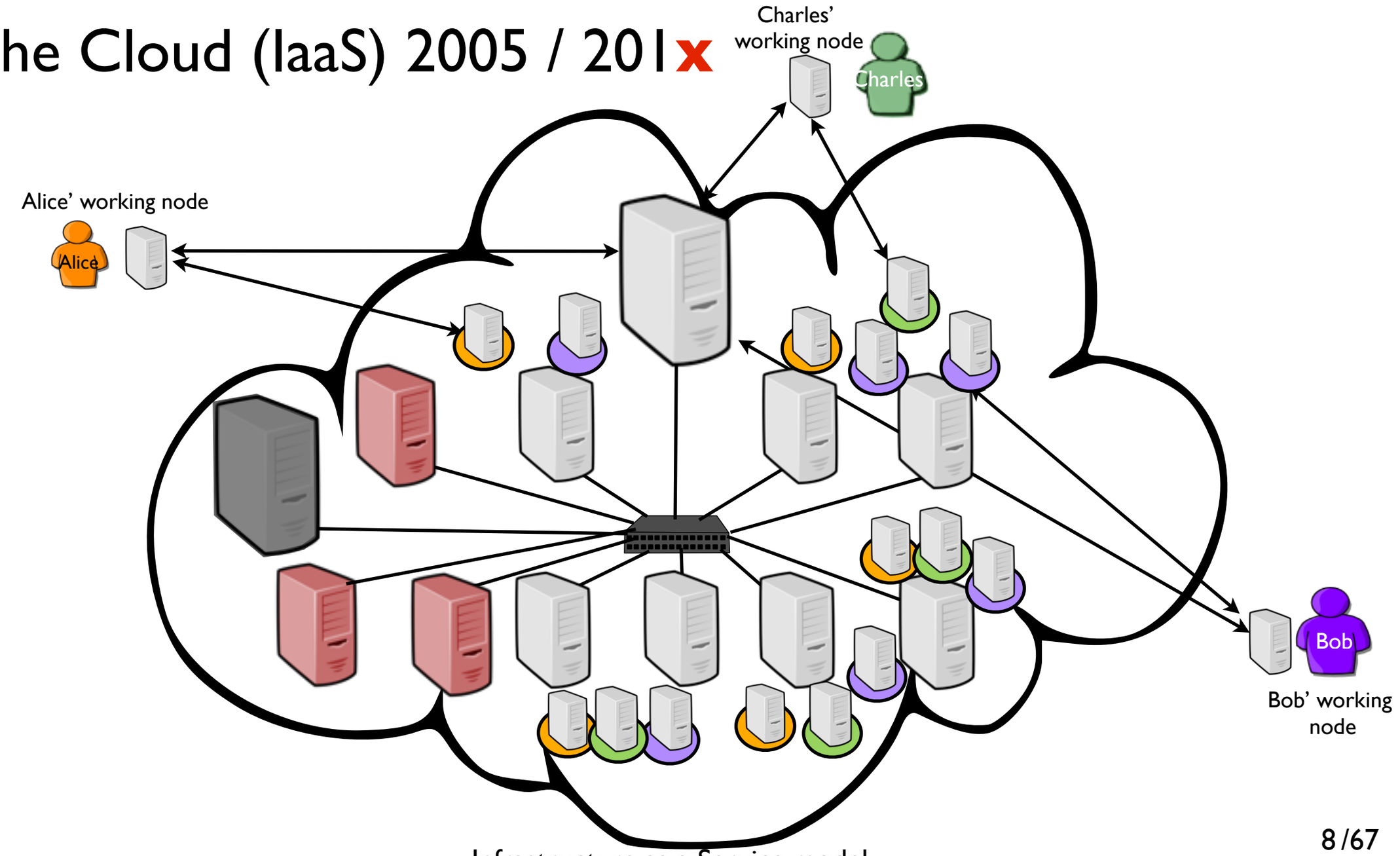
- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**
- The Cloud (IaaS) 2005 / 201**x**



Infrastructure as a Service model

Utility Computing - Successive Generations

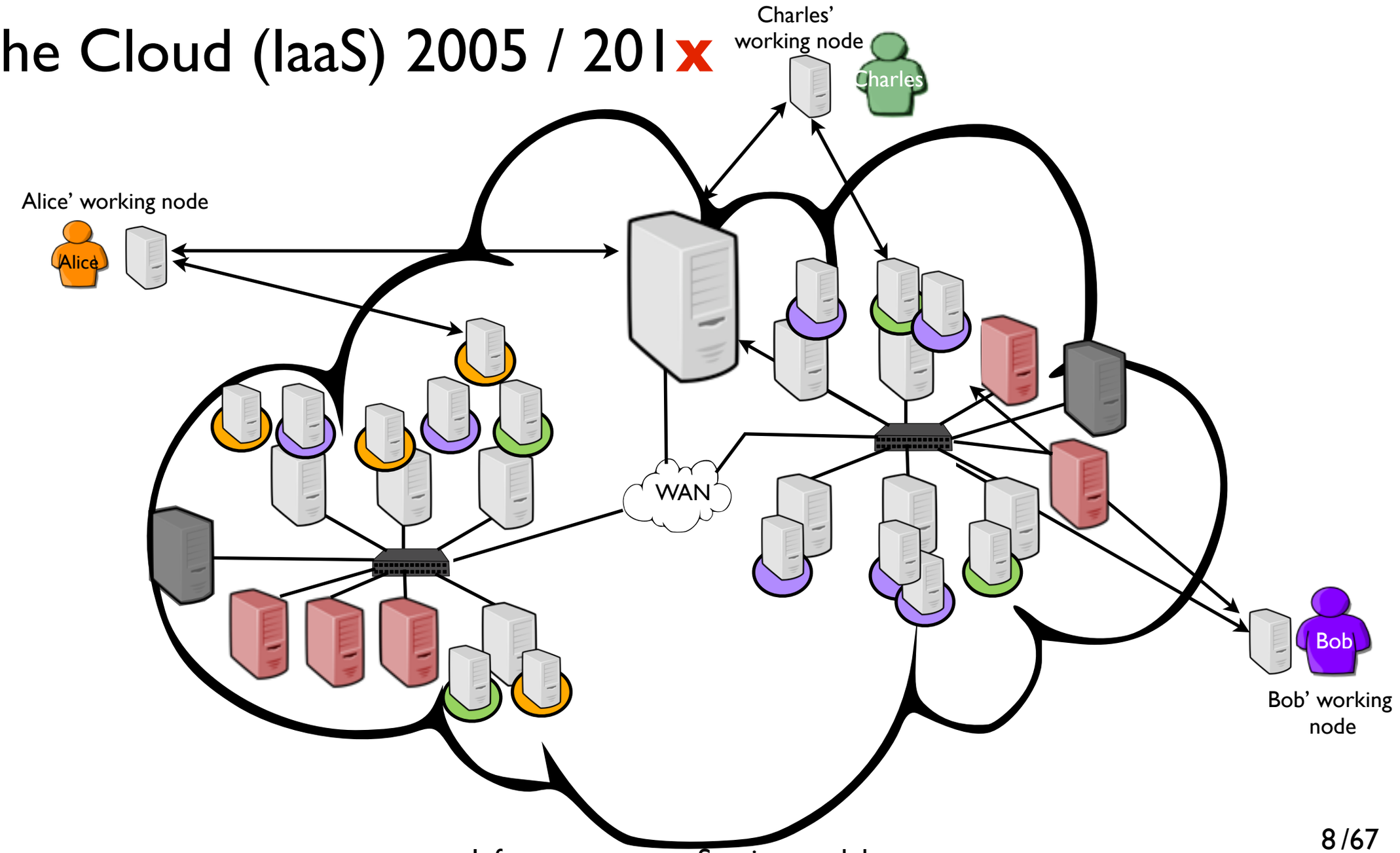
- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**
- The Cloud (IaaS) 2005 / 201**x**



Infrastructure as a Service model

Utility Computing - Successive Generations

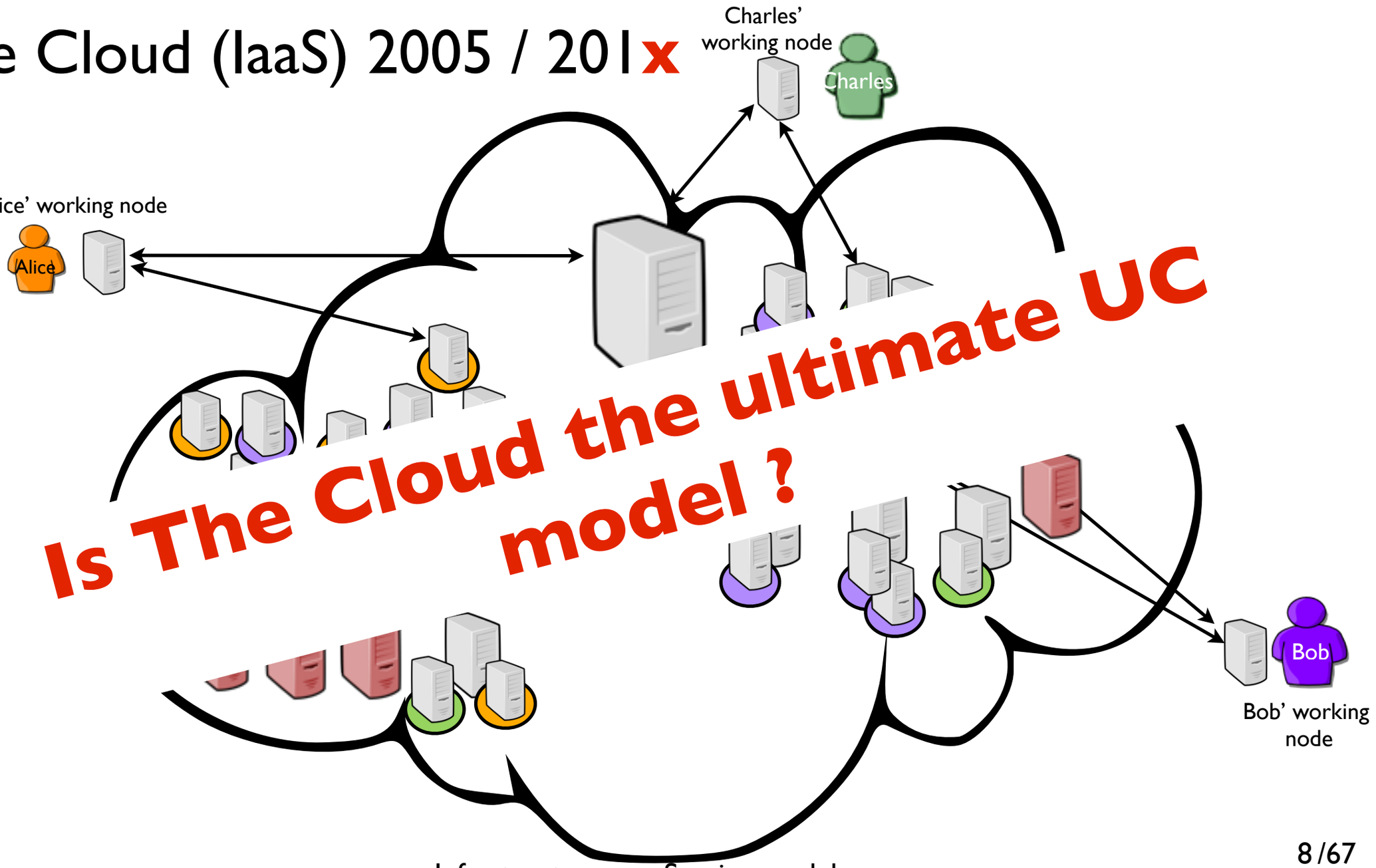
- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**
- The Cloud (IaaS) 2005 / 201**x**



Infrastructure as a Service model

Utility Computing - Successive Generations

- Network of Workstations 1990 / 20**xx**
- The Grid 1997 / 201**x**
- The Cloud (IaaS) 2005 / 201**x**



Is The Cloud the ultimate UC model ?

Operating IaaS Platforms
A state of the art
(in 5 slides)

Operating IaaS - CloudKit (*Cloud OS ?*)

- Proprietary proposals
- vCloud/vSphere (vmware) 60%

  ESXi

- XenServer/Xen Cloud platform (20%)

  Xen

- Microsoft System Center VM (20%)

  Hypervisors agnostic

Credits: http://www.v-index.com
Virtualization Industry Quaterly Survey

Operating IaaS - CloudKit (*Cloud OS ?*)

- Academic proposals

  Nimbus (Freeman and Keahey, University of Chicago)

  Based on GT4 and the Globus Virtual Workspace Service
  Target: cloud for science
  Tutorials and documentation in "grid space"

  Open Nebula (Montero & Llorente, DSA-Research at UCM)

  Support for the Xen, KVM and VMware
  Access to Amazon EC2 (cloud bursting)
  Probably, the most deployed in EU (2012)

  Eucalyptus (Wolsky, University of Santa Barbara)

  Web services based implementation of elastic/utility/cloud computing infrastructure

Operating IaaS - CloudKit (*Cloud OS ?*)

- Community proposals

  OpenStack

  - Supported by several industrials
  - Successor of OpenNebula for the core of the Ubuntu cloud proposal

  CloudStack

  - Supported by CITRIX
  - Full JAVA implementation
  - Apache project

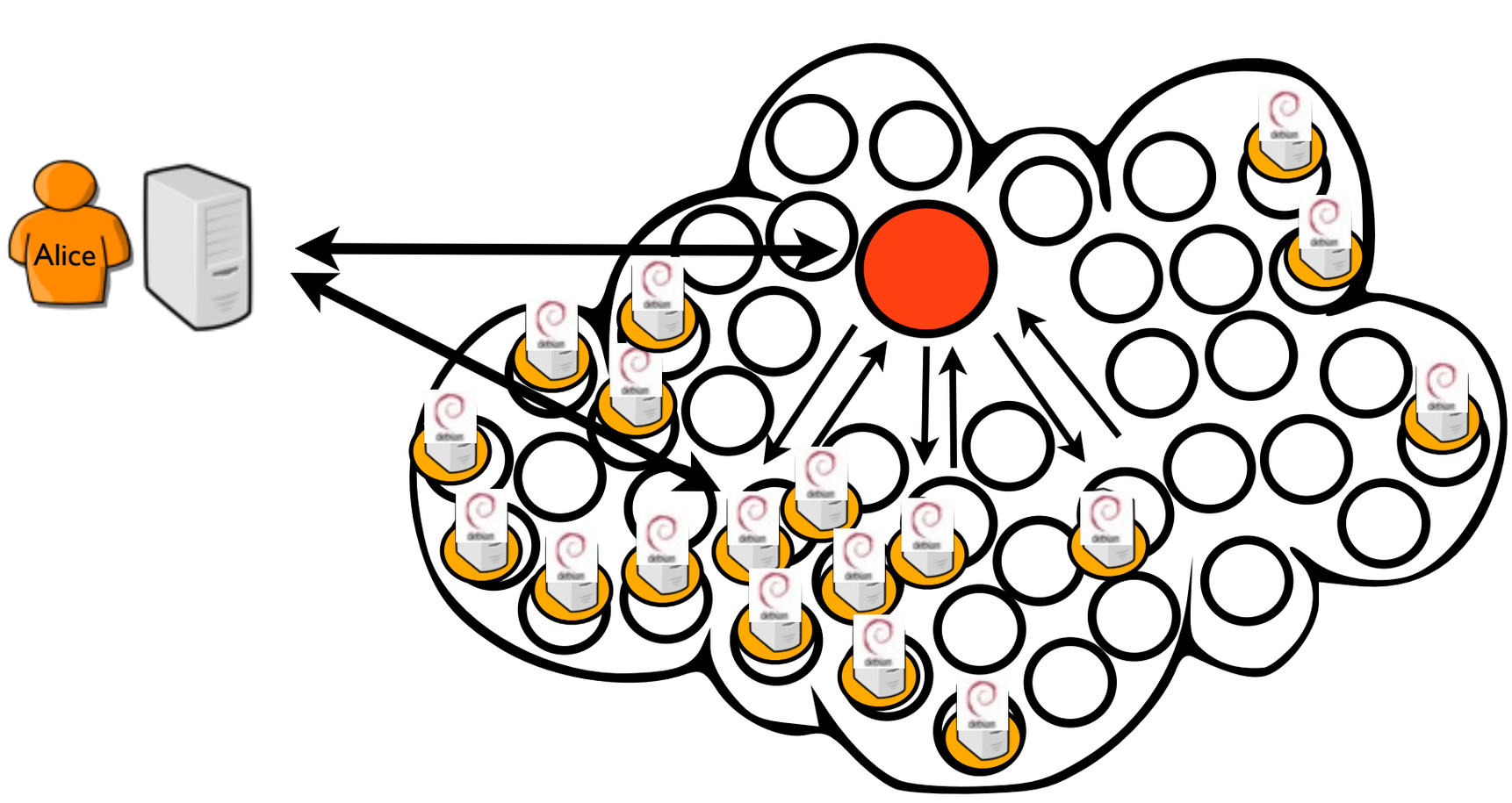
Putting everything into the cloud

- Mature for one *centralized* site !

  More flexibility ! ? Infinite resources ! ?

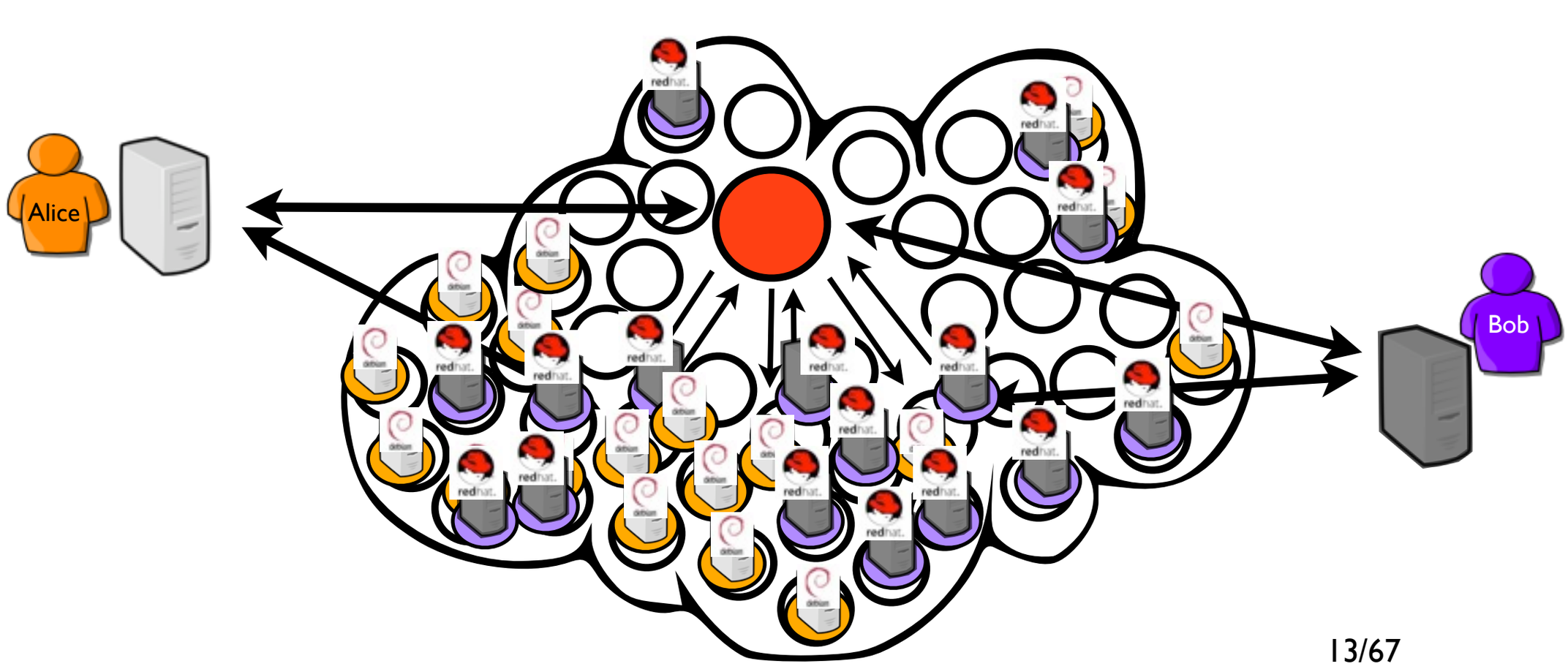
- New concerns !?

  Scalability (VM Sprawl)



Putting everything into the cloud

- Mature for one *centralized* site !
  - More flexibility ! ? Infinite resources ! ?
- New concerns !?
  - Scalability (VM Sprawl)



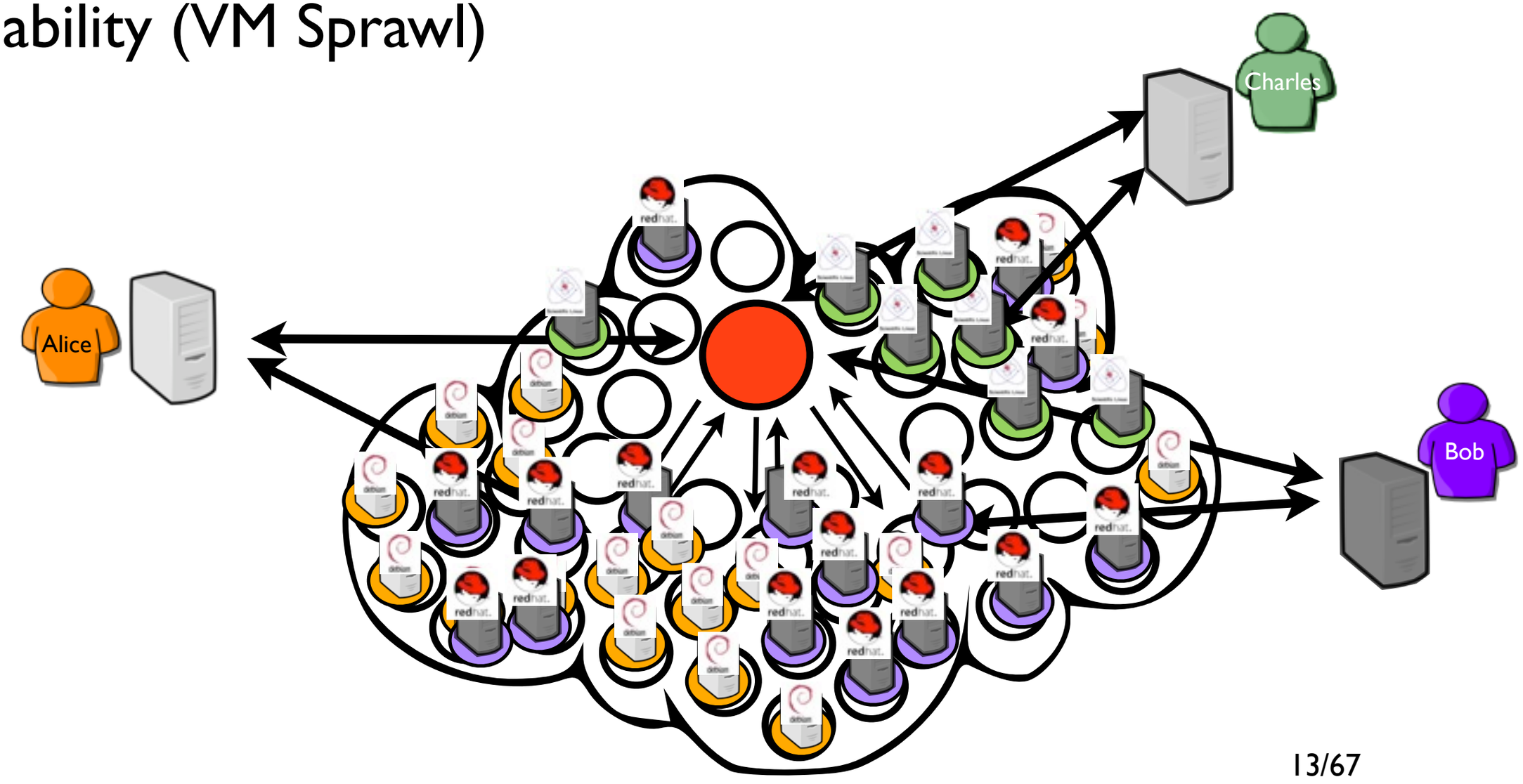
Putting everything into the cloud

- Mature for one *centralized* site !

  More flexibility ! ? Infinite resources ! ?

- New concerns !?

  Scalability (VM Sprawl)



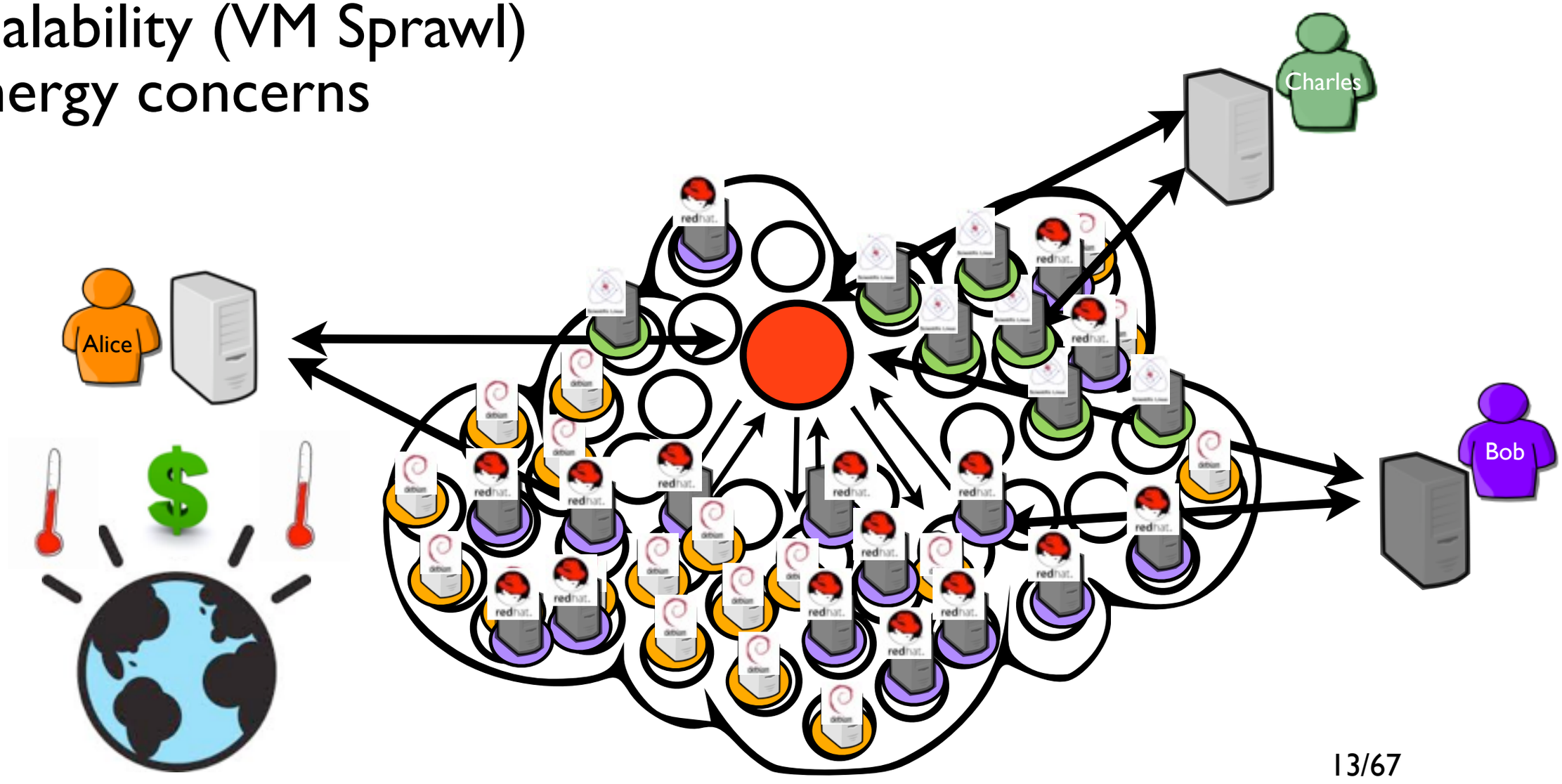
Putting everything into the cloud

- Mature for one *centralized* site !

  More flexibility ! ? Infinite resources ! ?

- New concerns !?

  Scalability (VM Sprawl)
  Energy concerns



Putting everything into the cloud

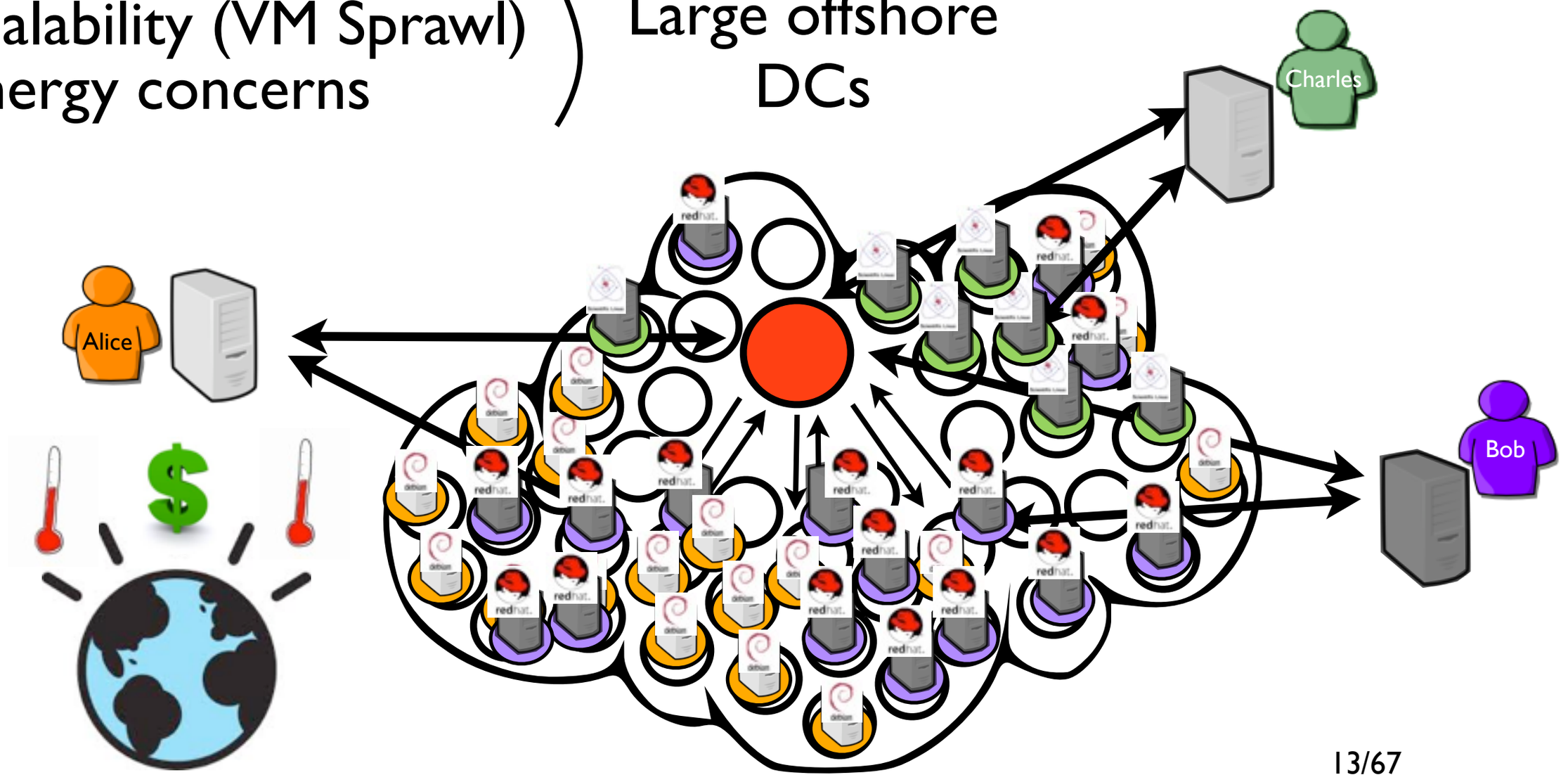
- Mature for one *centralized* site !

  More flexibility ! ? Infinite resources ! ?

- New concerns !?

  Scalability (VM Sprawl)
  Energy concerns

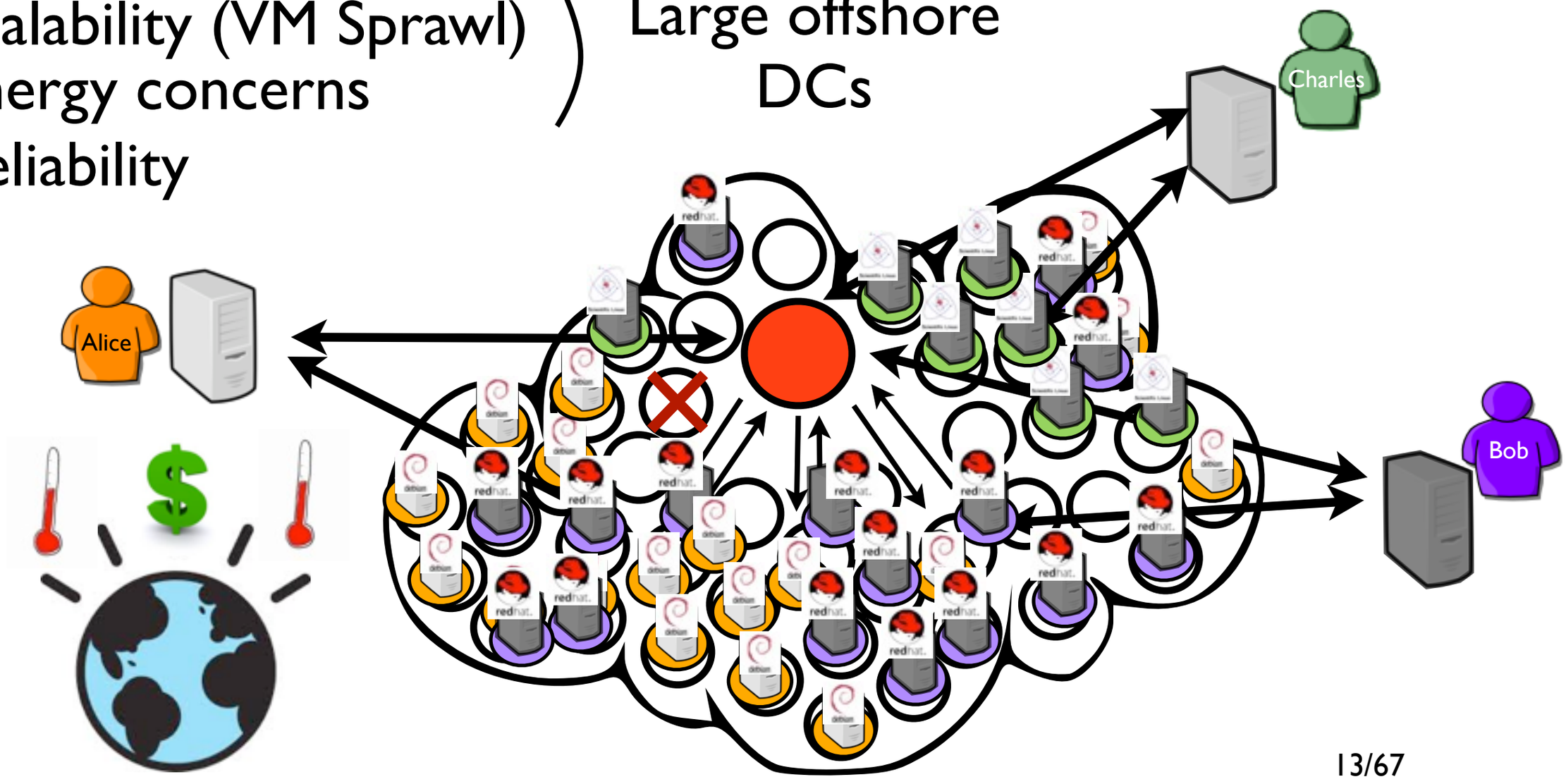
  ⟩ Large offshore DCs



Putting everything into the cloud

- Mature for one *centralized* site !
  - More flexibility ! ? Infinite resources ! ?
- New concerns !?
  - Scalability (VM Sprawl)
  - Energy concerns
  - Reliability

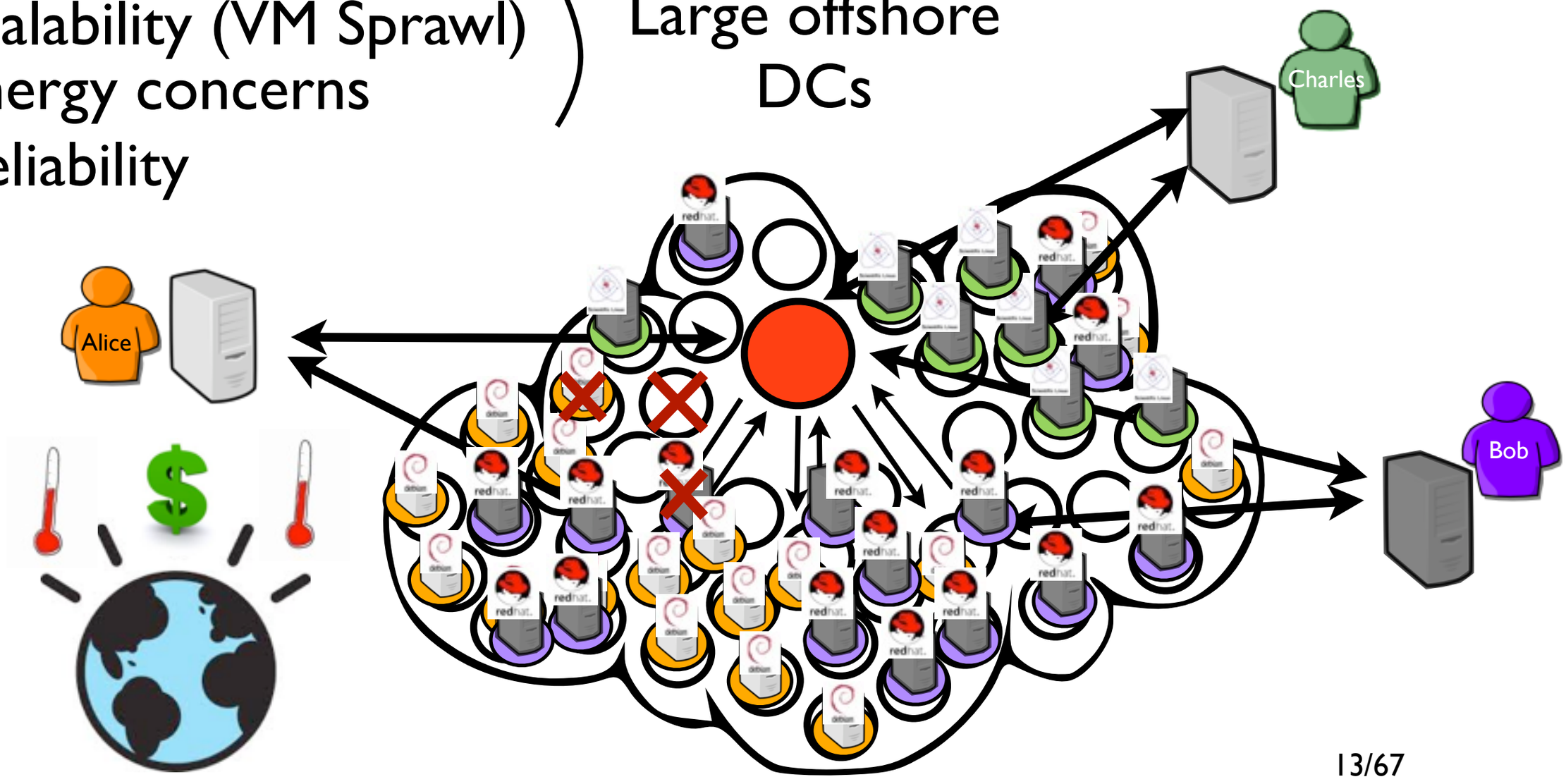
⟩ Large offshore DCs



Putting everything into the cloud

- Mature for one *centralized* site !
  - More flexibility ! ? Infinite resources ! ?
- New concerns !?
  - Scalability (VM Sprawl)
  - Energy concerns
  - Reliability

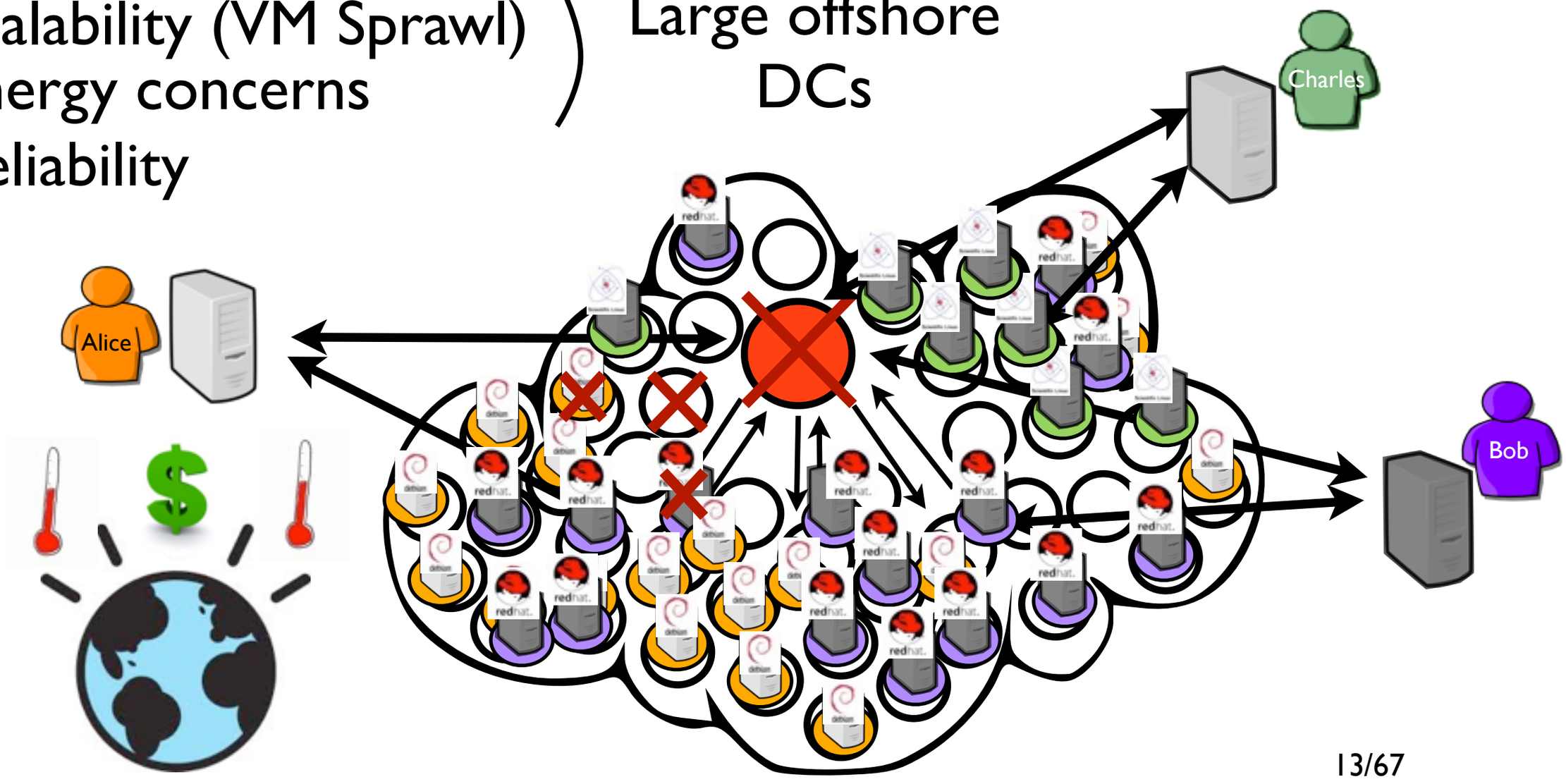
⟩ Large offshore DCs



Putting everything into the cloud

- Mature for one *centralized* site !
  - More flexibility ! ? Infinite resources ! ?
- New concerns !?
  - Scalability (VM Sprawl)
  - Energy concerns
  - Reliability

⟩ Large offshore DCs



Putting everything into the cloud

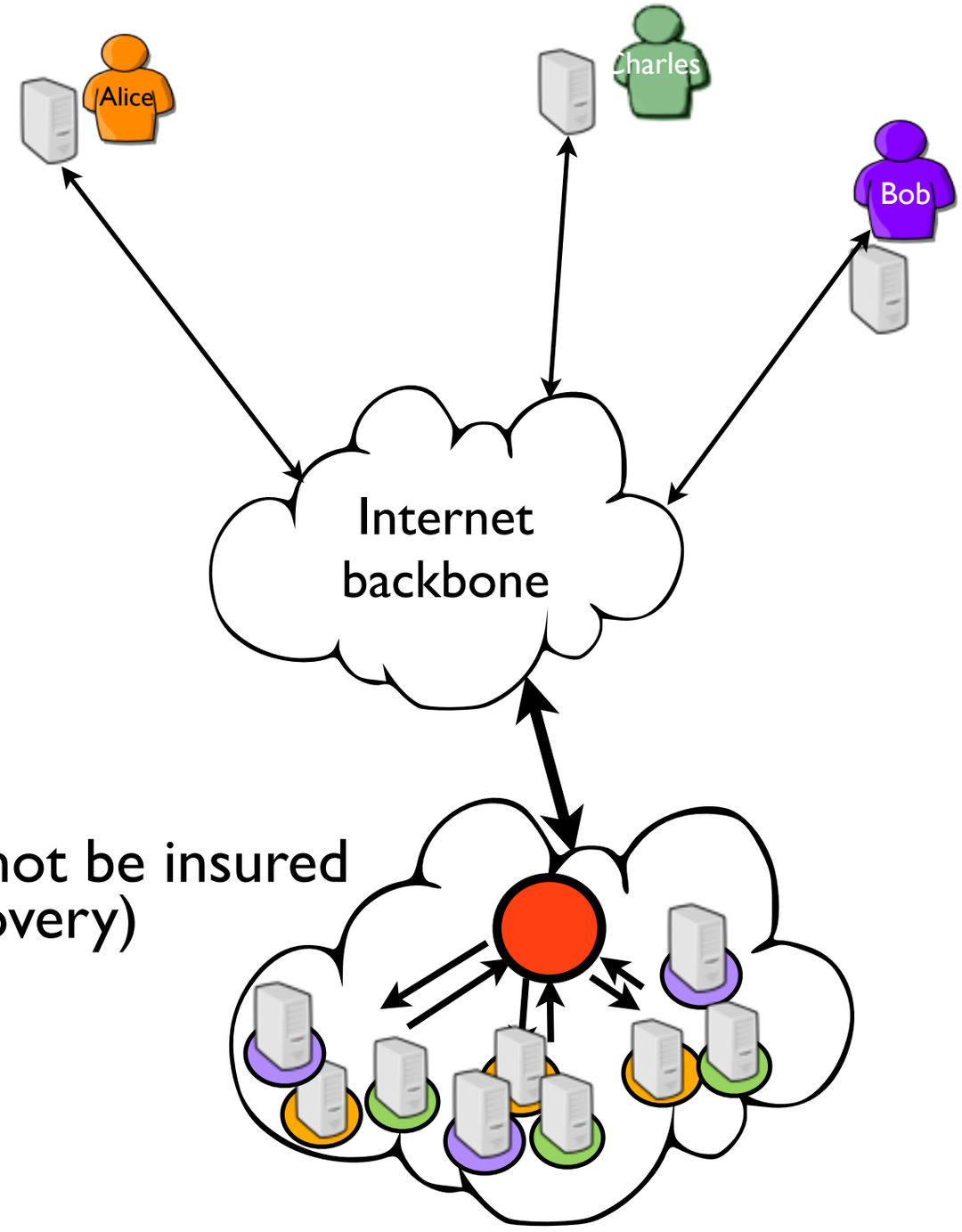
- Inherent limitations of the cloud computing model w.r.t public offers (or why building large offshore DCs is not appropriated).

1. Externalization of private applications/ data (jurisdiction concerns)

2. Overhead implied by the unavoidable use of the Internet to reach distant platforms

3. The connectivity to the application/data cannot be insured by centralized dedicated centers (disaster recovery)

4. Energy concerns (footprint but also physical limitations)



Putting everything into the cloud

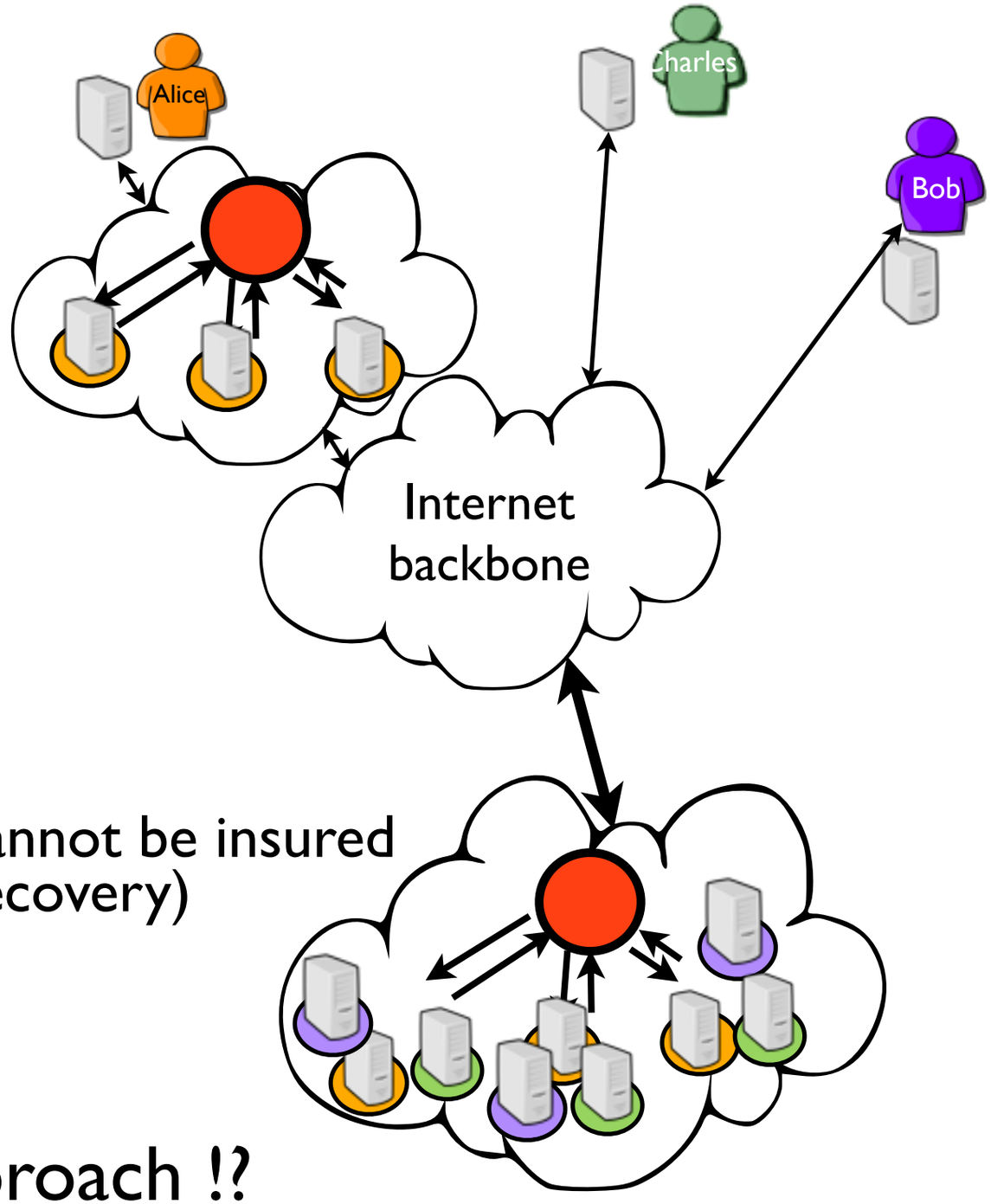
- Inherent limitations of the cloud computing model w.r.t public offers (or why building large offshore DCs is not appropriated).

  1. Externalization of private applications/ data (jurisdiction concerns)

  2. Overhead implied by the unavoidable use of the Internet to reach distant platforms

  3. The connectivity to the application/data cannot be insured by centralized dedicated centers (disaster recovery)

  4. Energy concerns (footprint but also physical limitations)



- Hybrid platforms: a promising approach !?

Putting everything into the cloud

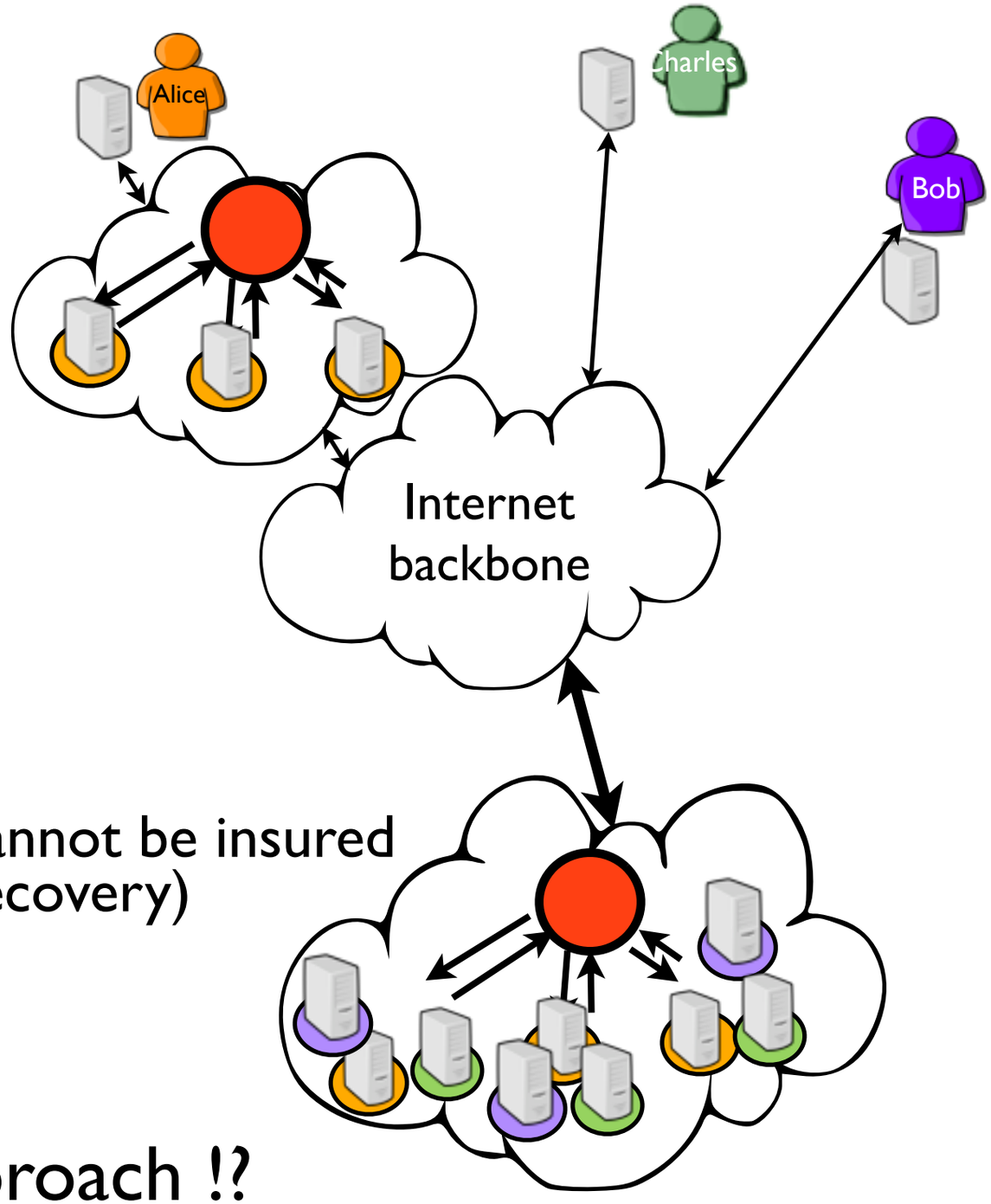
- Inherent limitations of the cloud computing model w.r.t public offers (or why building large offshore DCs is not appropriated).

  1. Externalization of private applications/ data (jurisdiction concerns)

  2. Overhead implied by the unavoidable use of the Internet to reach distant platforms

  3. The connectivity to the application/data cannot be insured by centralized dedicated centers (disaster recovery)

  4. Energy concerns (footprint but also physical limitations)



- Hybrid platforms: a promising approach !?

  Not really (points 1 and 2 still persist)

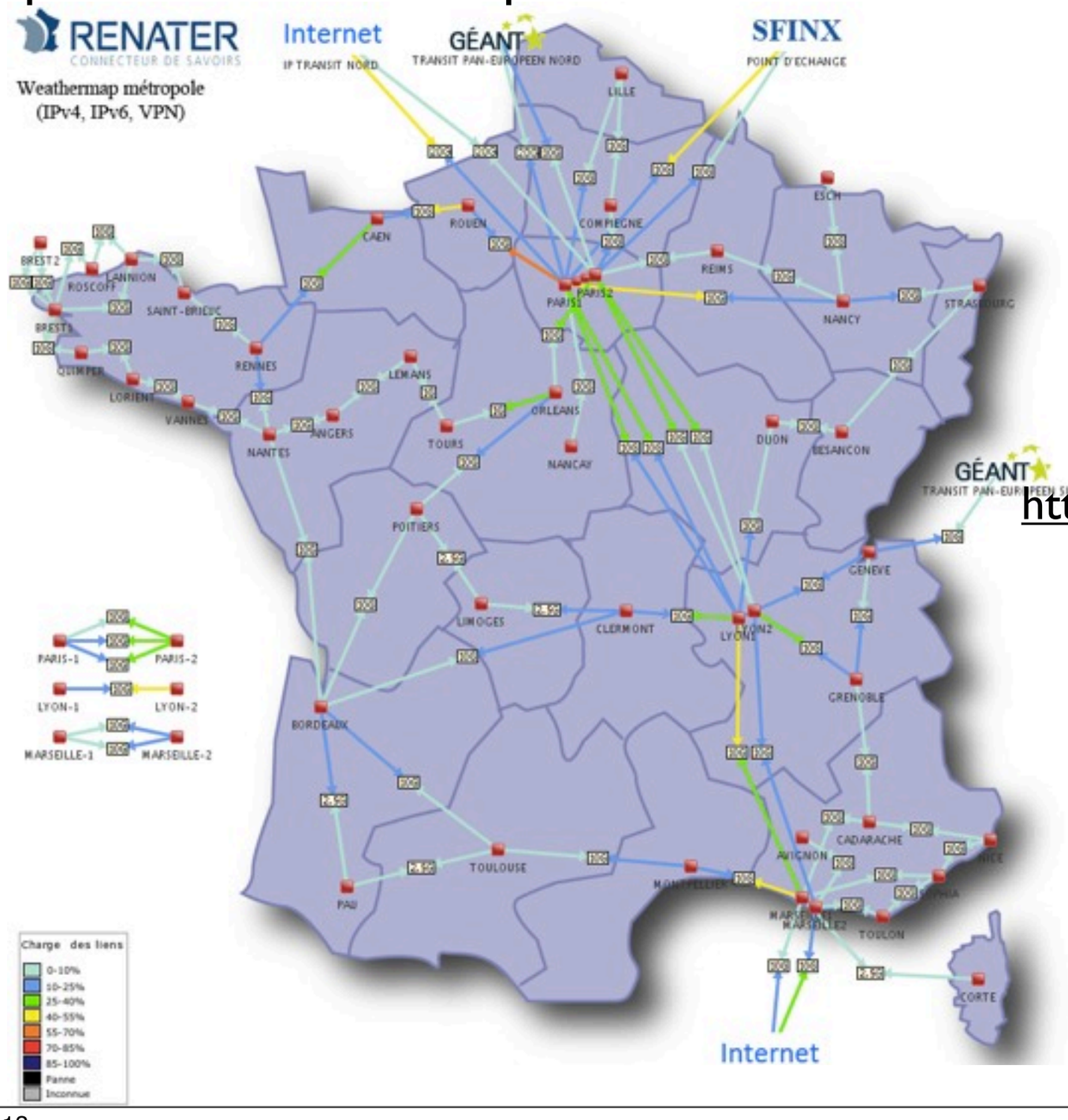
Can we address these concerns “all in one” ? ?

Locality Based Utility Computing
Toward LUC Infrastructures

Beyond the Cloud, the DISCOVERY Initiative

- Locality-based UC infrastructures

  The only way to deliver highly efficient and sustainable UC services is to provide UC platforms as close as possible to the end-users.

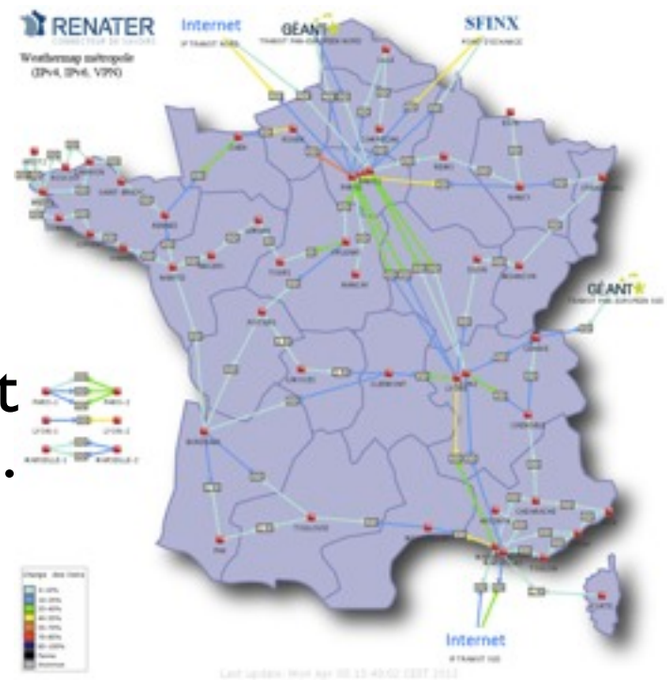


http://www.renater.fr/raccourci?lang=fr

Beyond the Cloud, the DISCOVERY Initiative

- Locality-based UC infrastructures

  The only way to deliver highly efficient and sustainable UC services is to provide UC platforms as close as possible to the end-users.



- Leveraging network backbones

  Extend any point of presence of a network backbone with UC servers (from major network hubs up to DSLAMs that are operated by telecom companies/network institutions).

- Leveraging the data furnaces concept

  Deploy UC servers in medium and large institutions and use them as sources of heat inside public buildings such as hospitals or universities

- Combining both approaches ! ?

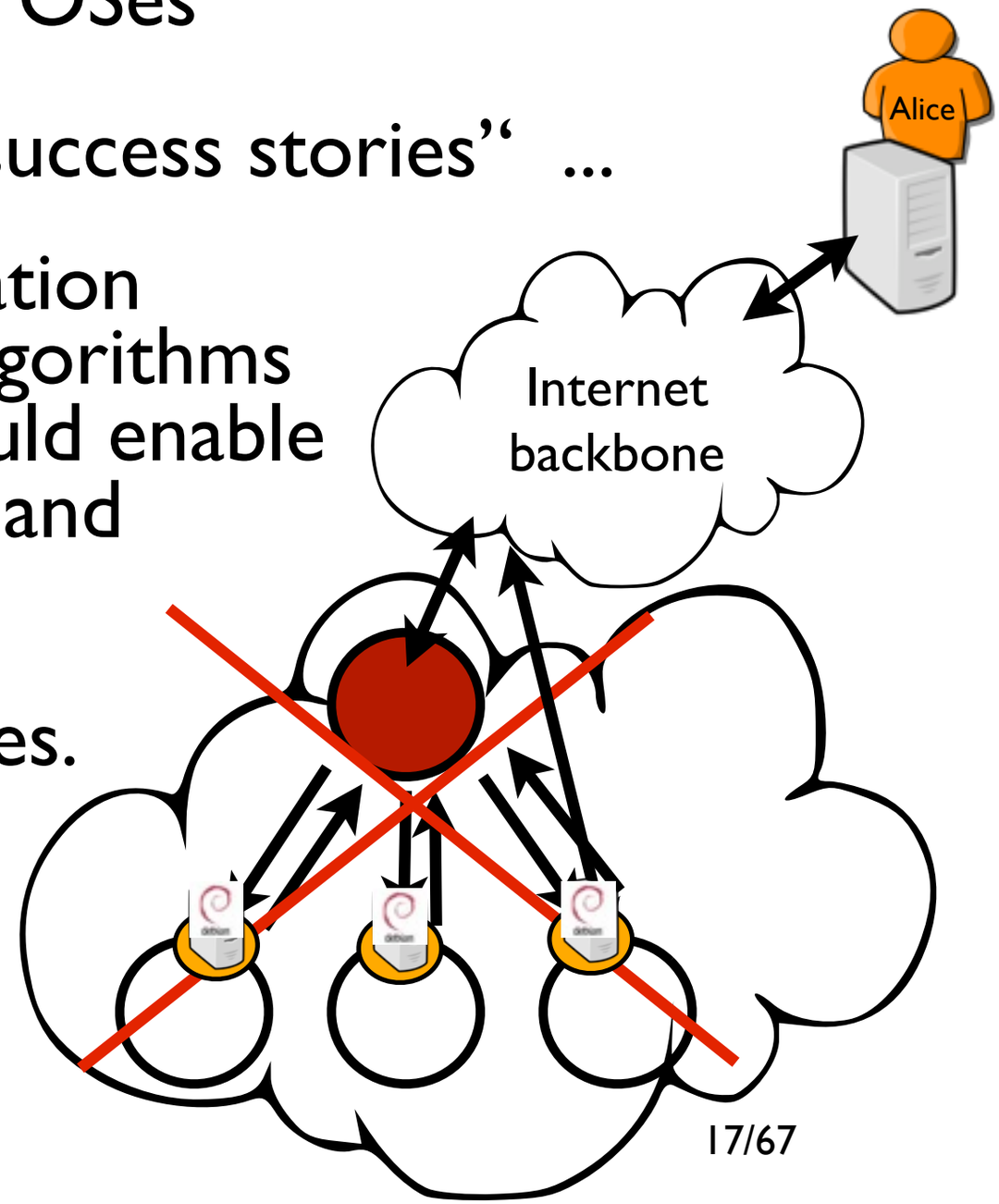
⇒ Operating such widely distributed resources requires the definition of a fully distributed system

The DISCOVERY Proposal

- DIStributed and COoperative framework to manage Virtual EnviRonments autonomicallY
- Designing/implementing Distributed OSes

Deeply investigated with no "real success stories" ...

... But maturity of system virtualization capabilities as well as large scale algorithms and autonomous mechanisms should enable to design and implement a unified and autonomic system manipulating virtual environments (VEs) like traditional OS manipulate processes.

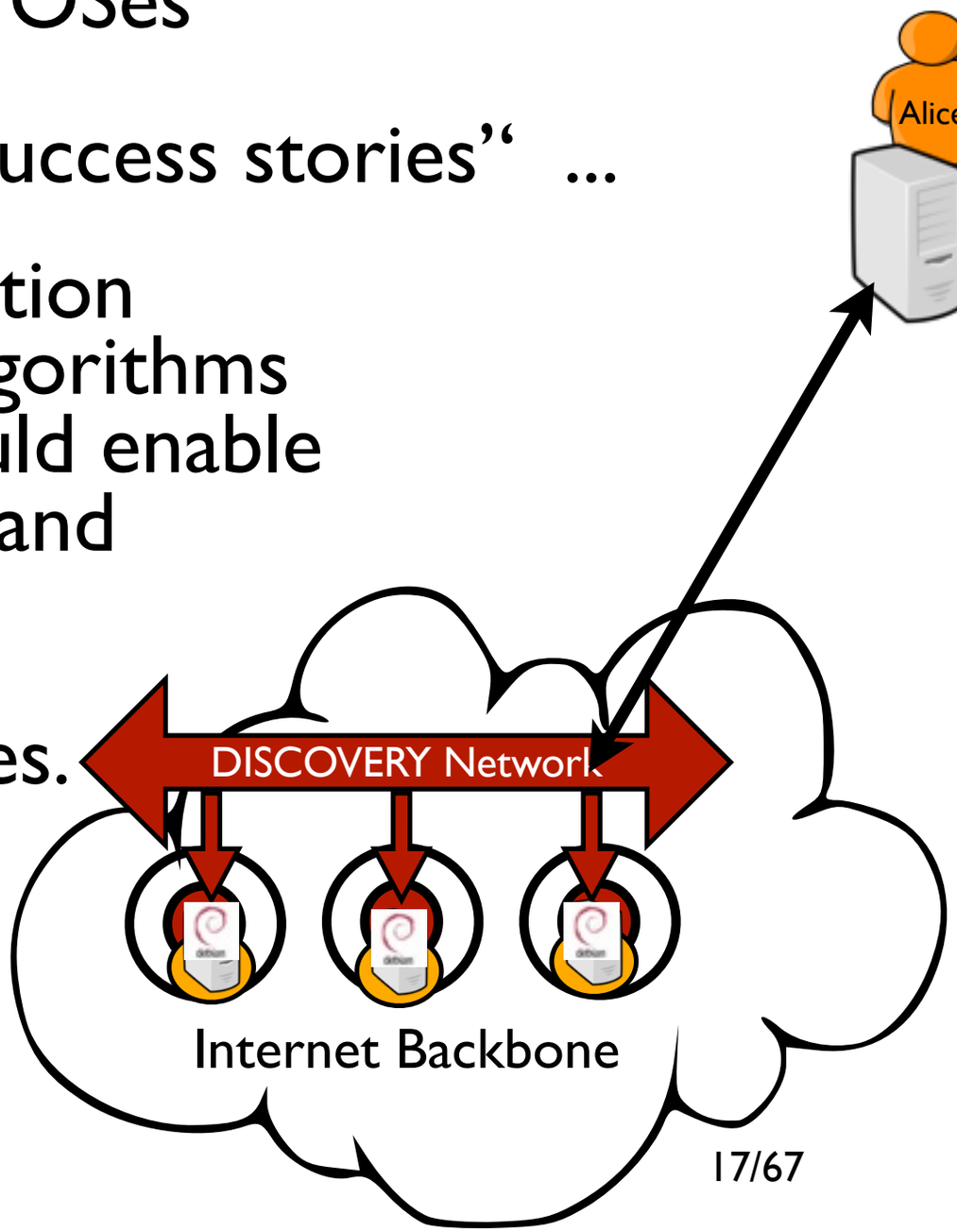


The DISCOVERY Proposal

- DIStributed and COoperative framework to manage Virtual EnviRonments autonomicallY
- Designing/implementing Distributed OSes

Deeply investigated with no "real success stories" ...

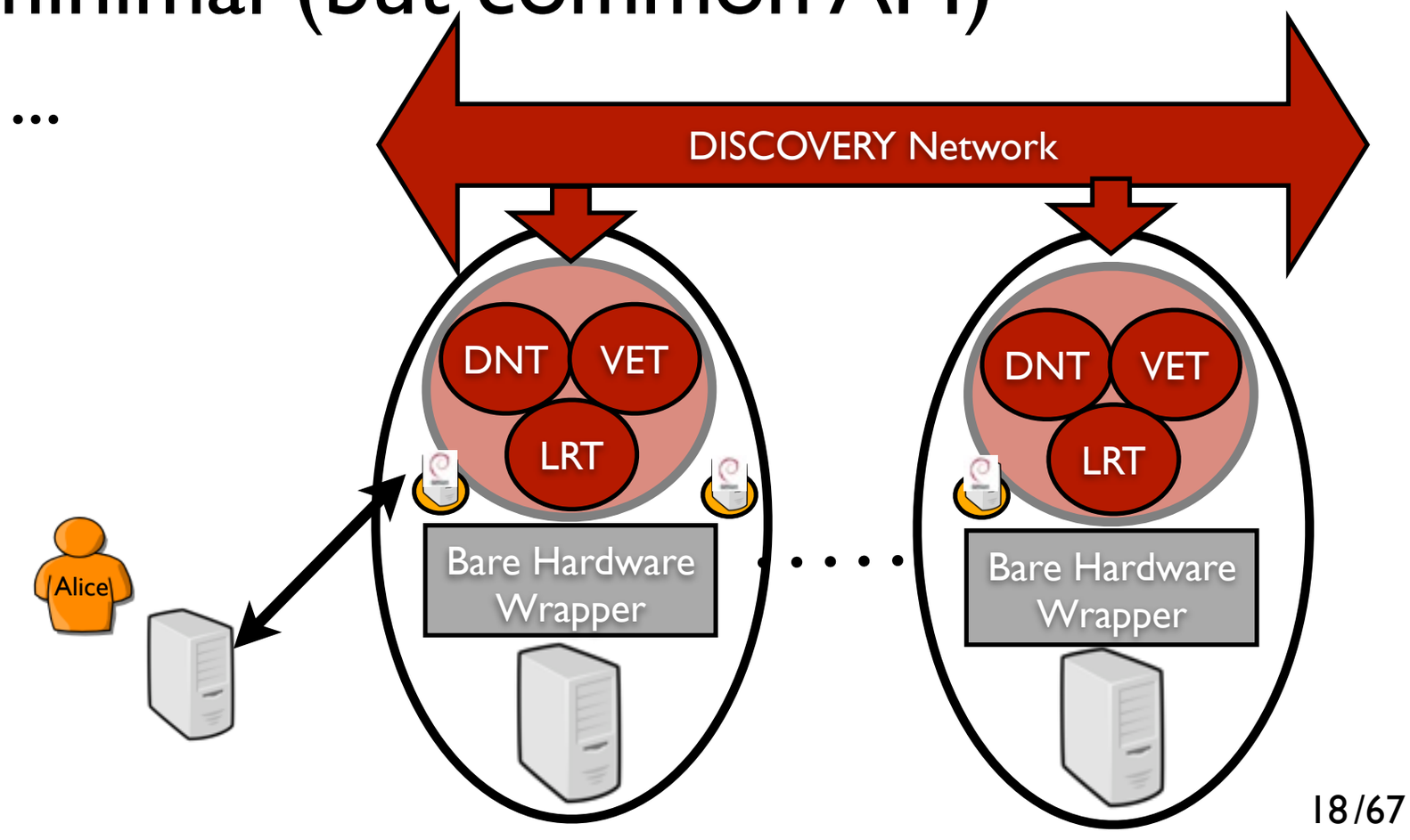
... But maturity of system virtualization capabilities as well as large scale algorithms and autonomous mechanisms should enable to design and implement a unified and autonomic system manipulating virtual environments (VEs) like traditional OS manipulate processes.



The LUC OS Agent - Overview

- 3 services
  Discovery Network Tracker (DNT)
  Virtual Environments Tracker (VET)
  Local Resources Tracker (LRT)

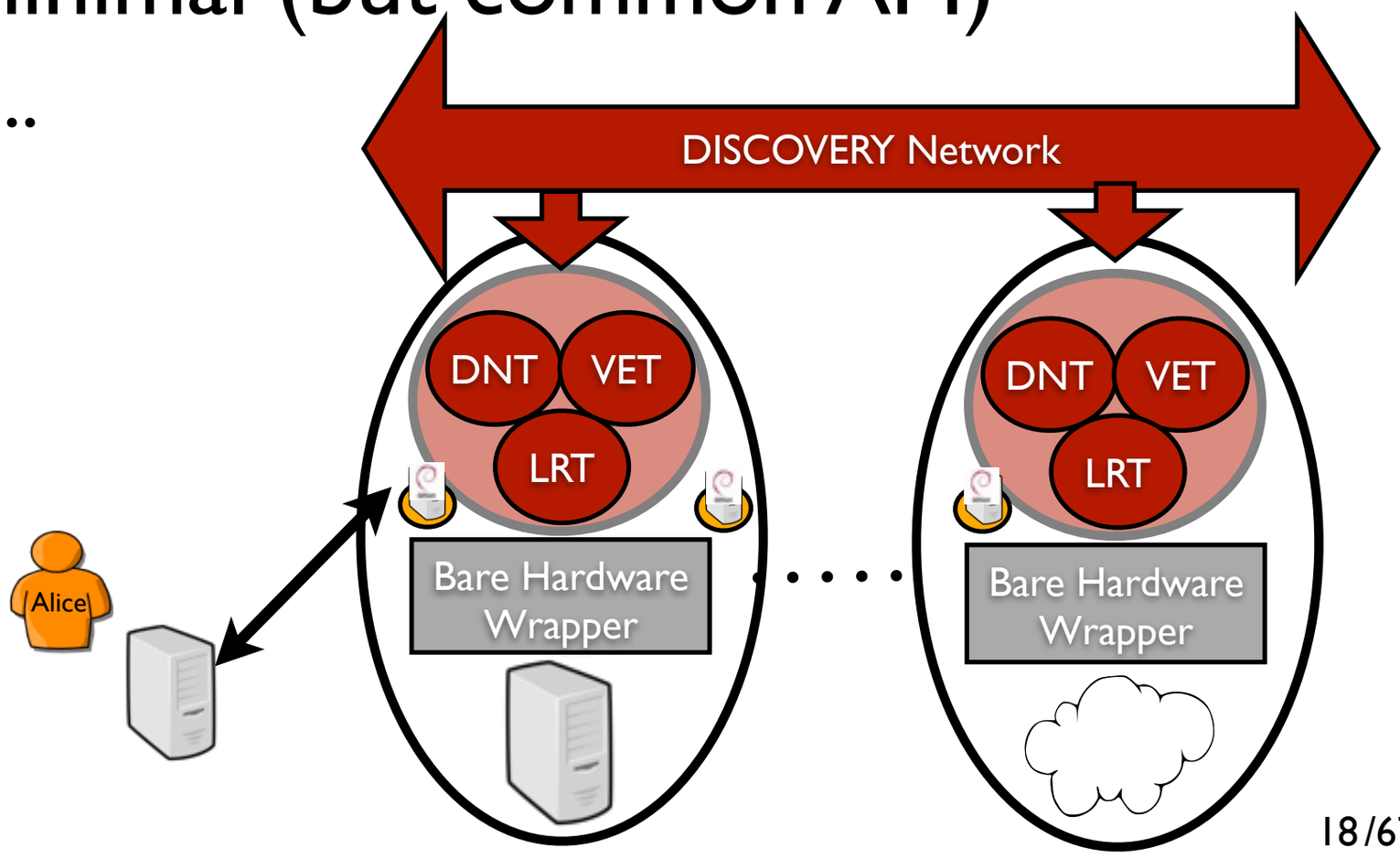
- Relying on a minimal (but common API)
  libvirt / OCCI / ...



The LUC OS Agent - Overview

- 3 services
  Discovery Network Tracker (DNT)
  Virtual Environments Tracker (VET)
  Local Resources Tracker (LRT)

- Relying on a minimal (but common API)
  libvirt / OCCI / ...



The DISCOVERY Initiative

- Focusing on the design and the implementation of a complete OS for IaaS platforms based on VMs and VEs (group of VMs) as the fundamental granularity

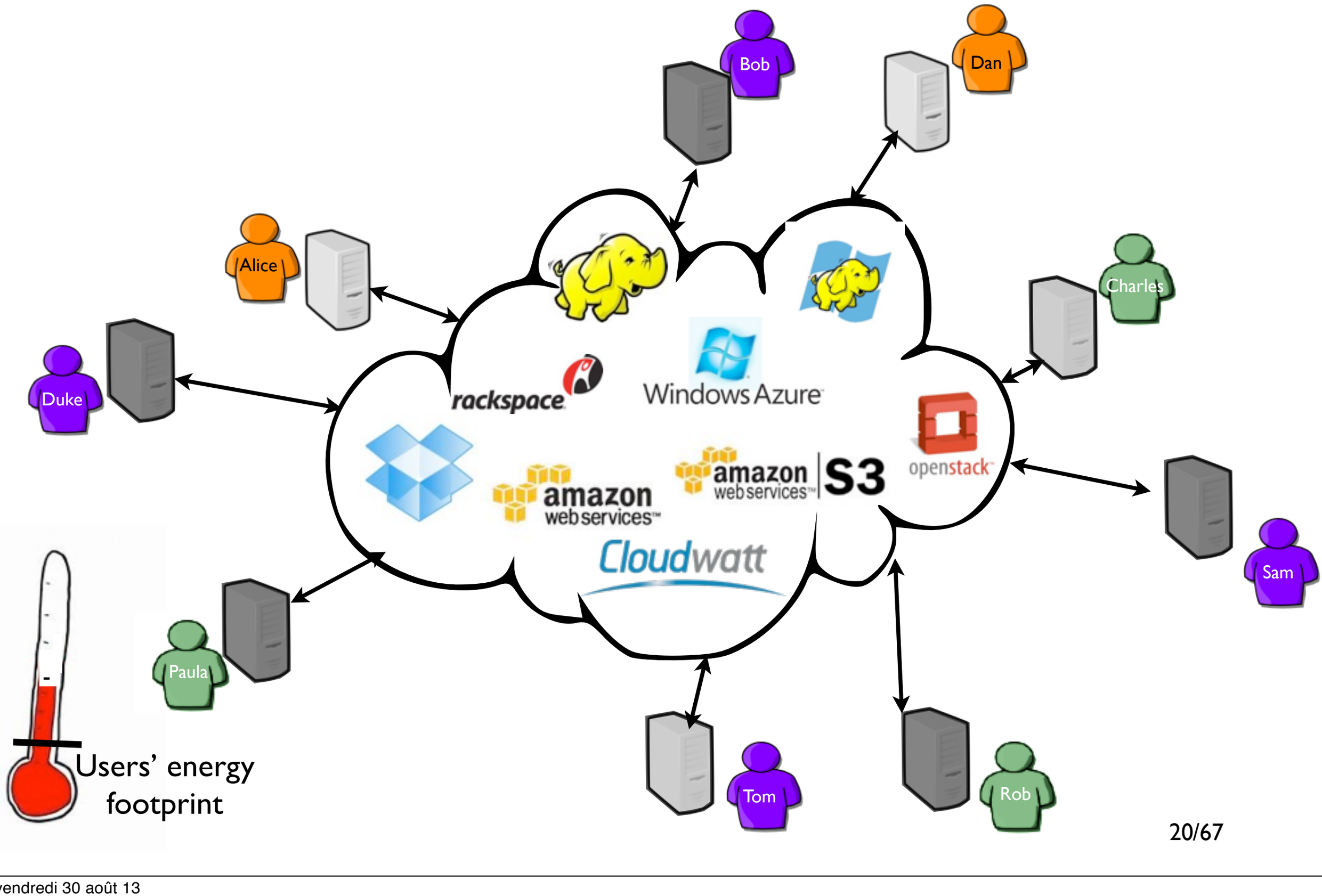
  Scalability, targeting the management of hundred thousands of VMs upon thousands of physical machines (PMs)
  Reliability, considering "hardware failures as the norm rather the exception"
  Reactivity, handling each reconfiguration event as swiftly as possible to maintain VEs' QoS.

- May look simple but lots of scientific/technical challenges

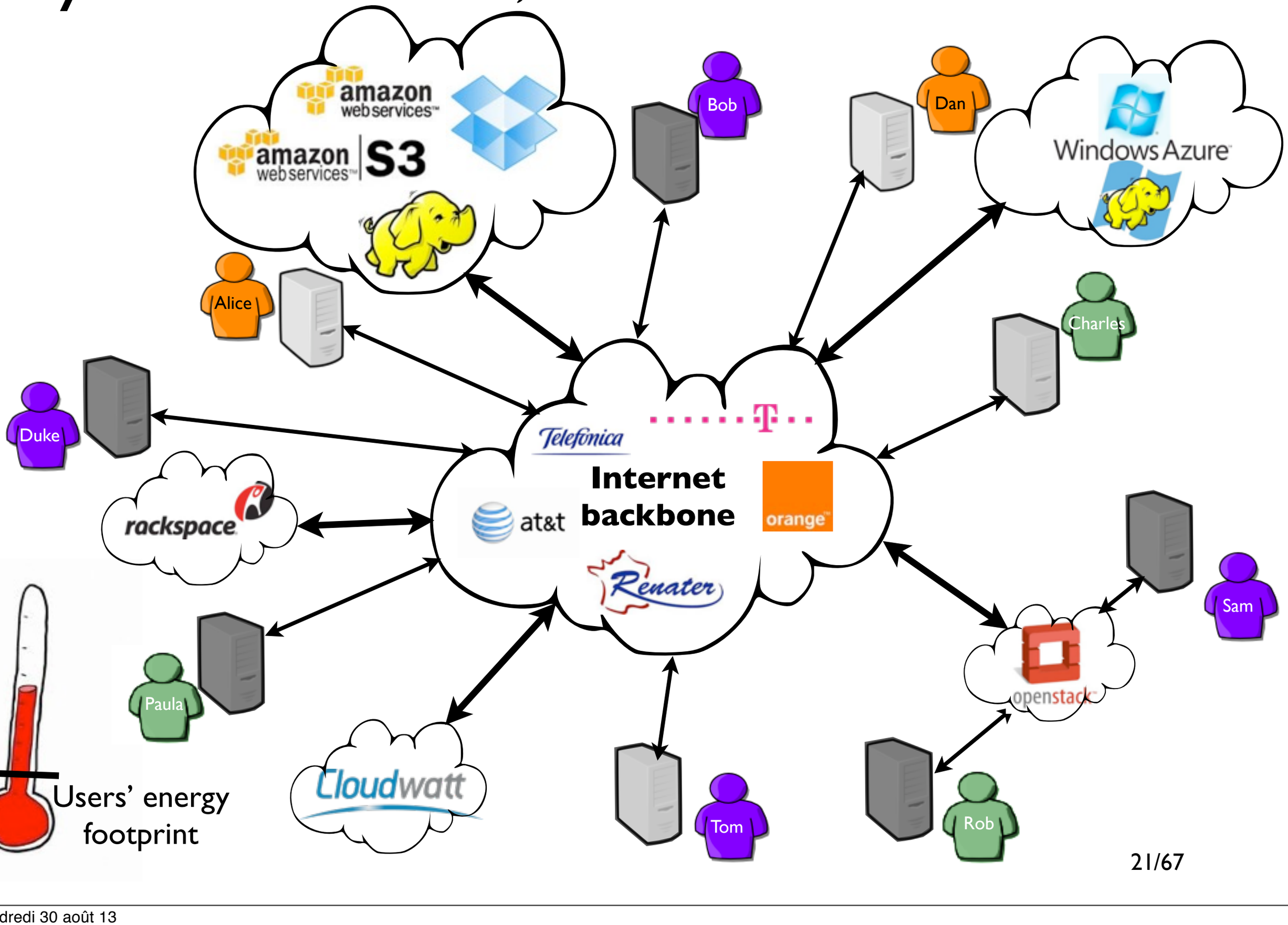
  Cost of the DISCOVERY network !? partial view of the system !?
  Impact on the others VMs !?, management of VM images !?
  Which software abstractions to make the development easier and more reliable (distributed event programming) ?

- A *Bittorent* like system ... but with stronger assumptions

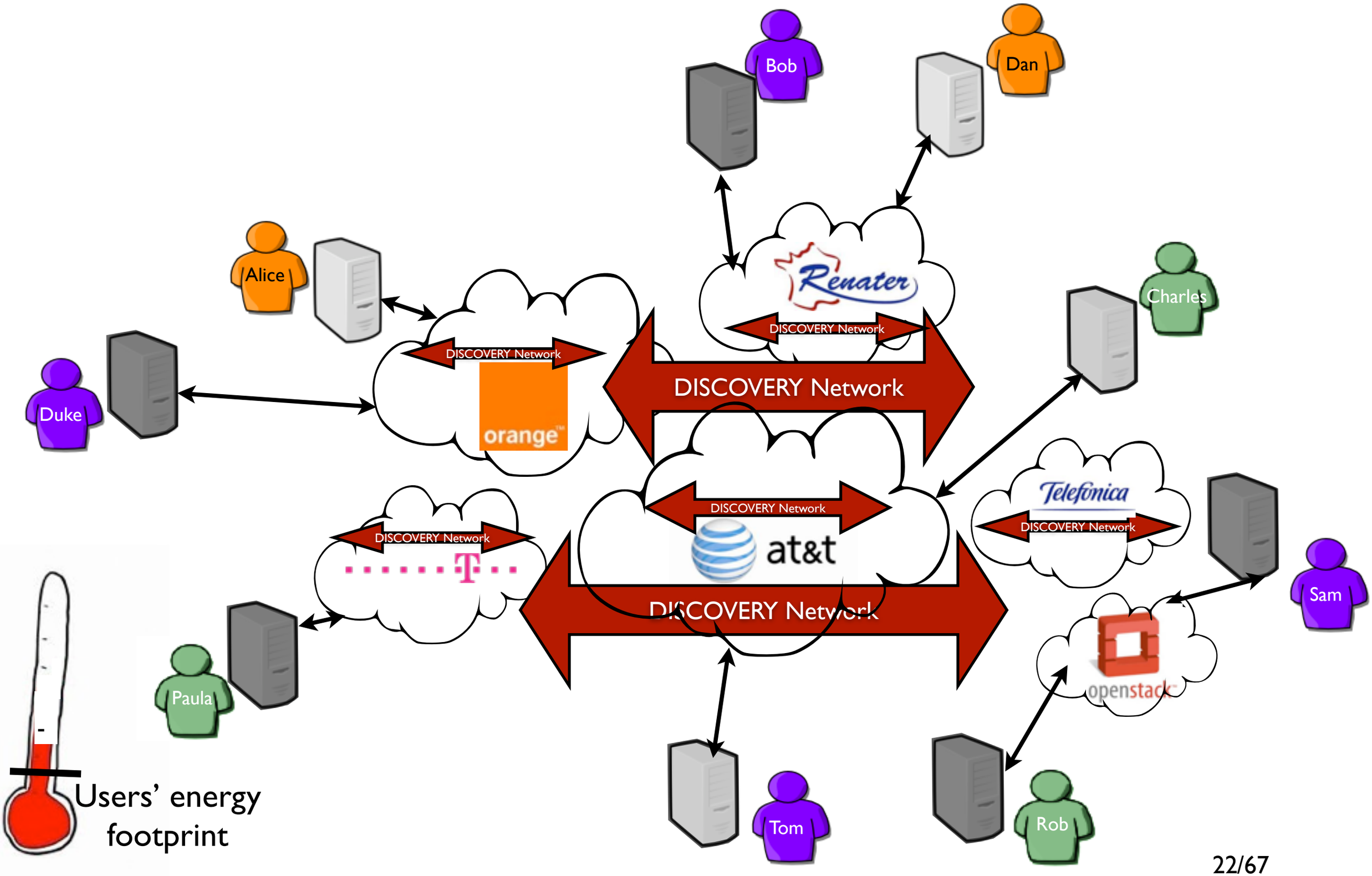
Beyond the Cloud, the DISCOVERY Initiative



Beyond the Cloud, the DISCOVERY Initiative



Beyond the Cloud, the DISCOVERY Initiative



The DISCOVERY Initiative

- Leveraging former projects but still on the starting blocks!
- Strong interests of large companies
(SAP, Orange Lab, Citrix, ...)
- RENATER
- An important actor to follow: Akamai
- Preliminary works with promising results
(especially on the LRT: a first POC)
- Long term objective: impact on the design of distributed applications in order to take advantage of the locality
(building S3 like system)

The DISCOVERY Initiative

- Thank you / Questions ?
- What's next

  - Focus on LRT (Flavien Quesnel's Phd, ended in Feb 2013)
  - Discovery internals in a nutshell

- On going work - The discovery framework from the Software Programming point of view (Jonathan Pastor's Phd, 2012/2015)

CRS4 IDEAS BECOME LIFE · MINES Nantes · lina · Inria · EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE · Université de Neuchâtel unine · BSC

http://beyondtheclouds.github.io/

The DISCOVERY Initiative

- Thank you / Questions ?
- What's next

  - Focus on LRT (Flavien Quesnel's Phd, ended in Feb 2013)
  - Discovery internals in a nutshell

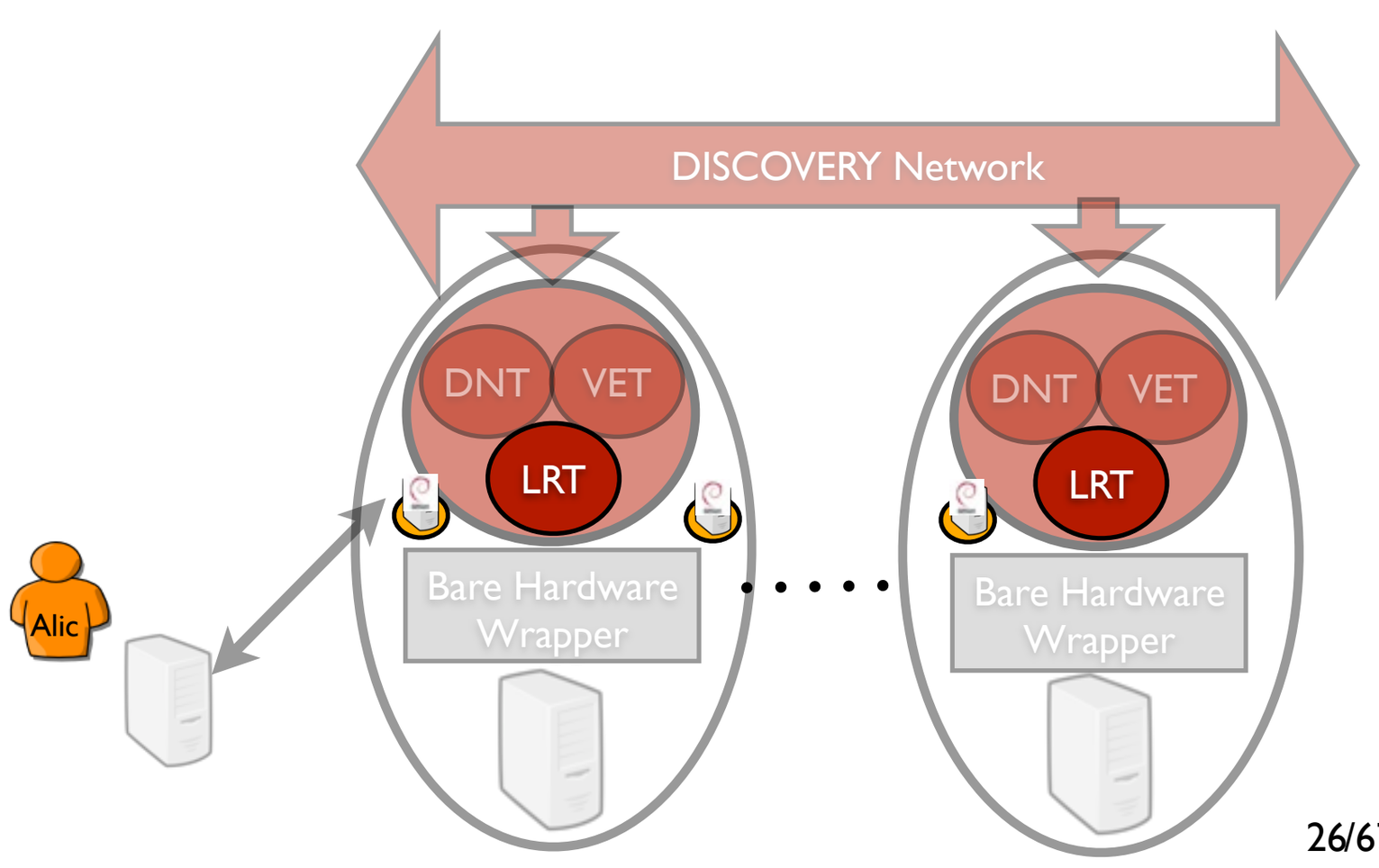
- On going work - The discovery framework from the Software Programming point of view (Jonathan Pastor's Phd, 2012/2015)

http://beyondtheclouds.github.io/

The LUC OS Agent
Focus on the LRT

The LUC OS Agent
Local Resource Tracker

- The LRT is in charge of monitoring and dynamically balancing VMs according to their effective usage of the physical resources.

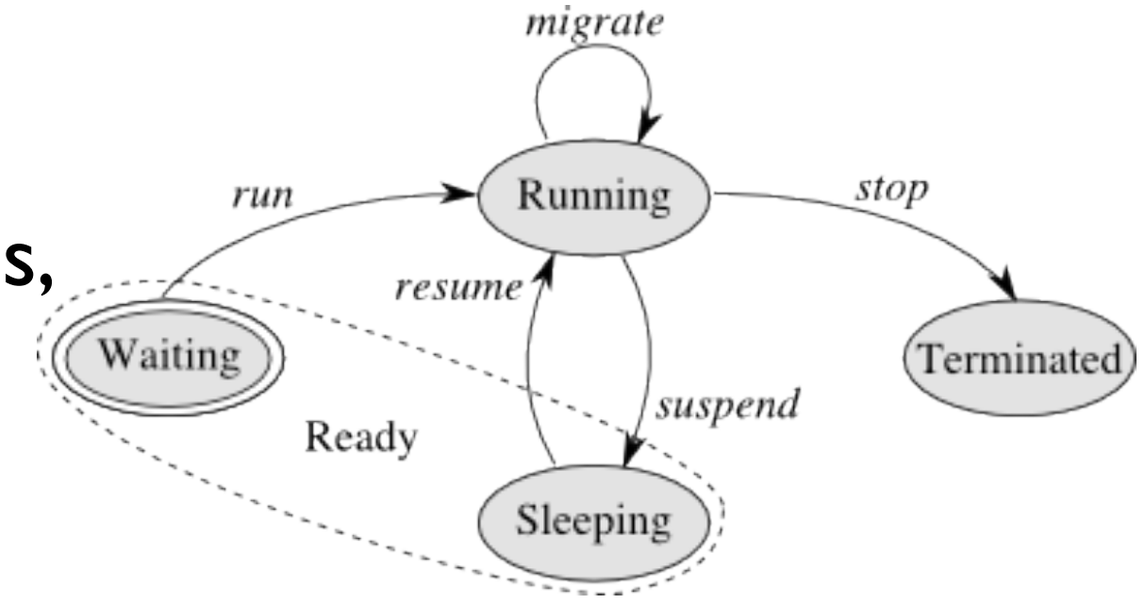


Background on VM dynamic scheduling

Background - a VE-based OS

- General idea: manipulate VEs instead of processes
  (a VE is a users' working environment, possibly composed of several interconnected VMs)

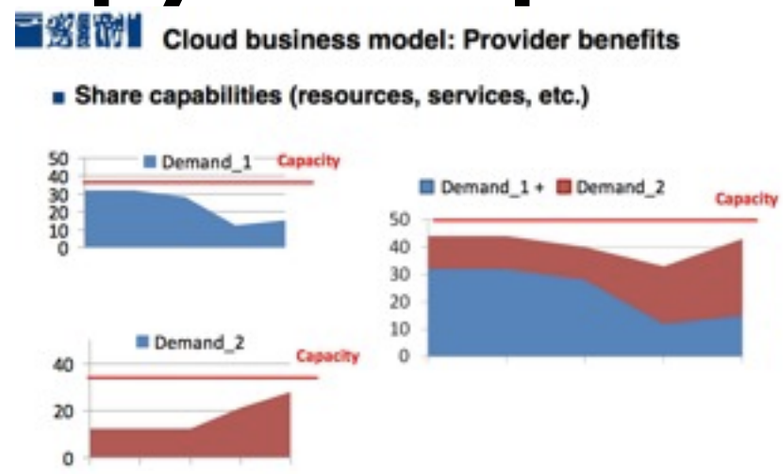
- In a similar way of usual processes, each VE is in a particular state:



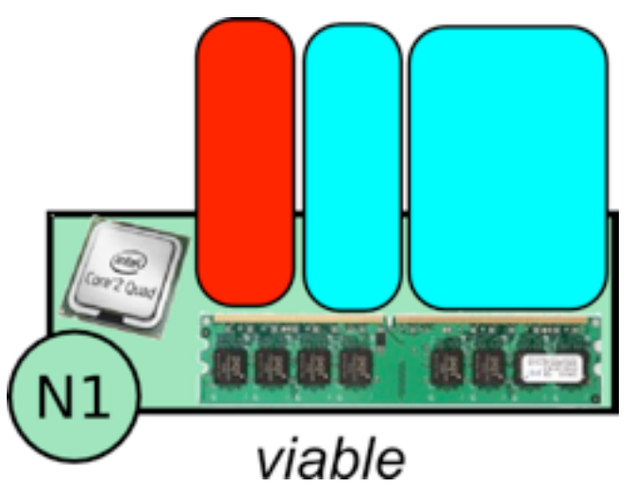
- Perform VE context switches (a set of VM context switches) to rebalance the LUC infrastructure according to the: scheduler objectives / available resources / waiting queue / ...

Background - The Entropy Proposal

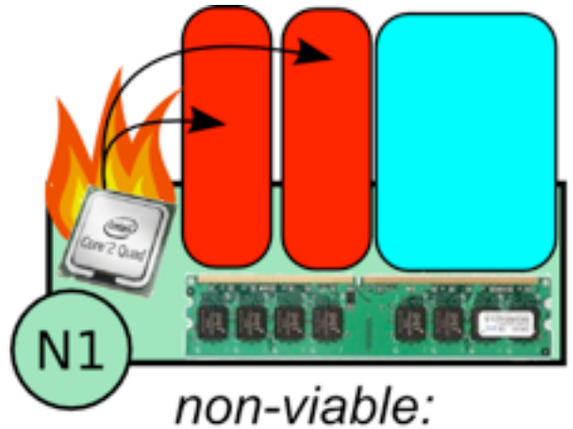
- Fine management of resources (efficiency and energy constraints)
- Find the "right" mapping between needs of VMs and resources provided by PMs



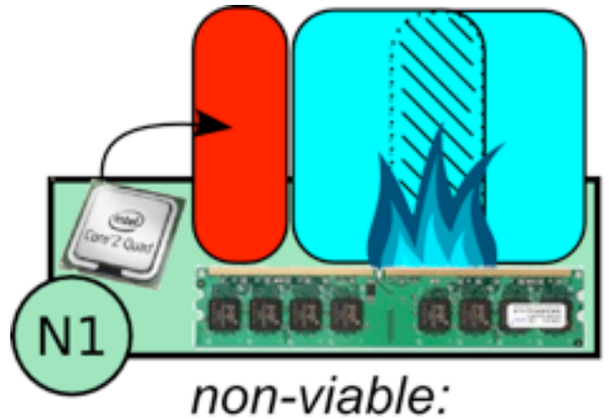
credits: S. Tata, Telecom Summer School 2013



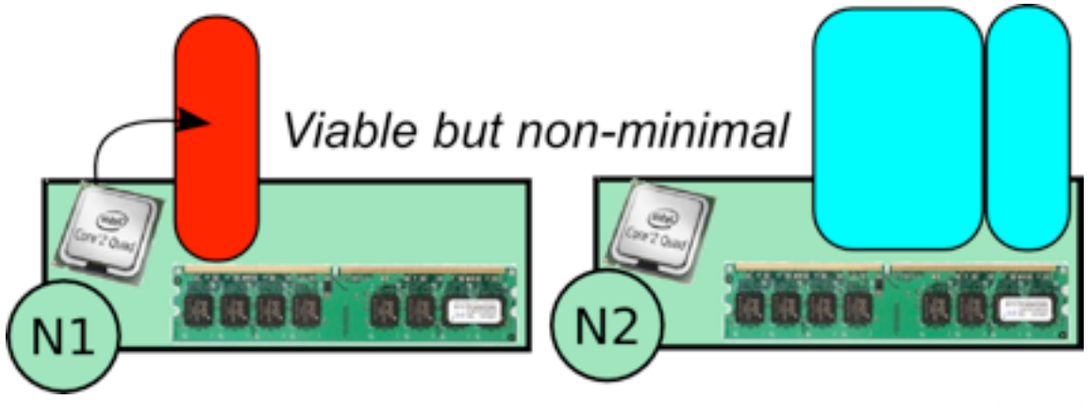
viable



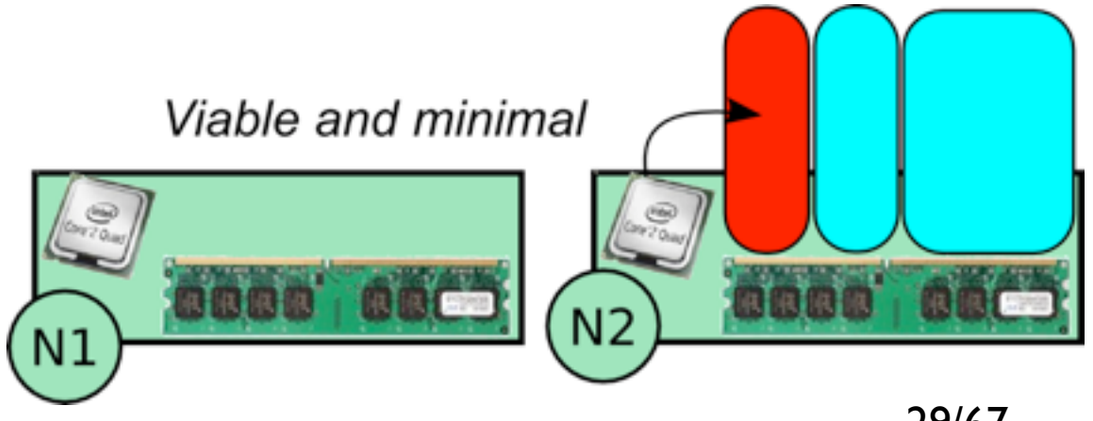
non-viable:
2 active VMs for one CPU



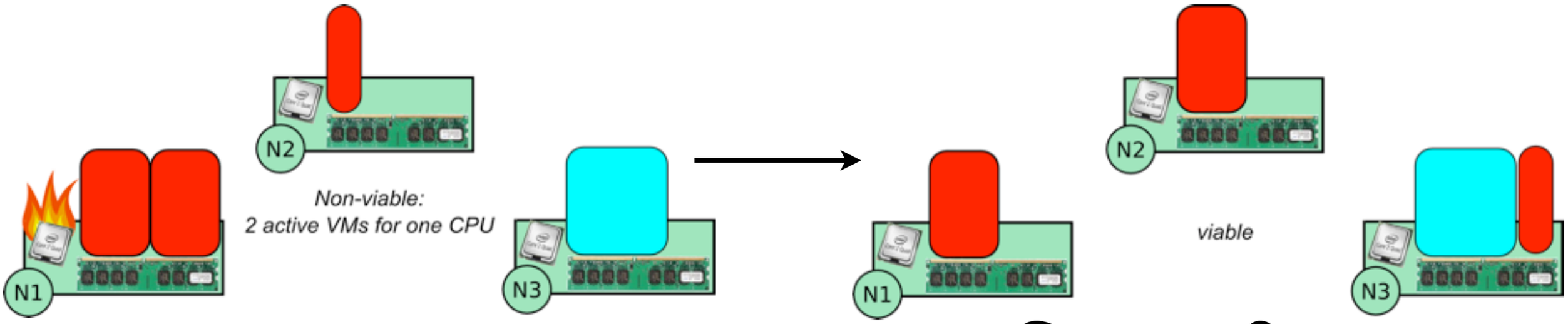
non-viable:
memory overcommitment



credits: F. Hermenier, OSDI poster session 2008

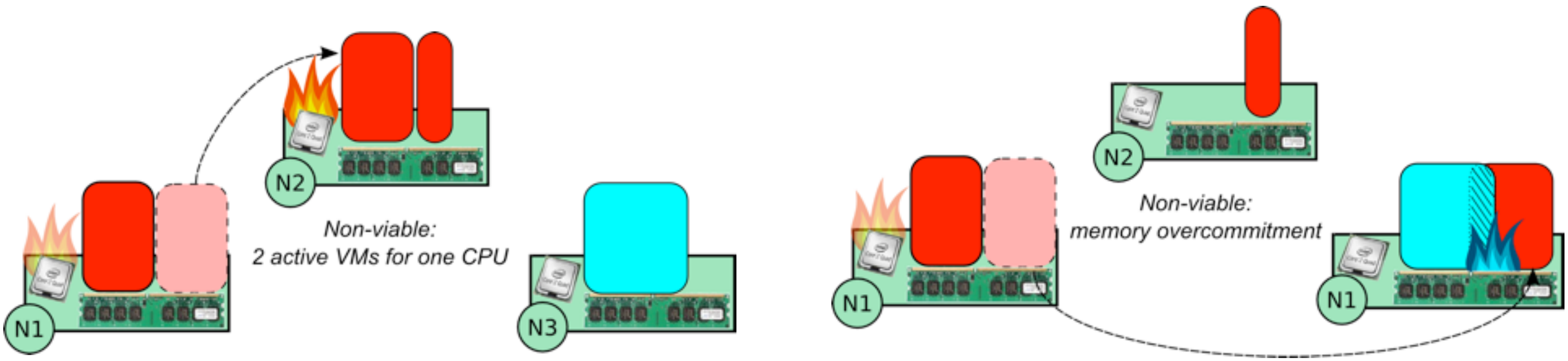


Background - The Entropy Proposal



Current Status

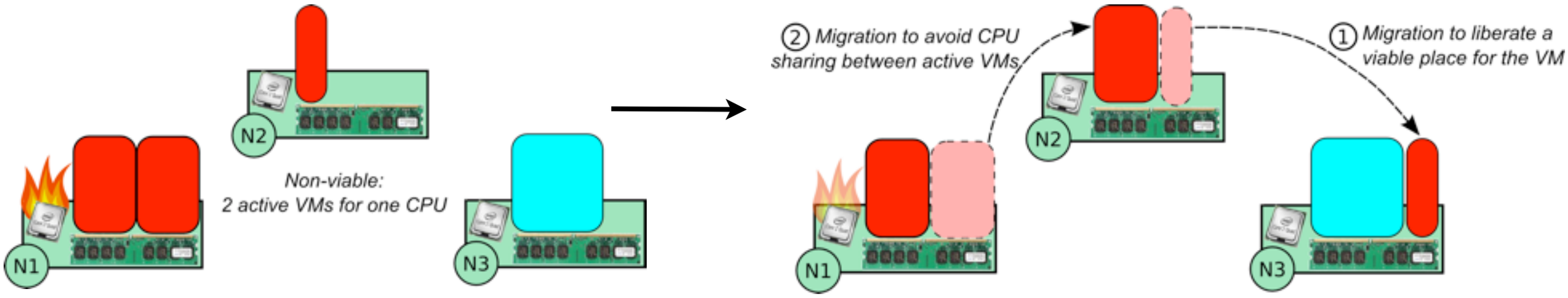
Correct Status



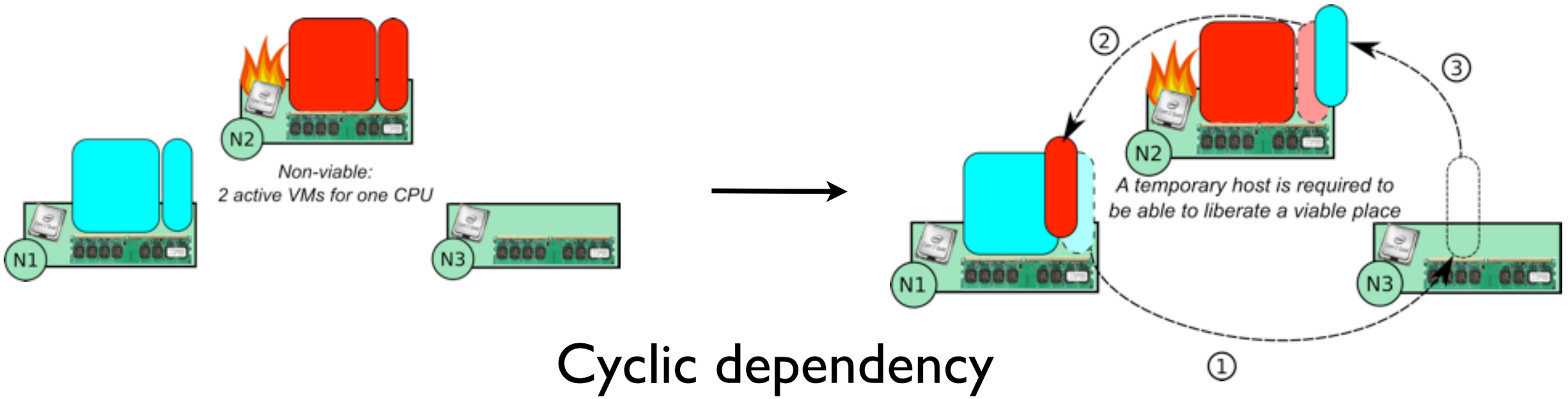
Non-viable manipulations

Background - The Entropy Proposal

- Order VM Operations



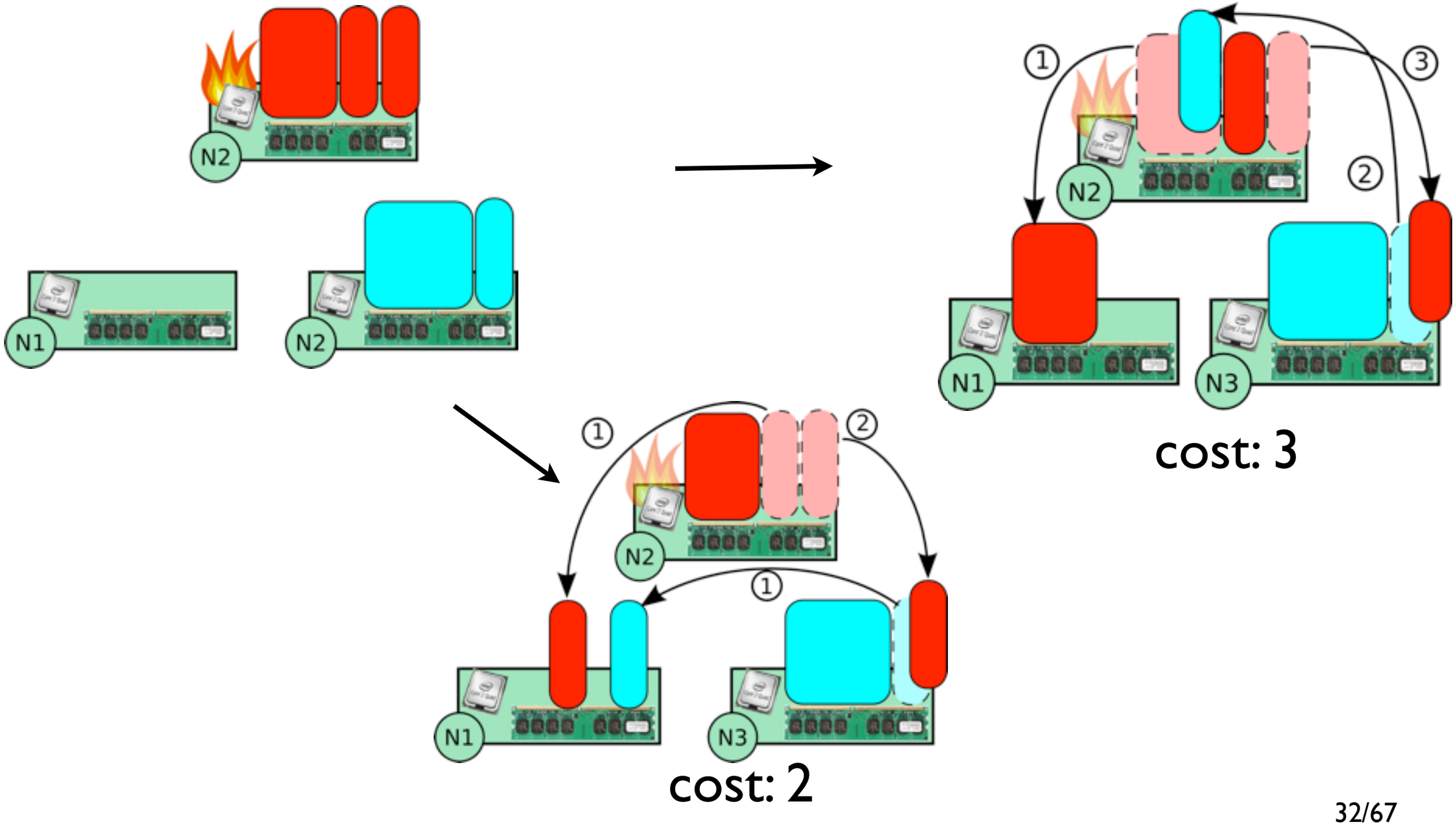
Sequential dependency



Cyclic dependency

Background - The Entropy Proposal

- Optimizing the reconfiguration process



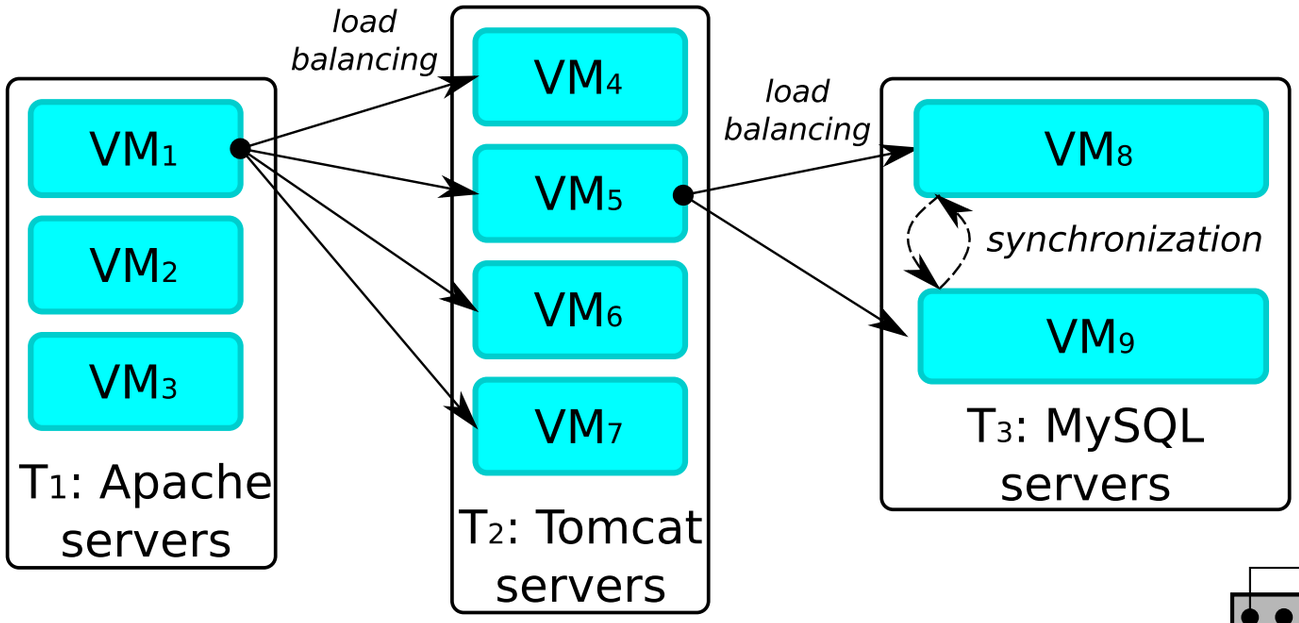
Background - The Entropy Proposal More Constraints

- Manipulate VEs dynamically can lead to non desired configurations
- Additional constraints should be considered

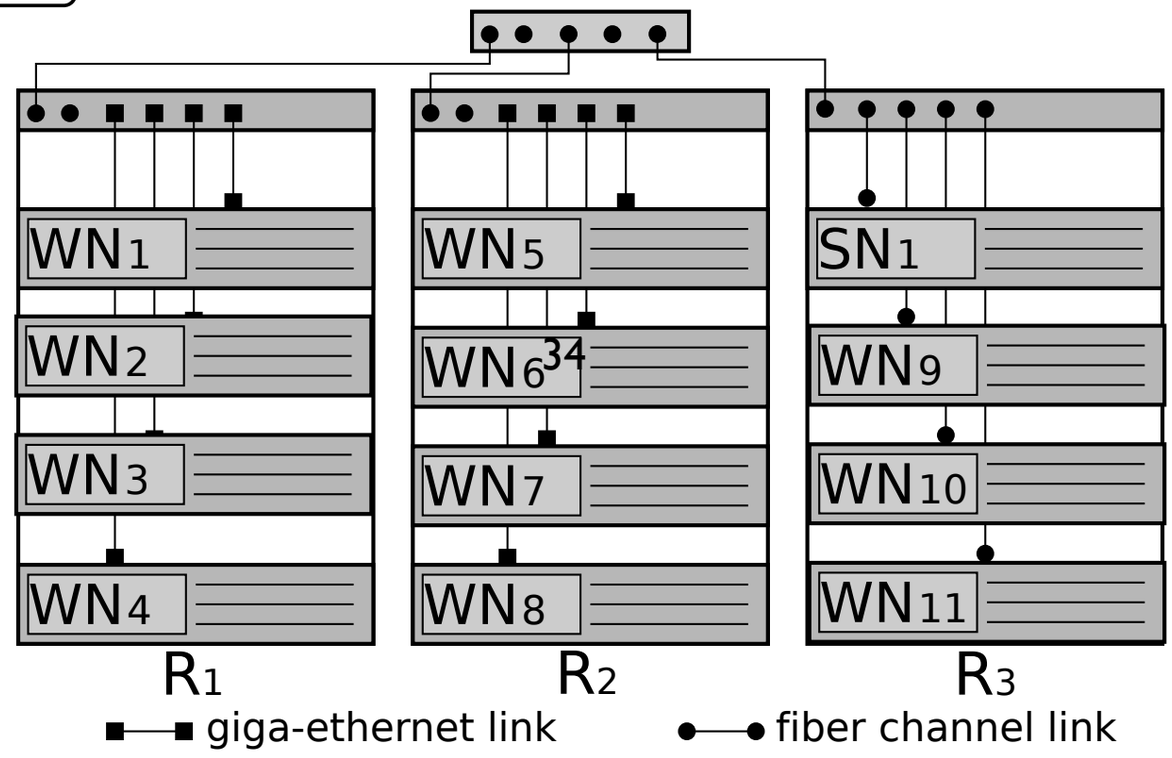
  To take into account particular requirements according to the infrastructure (performance, HA, maintenance operations....)

  To maintain VE "consistency" during reconfigurations

Background - Plasma and Entropy

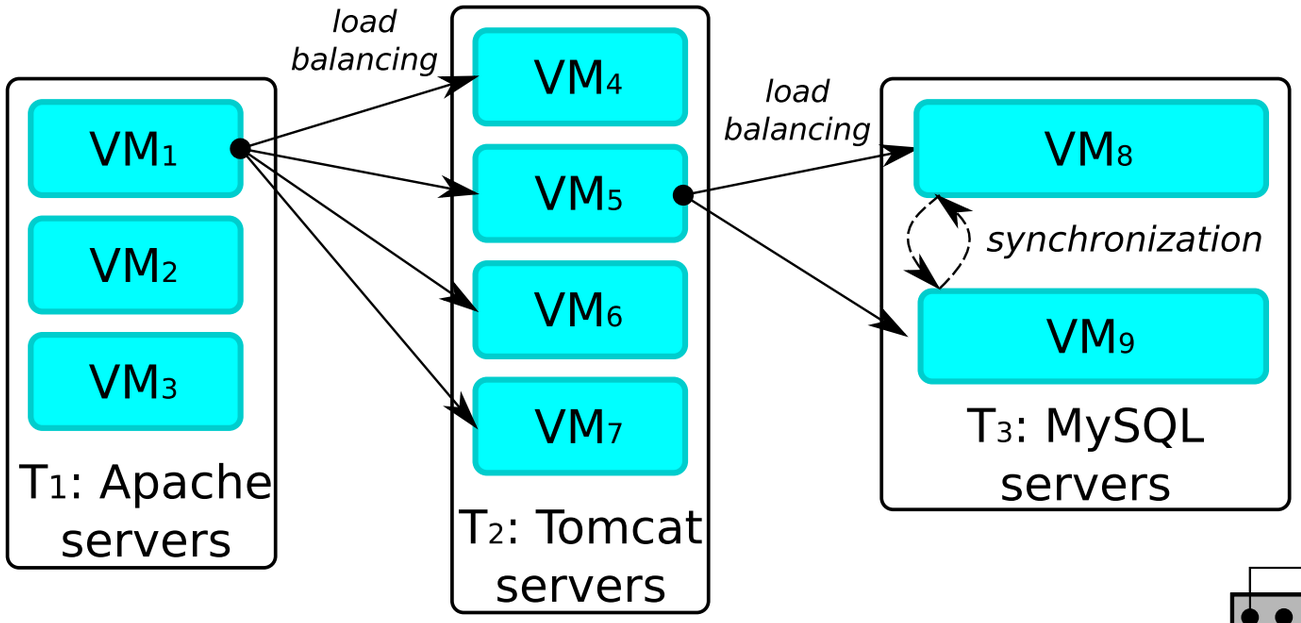


Virtualized HA application



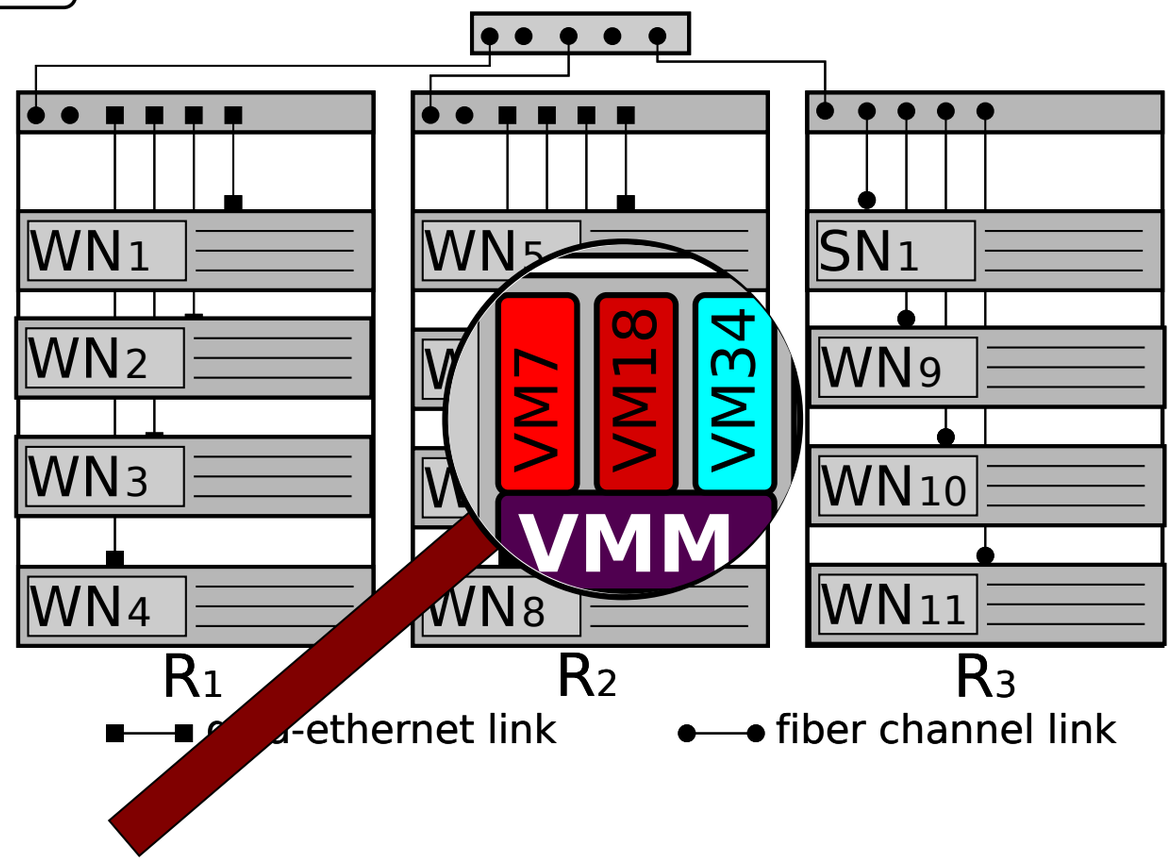
Credits: Hermenier et al., RR-7545 INRIA

Background - Plasma and Entropy



Virtualized HA application

Plasma, a DSL to describe:
- the infrastructure
- the VEs and their placement constraints



Credits: Hermenier et al., RR-7545 INRIA

Background - Plasma and Entropy

- ban({VM1, VM2}, {N1, N2})

  Prevents a set of VMs from being hosted on a given set of nodes

- fence({VM1, VM2}, {N1, N2})

  Forces a set of VMs to be hosted on a set of nodes

- spread({VM1, VM2})

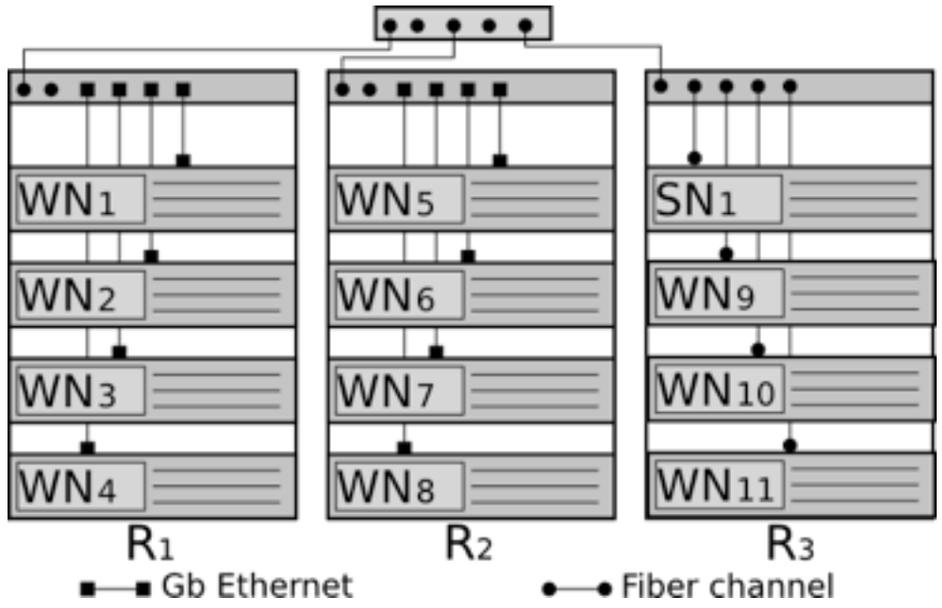
  Ensures that the specified VMs are never hosted on the same node at the same time

- latency({VM1, VM2}, {{N1, N2}, {N3, N4}})

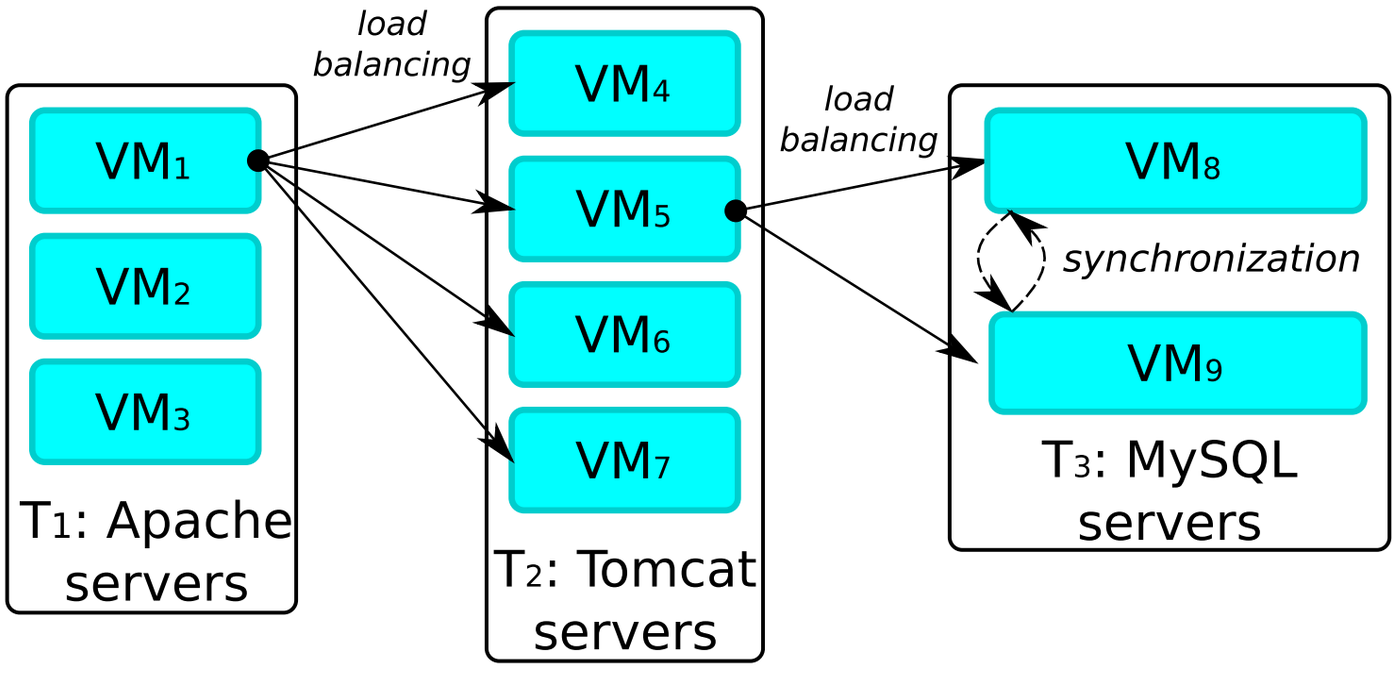
  Forces a set of VMs to be hosted on a single group of nodes

- See more on **http://btrp.inria.fr/**

Infrastructure/Application Description




```
// Infrastructure
$R1 = {WN1 ,WN2 ,WN3 ,WN4 };
$R2 = WN [5..8];
$R3 = WN [9..11] + {SN1 };

// Classes of latency
$small = {$R3 };
$medium = $R [1..3];

// Constraints
ban ( $ALL_VMS , {SN1 });
ban ( $ALL_VMS , {WN5 });
fence ($A1 , $R2 + $R3 );
```

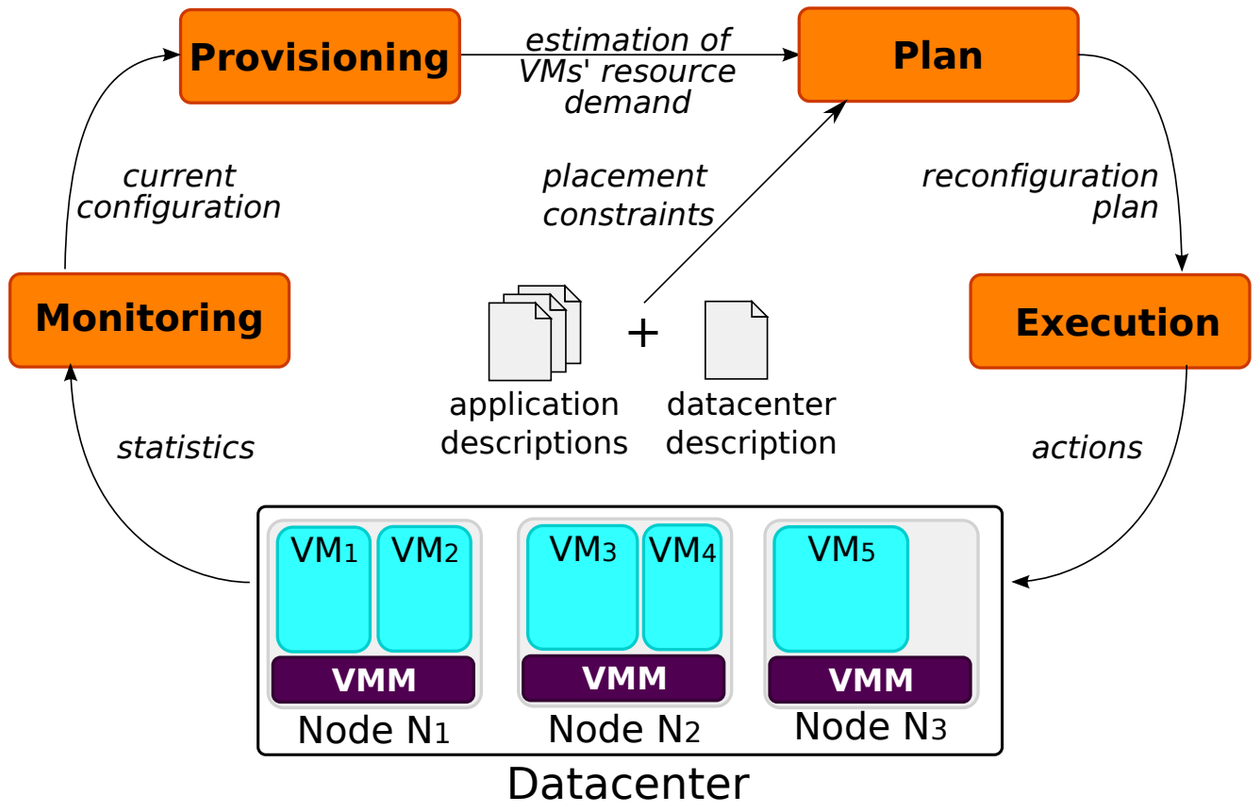



```
// The 3 tiers
$T1 = {VM1 ,VM2 ,VM3 };
$T2 = VM [4..7];
$T3 = VM [8..9];

// Fault tolerance to hw. failures
spread($T1);
spread($T2);
spread($T3);

// Efficient synchronization
latency ($T3 , $small );
```

Background - Entropy / btrPlace

An autonomic framework to maintain viable VE placements

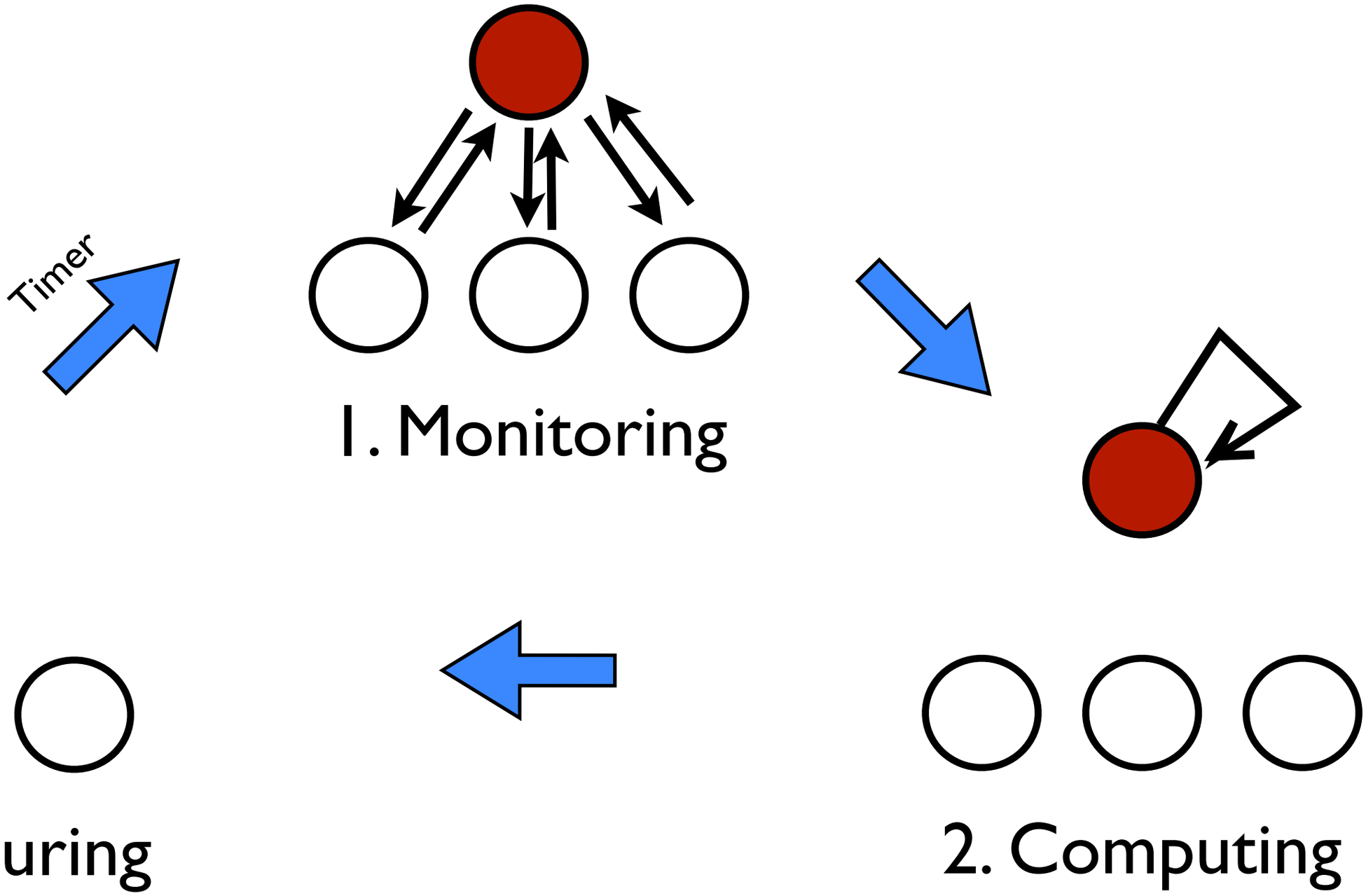
Developed since 2006 (ANR SelfXL / ANR Emergence, 10 persons / EasyVirt)
ASCOLA Research Group (Mines Nantes)
Oasis Research Team (University Of Nice Sophia Antipolis)



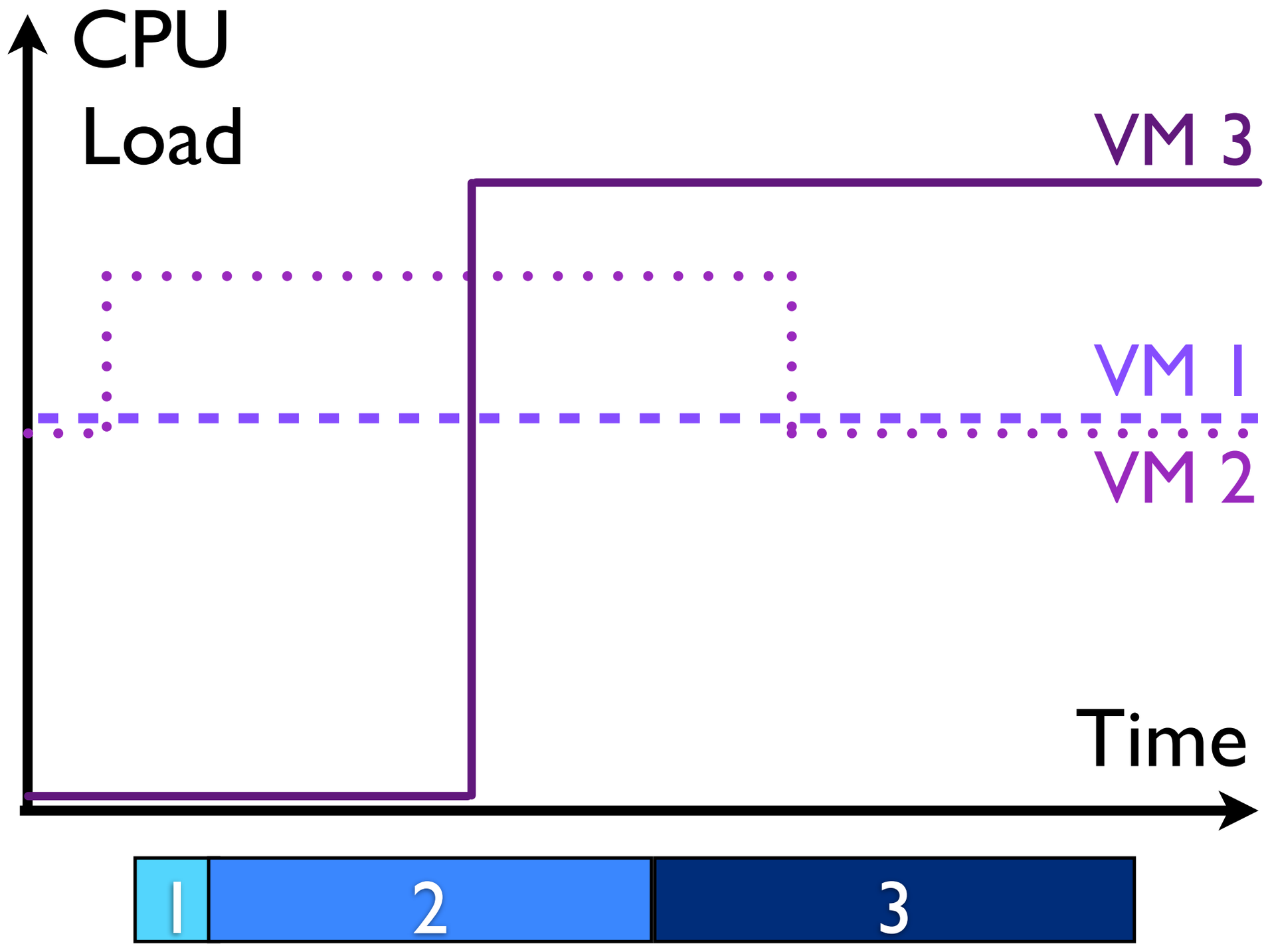
Scalability / Reactivity / Cost of reconfigurations

Dynamic Scheduling of VMs

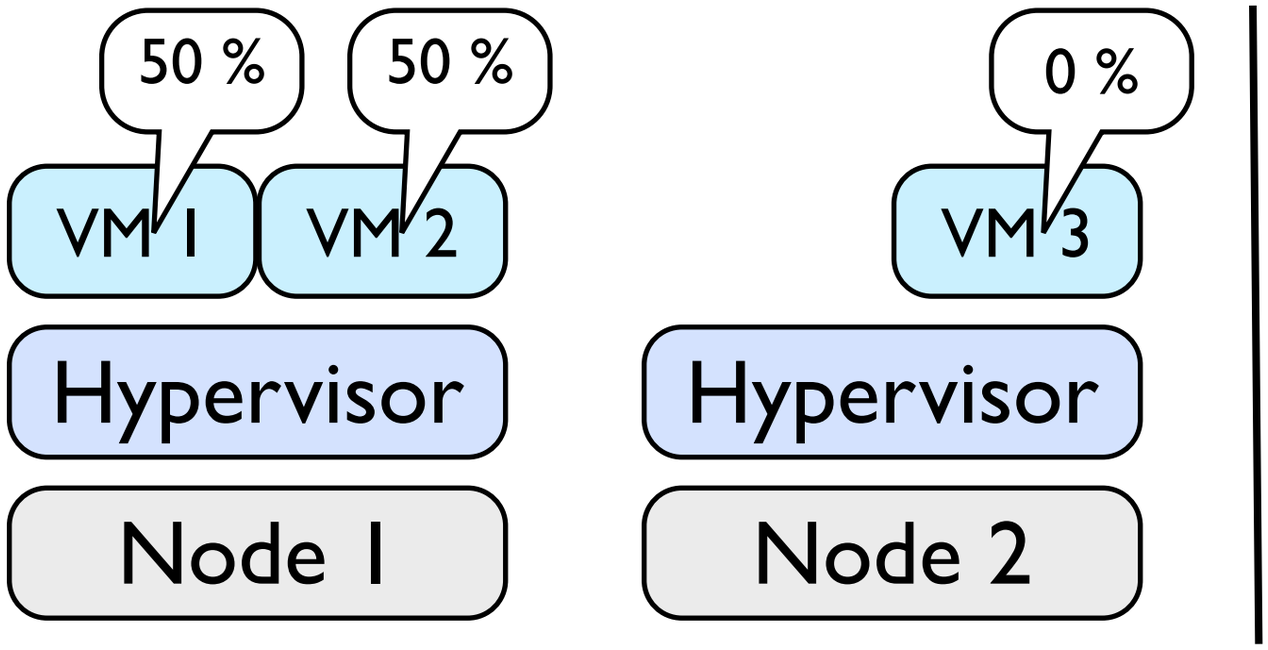
Centralized approach



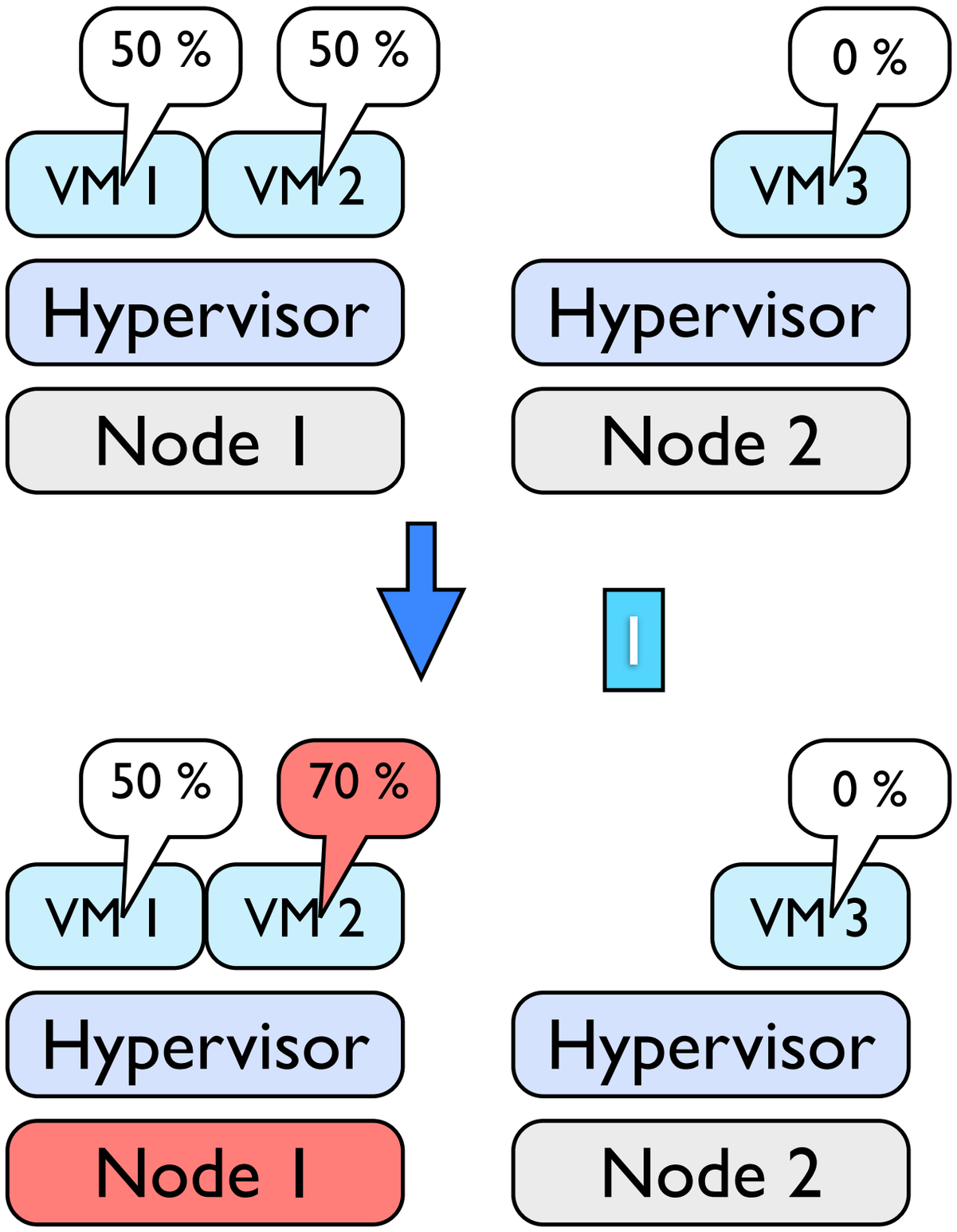
Scalability vs Reactivity



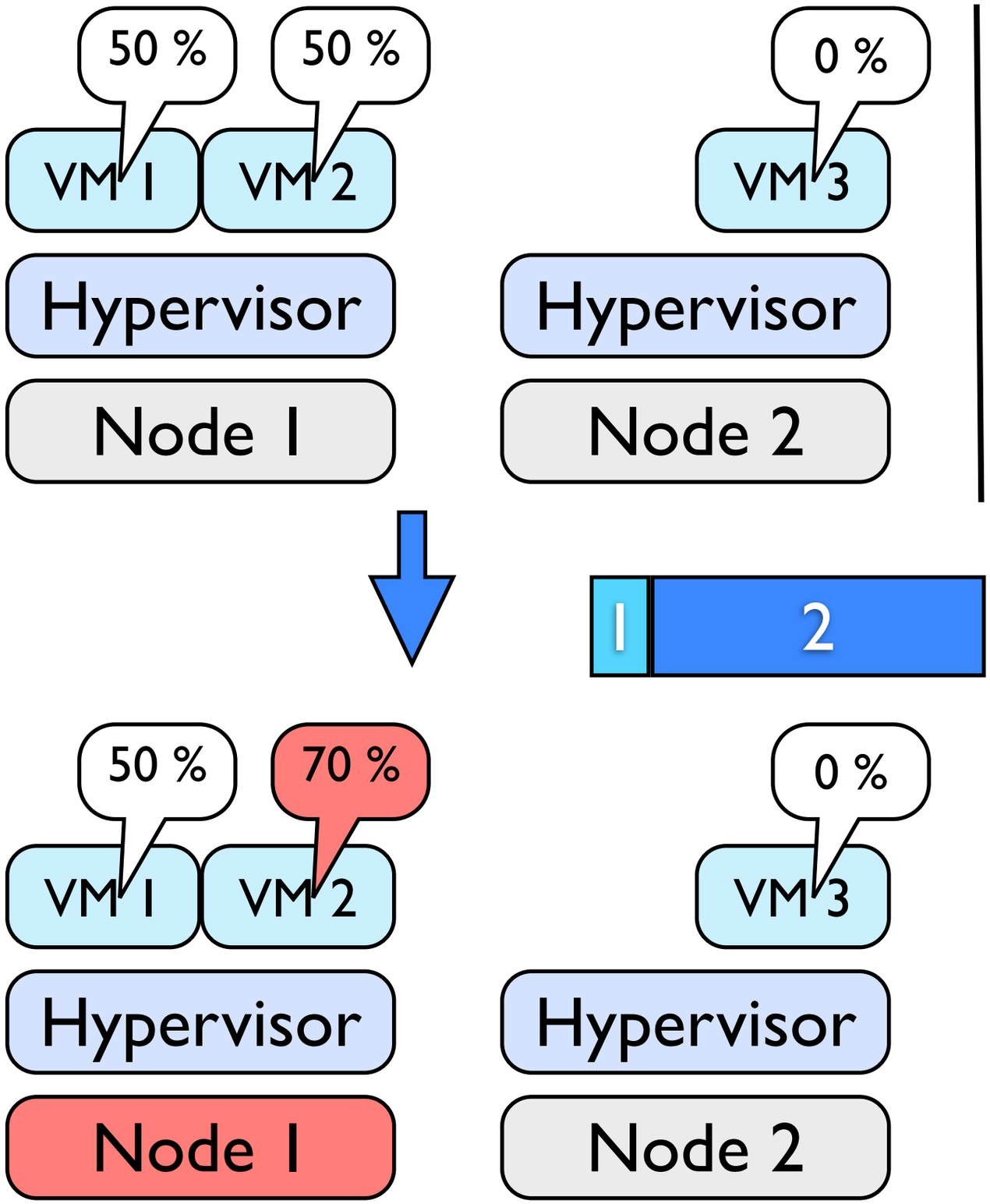
Scalability vs Reactivity



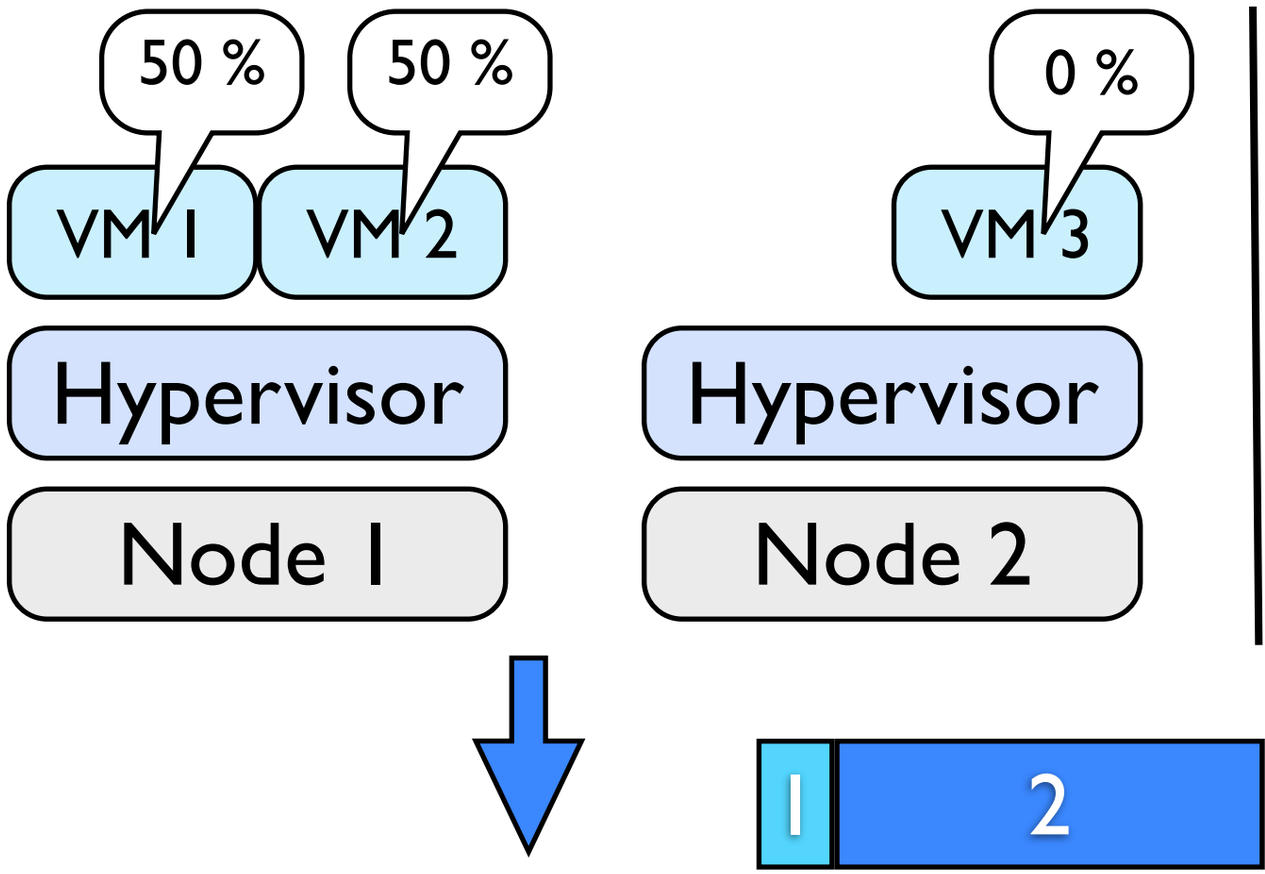
Scalability vs Reactivity

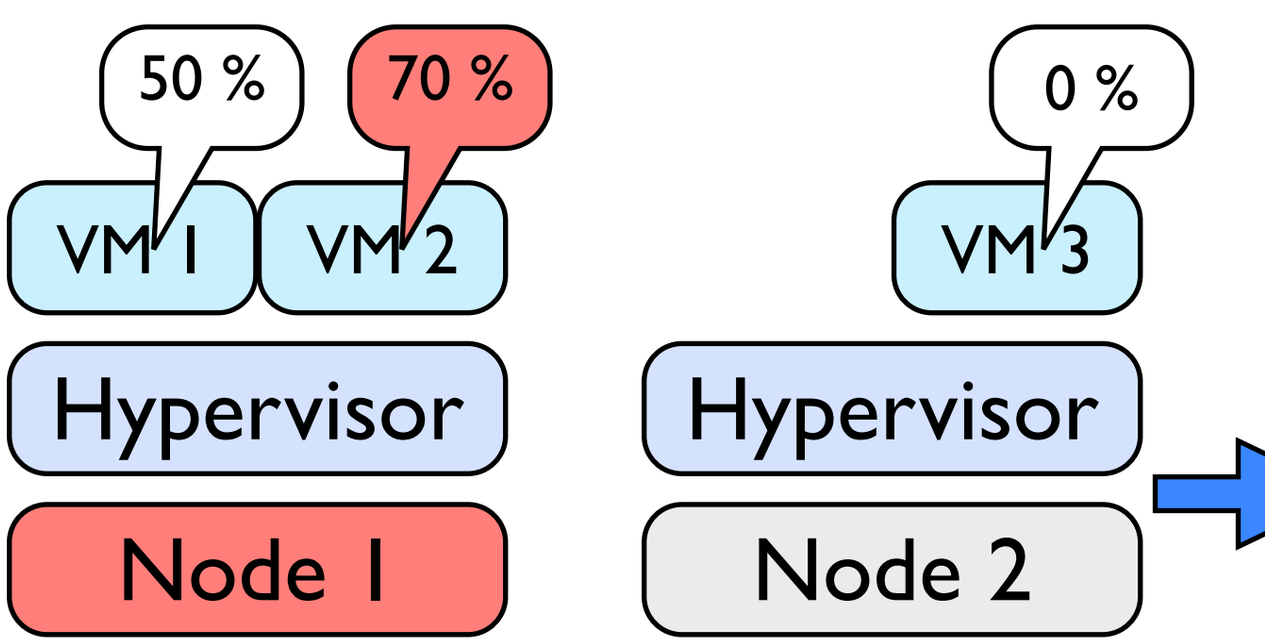


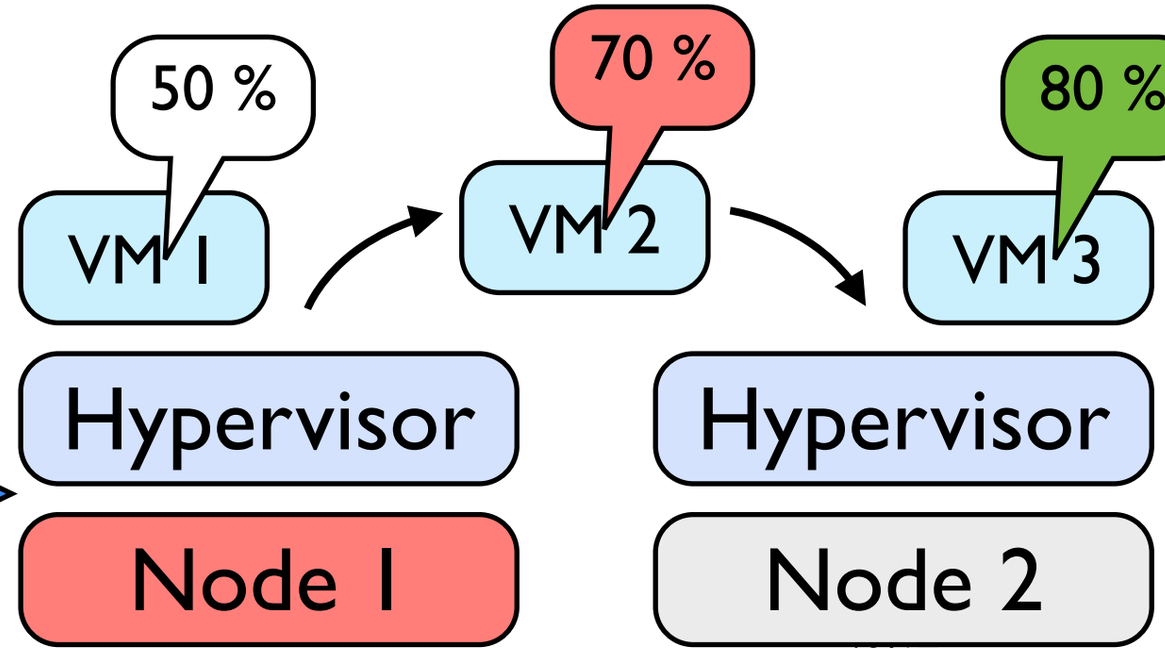
Scalability vs Reactivity



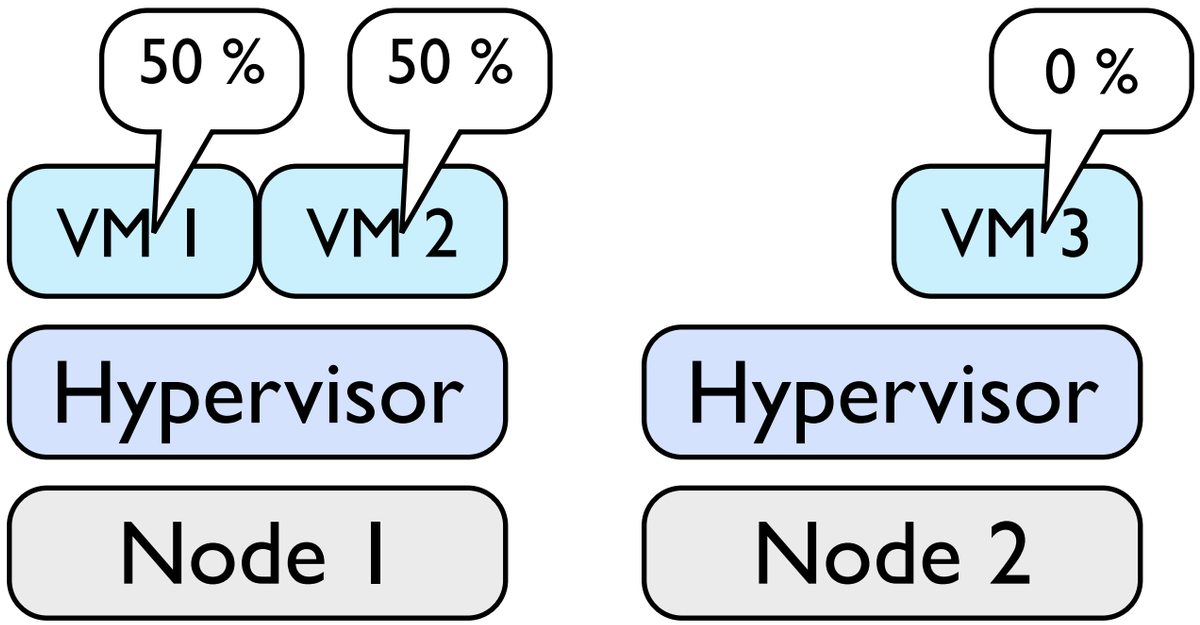
Scalability vs Reactivity

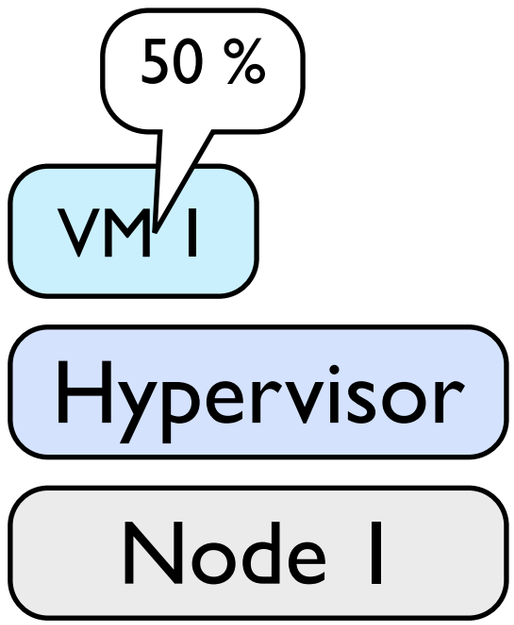


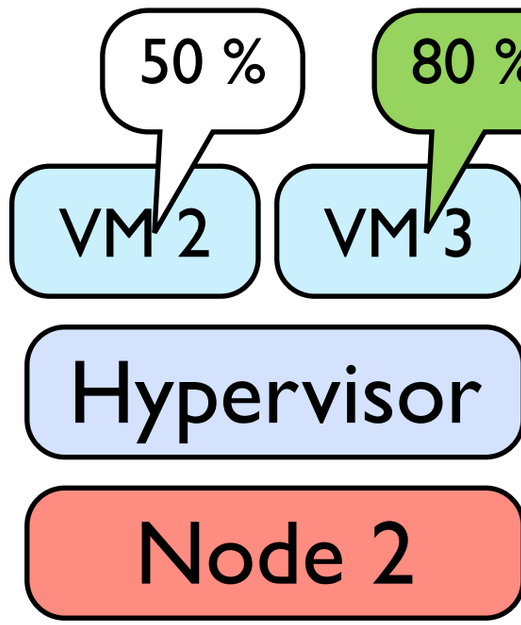


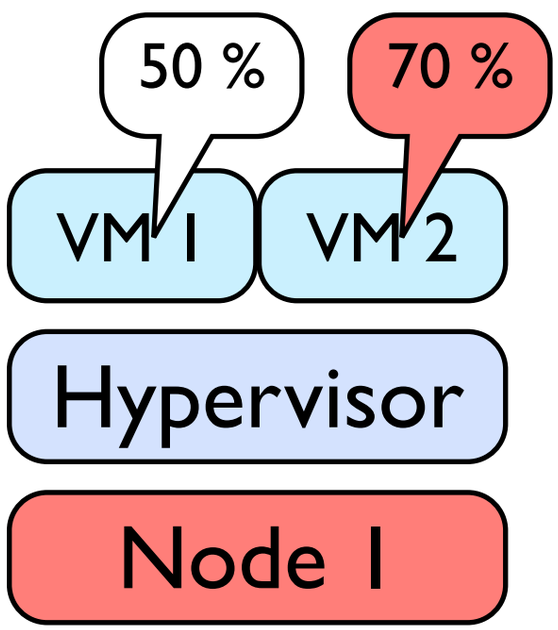


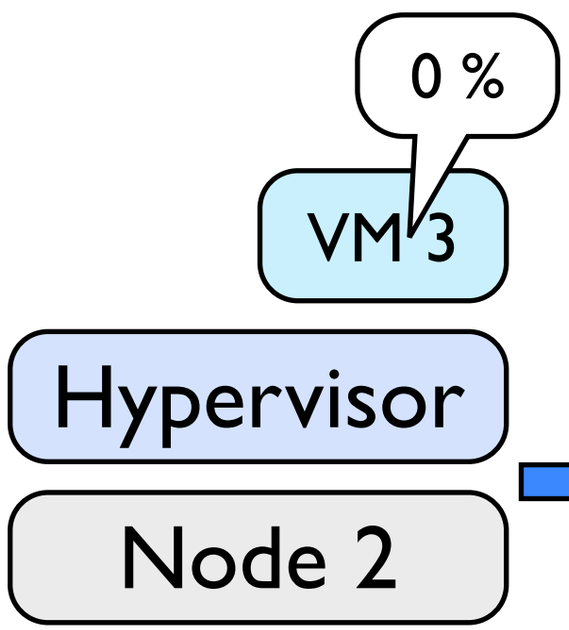
Scalability vs Reactivity

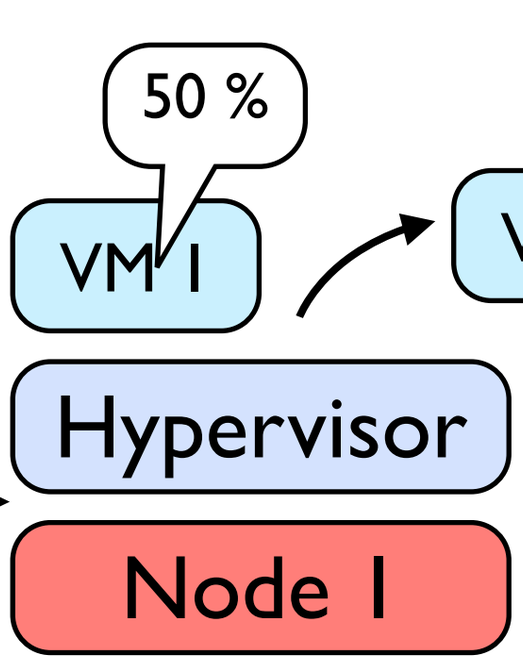


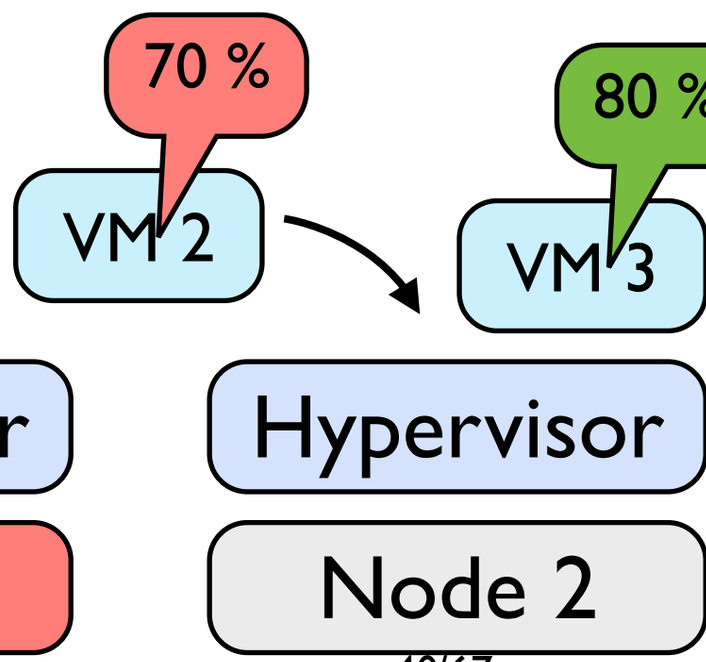




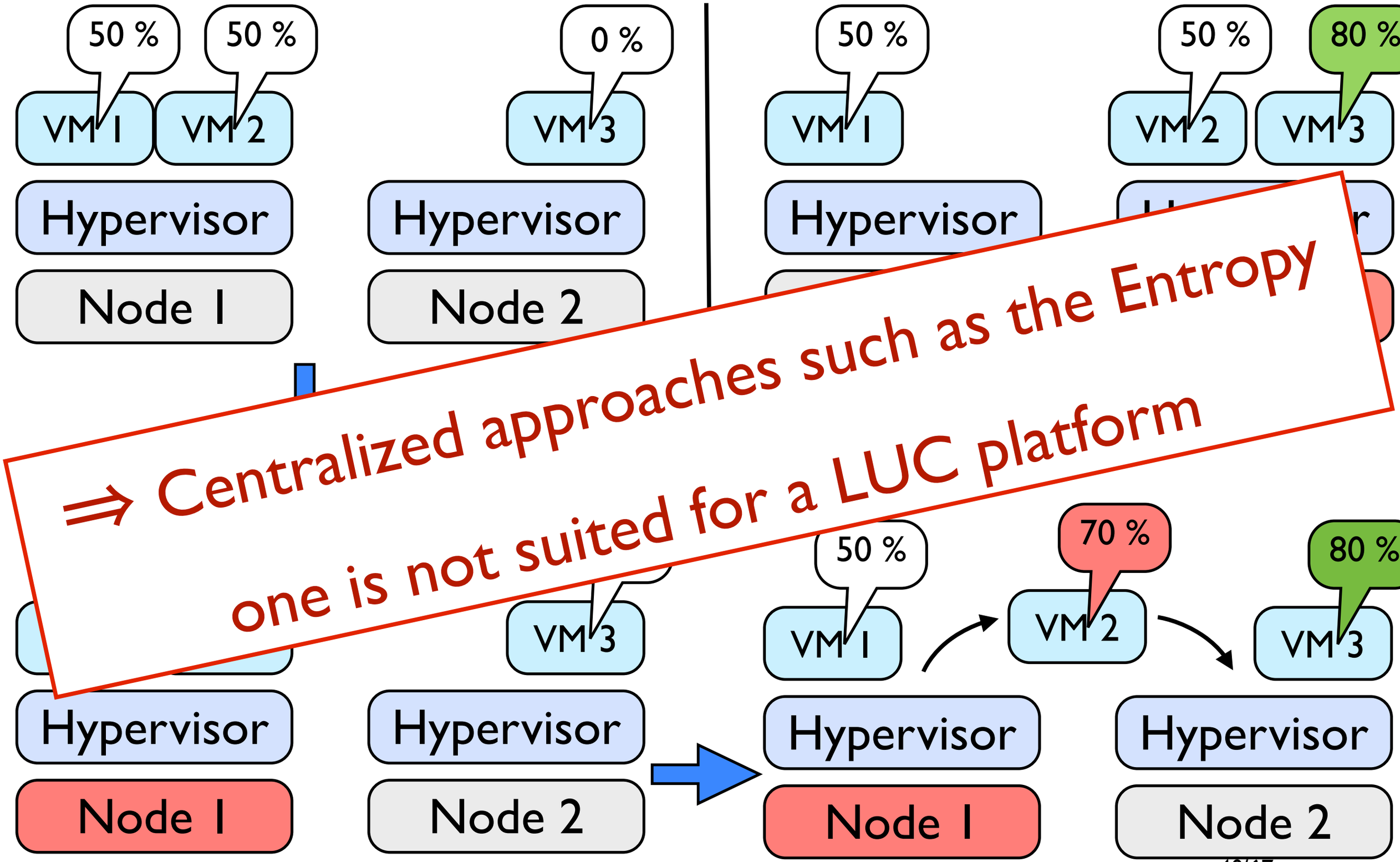


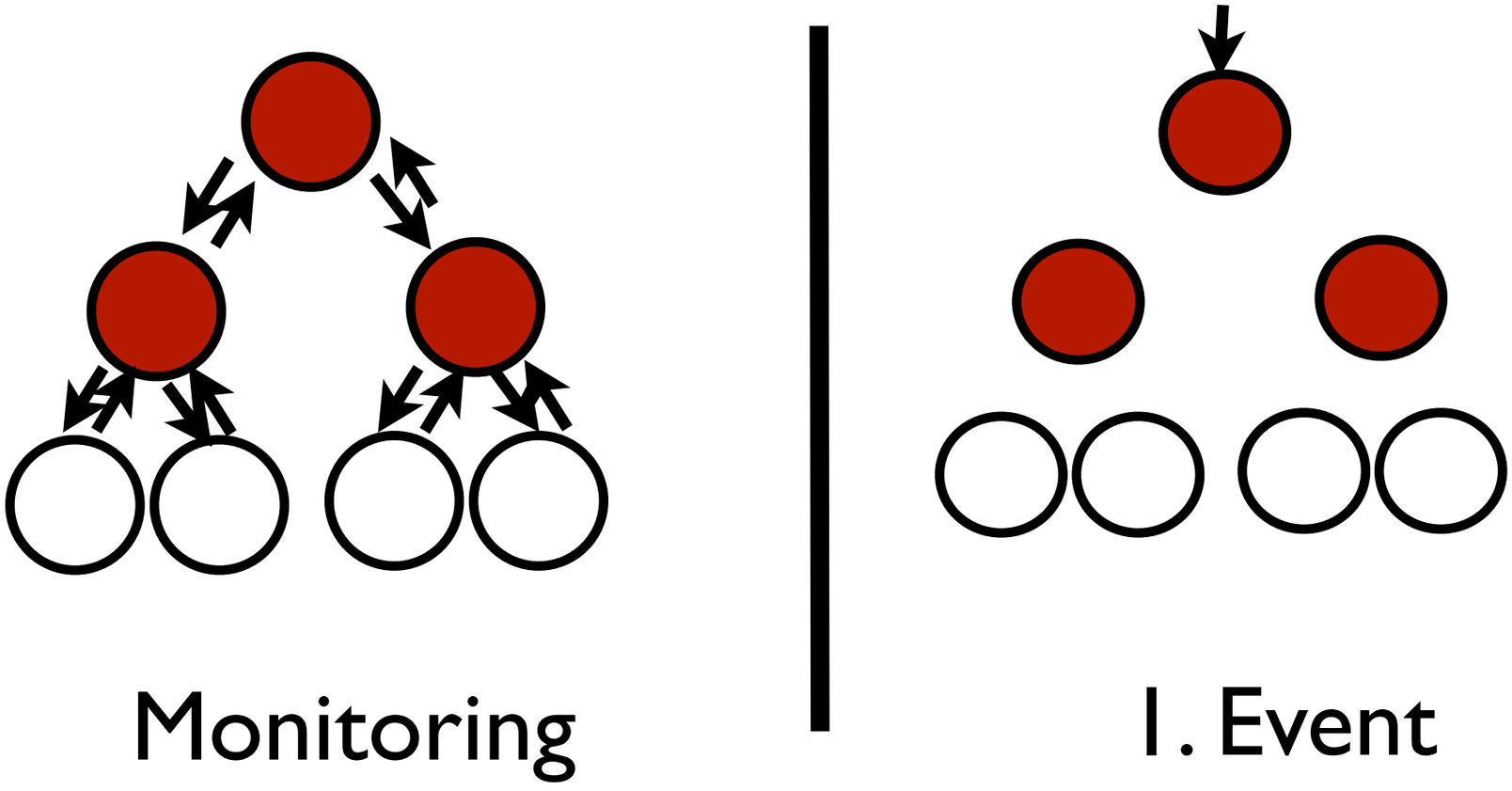






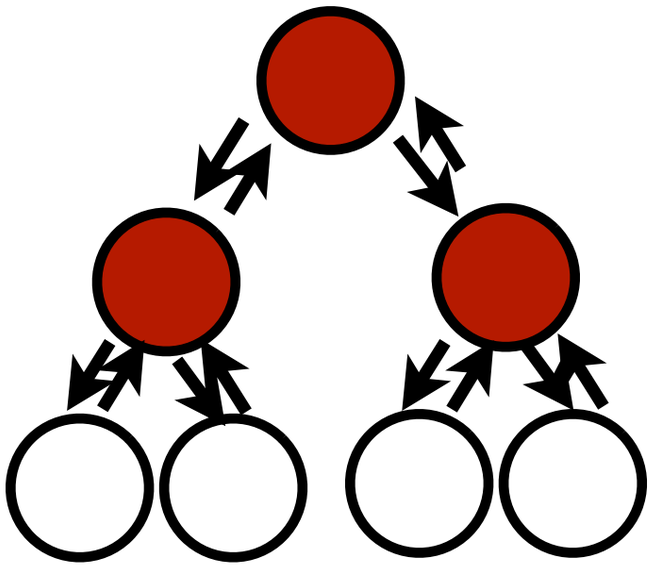
Scalability vs Reactivity



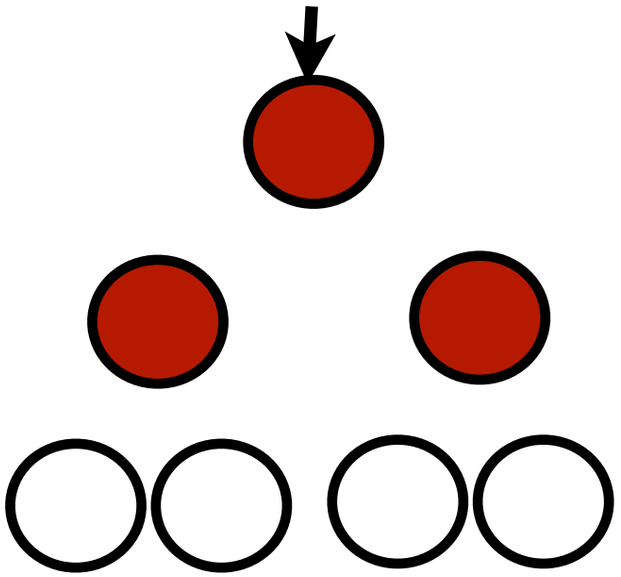


Hierarchical architecture

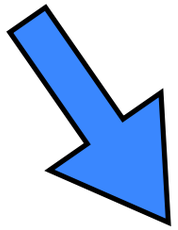
Snooze [Feller et al., CCGRID 12]

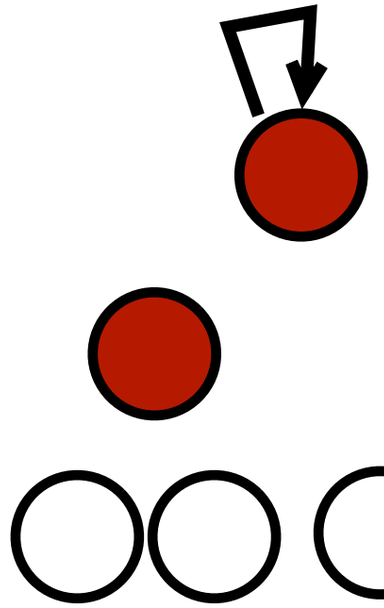


Monitoring



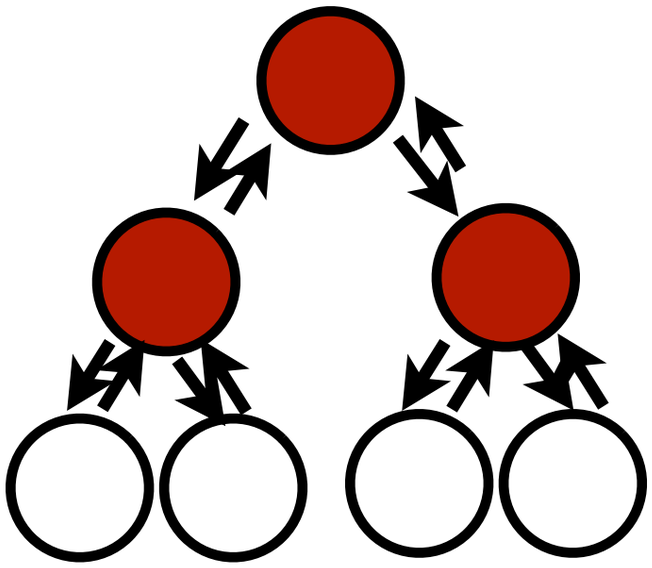
1. Event



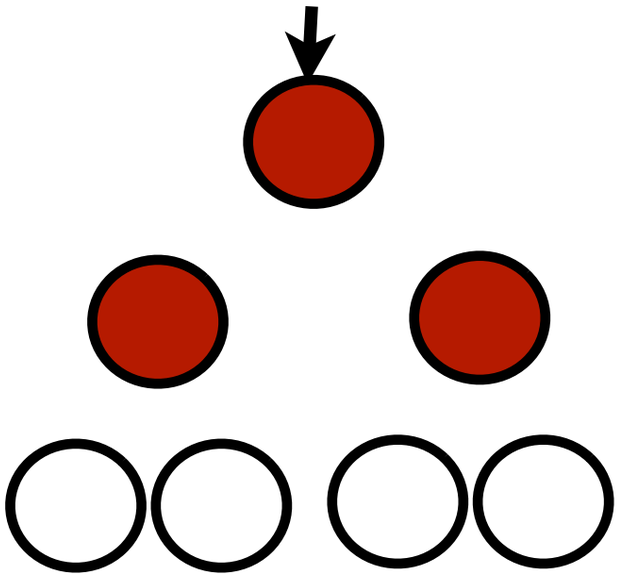


2. Computing

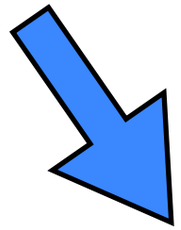
Hierarchical architecture
Snooze [Feller et al., CCGRID 12]

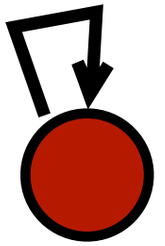


Monitoring

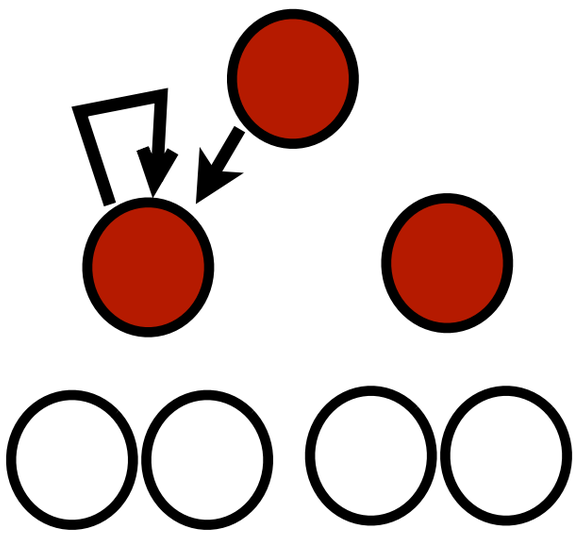


1. Event

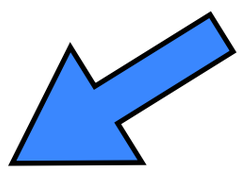




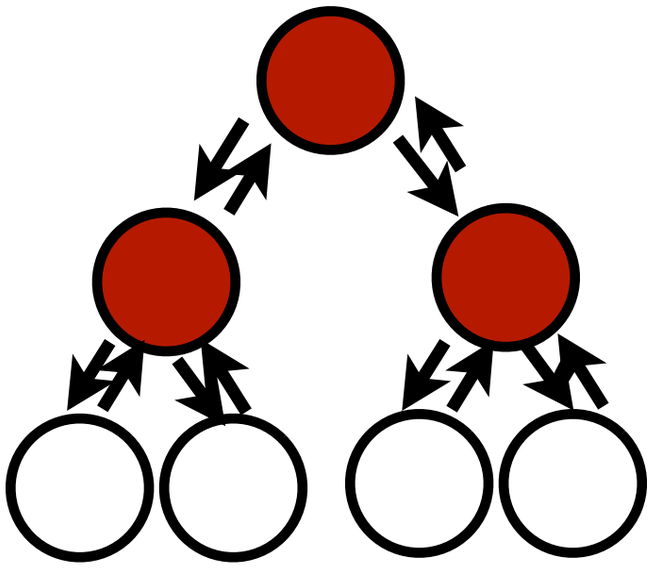
2. Computing



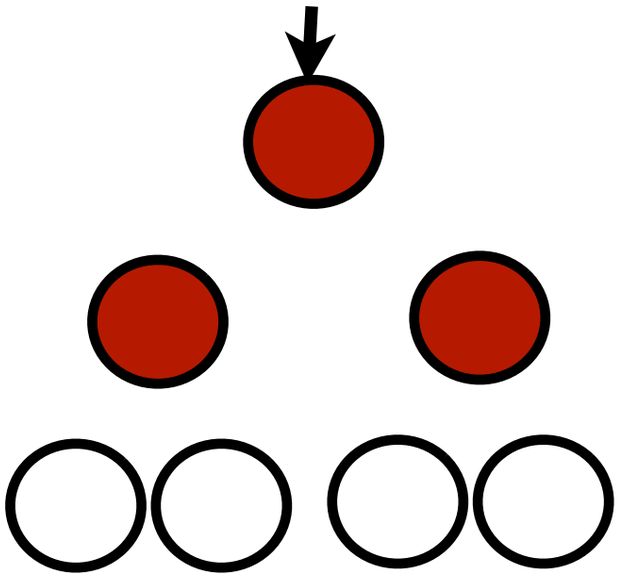
3. Computing



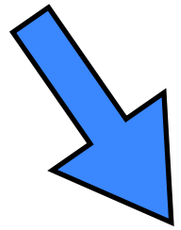
Hierarchical architecture
Snooze [Feller et al., CCGRID 12]

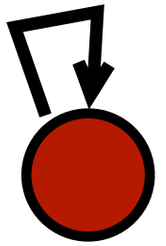


Monitoring

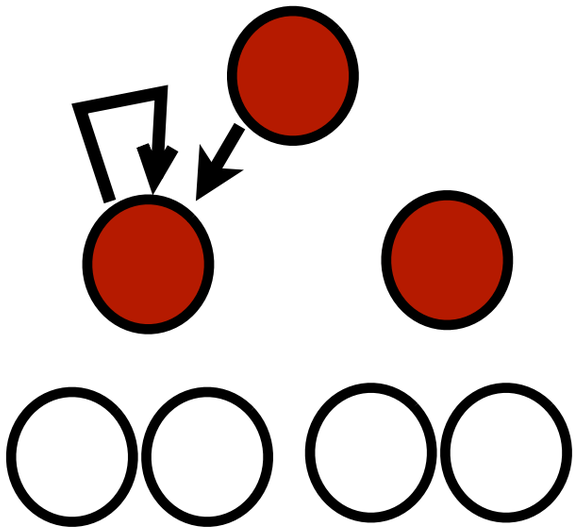


1. Event

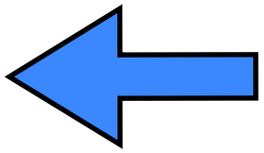


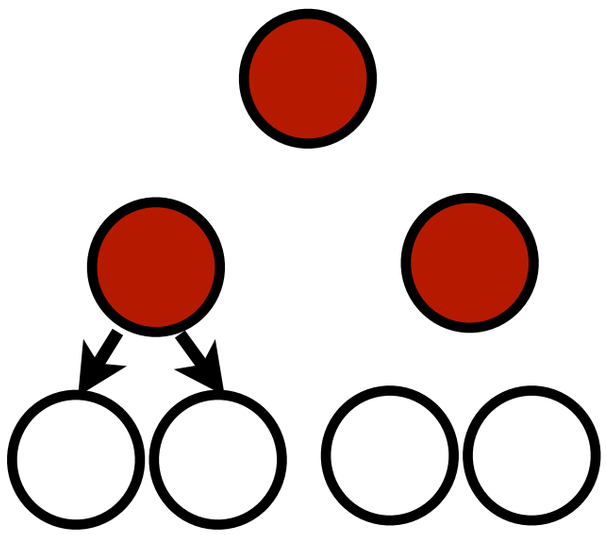


2. Computing



3. Computing





4. Reconfiguring

Hierarchical architecture
Snooze [Feller et al., CCGRID 12]

The LUC OS - VEs Scheduling

- Make dynamic partitioning of the system according to the effective usage of resources
- Make direct cooperations between hypervisors (no service node)
- The DVMS Proposal

Event driven

P2P Like system

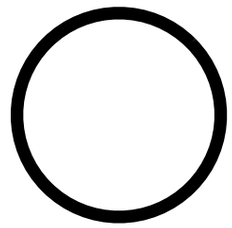
Local interactions between nodes

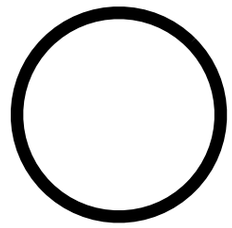
Scheduling performed on partitions of the system, created dynamically (nodes are reserved for an exclusive use by a scheduler, to prevent several schedulers from migrating the same VMs)

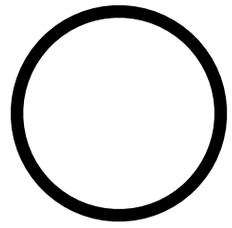
The LUC OS - VEs Scheduling

Event occurs on node$_i$

Event





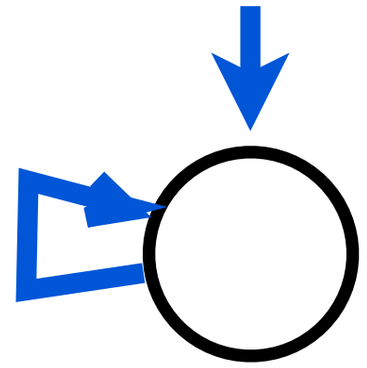


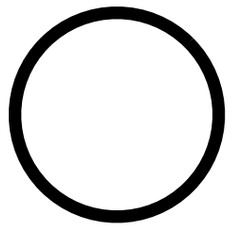
The LUC OS - VEs Scheduling

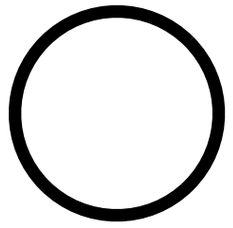
Event occurs on $node_i$

Can current node scheduler calculate valid schedule?

Event







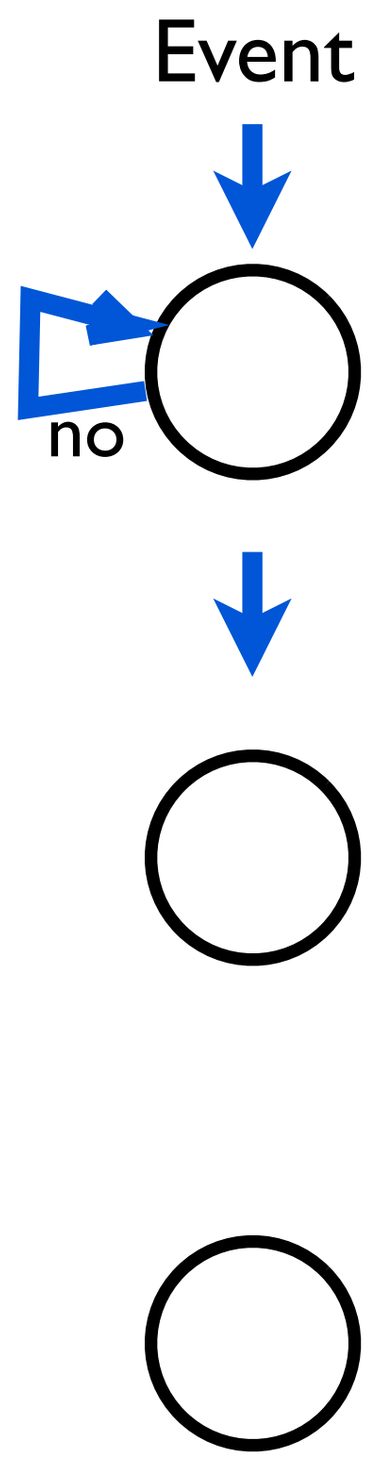
The LUC OS - VEs Scheduling

Event occurs on node$_i$

Can current node scheduler calculate valid schedule?

no

Contact neighbor and ask it to solve the problem



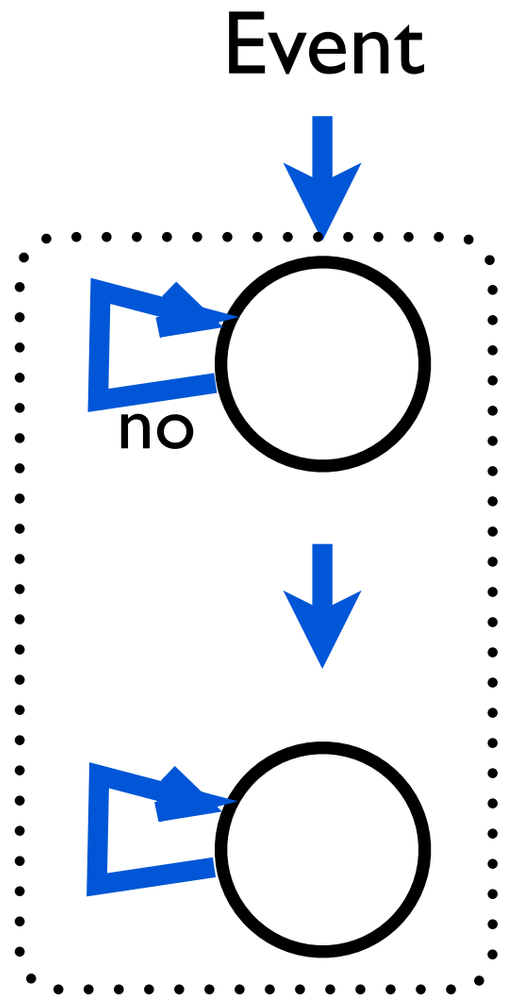
The LUC OS - VEs Scheduling

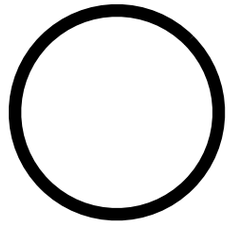
Event occurs on node$_i$

Can current node scheduler calculate valid schedule?

no

Contact neighbor and ask it to solve the problem





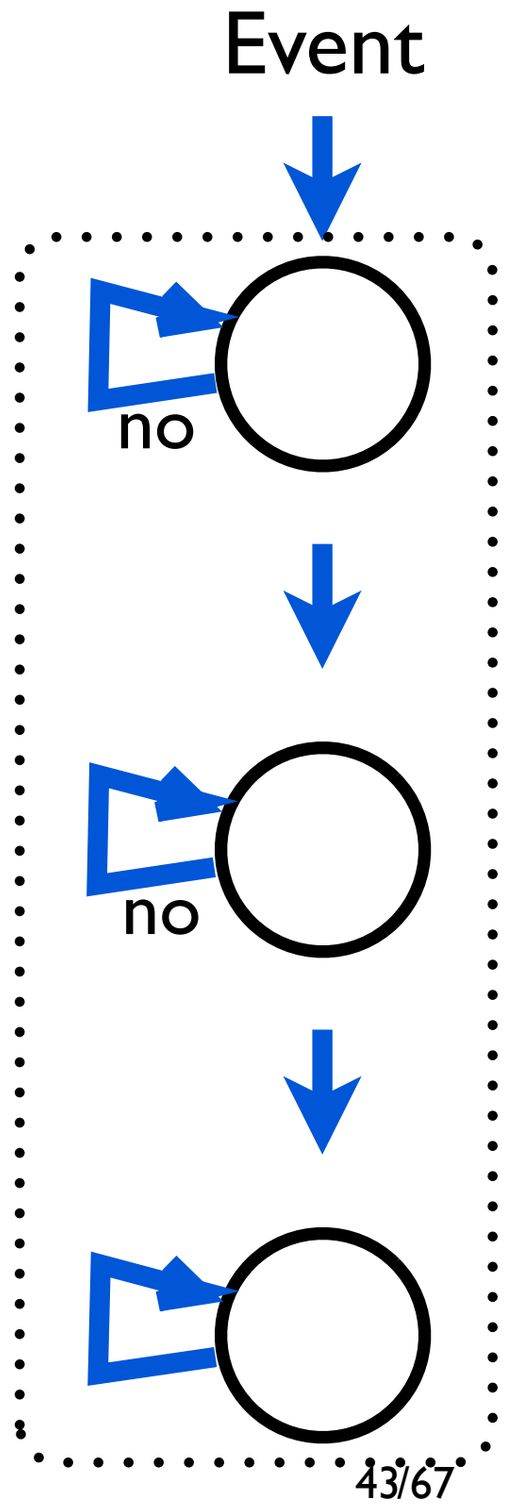
The LUC OS - VEs Scheduling

Event occurs on node$_i$

Can current node scheduler calculate valid schedule?

no

Contact neighbor and ask it to solve the problem



The LUC OS - VEs Scheduling

Event occurs on node$_i$

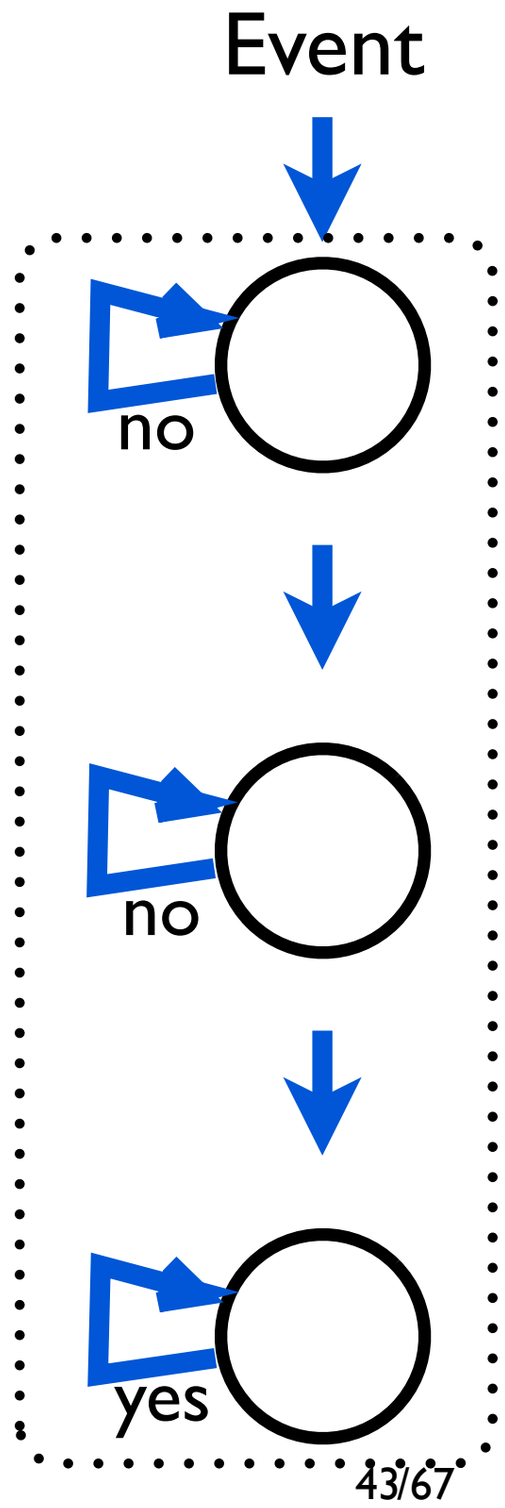
Can current node scheduler calculate valid schedule?

no

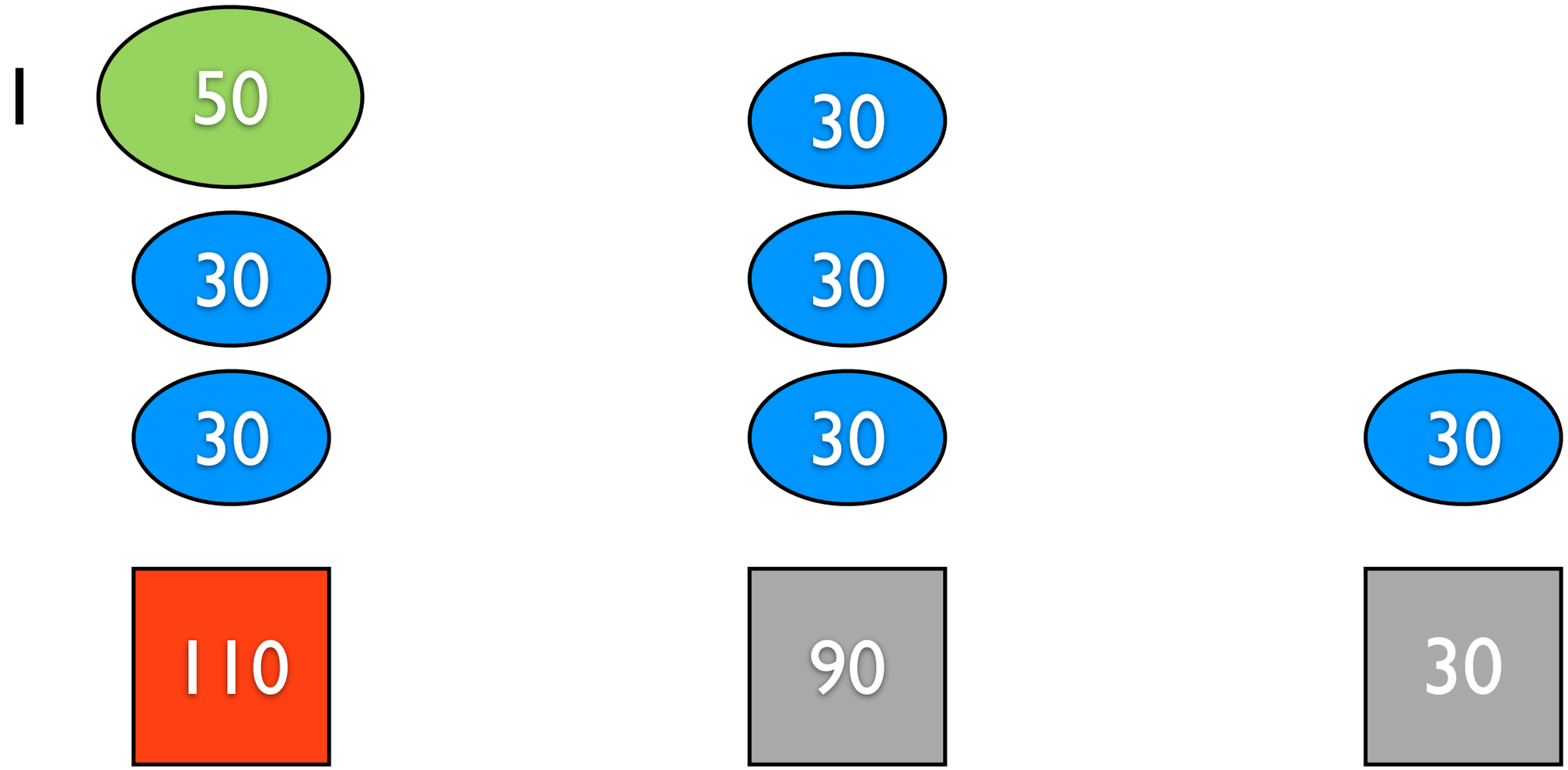
Contact neighbor and ask it to solve the problem

yes

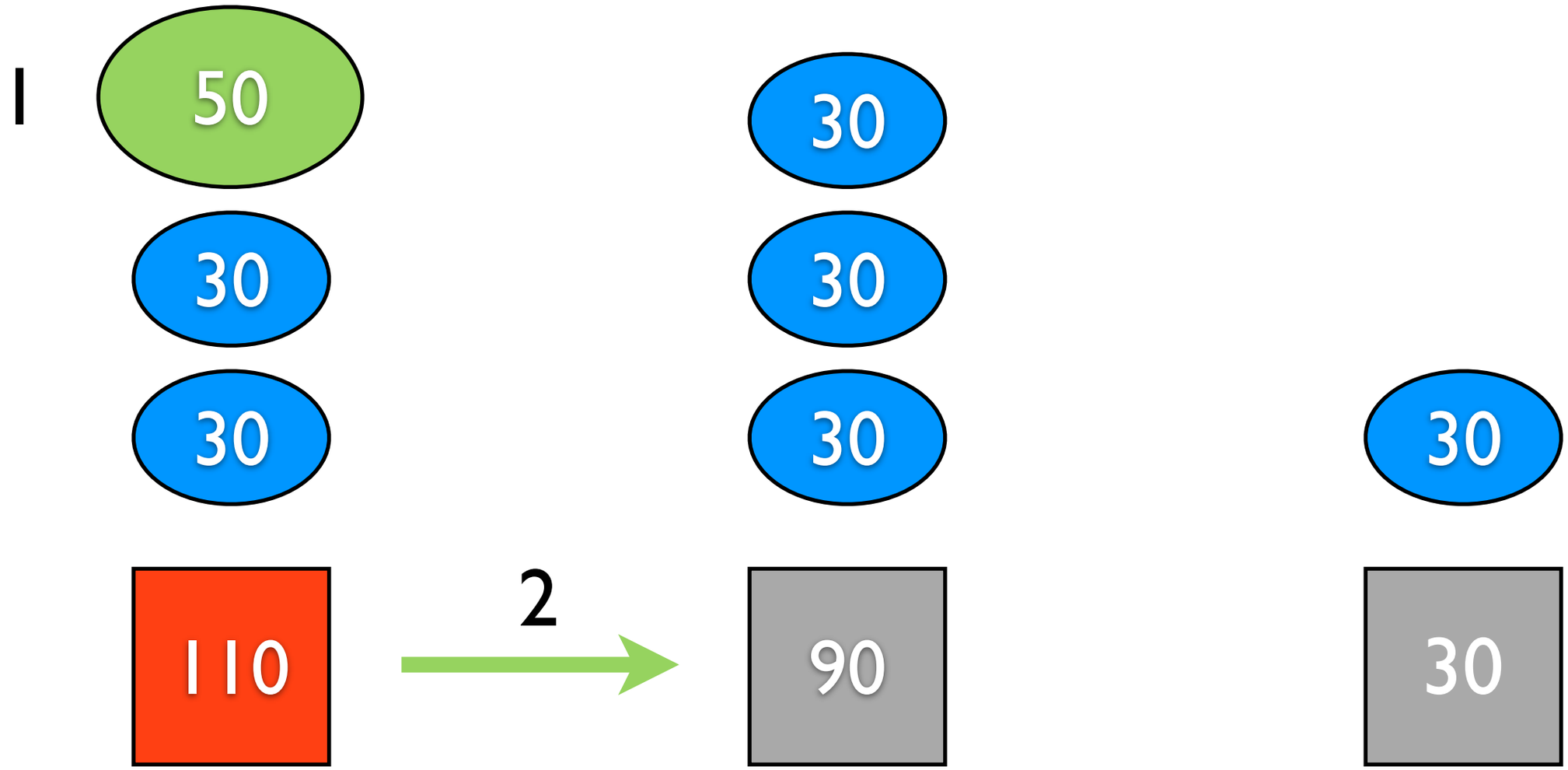
Apply the schedule



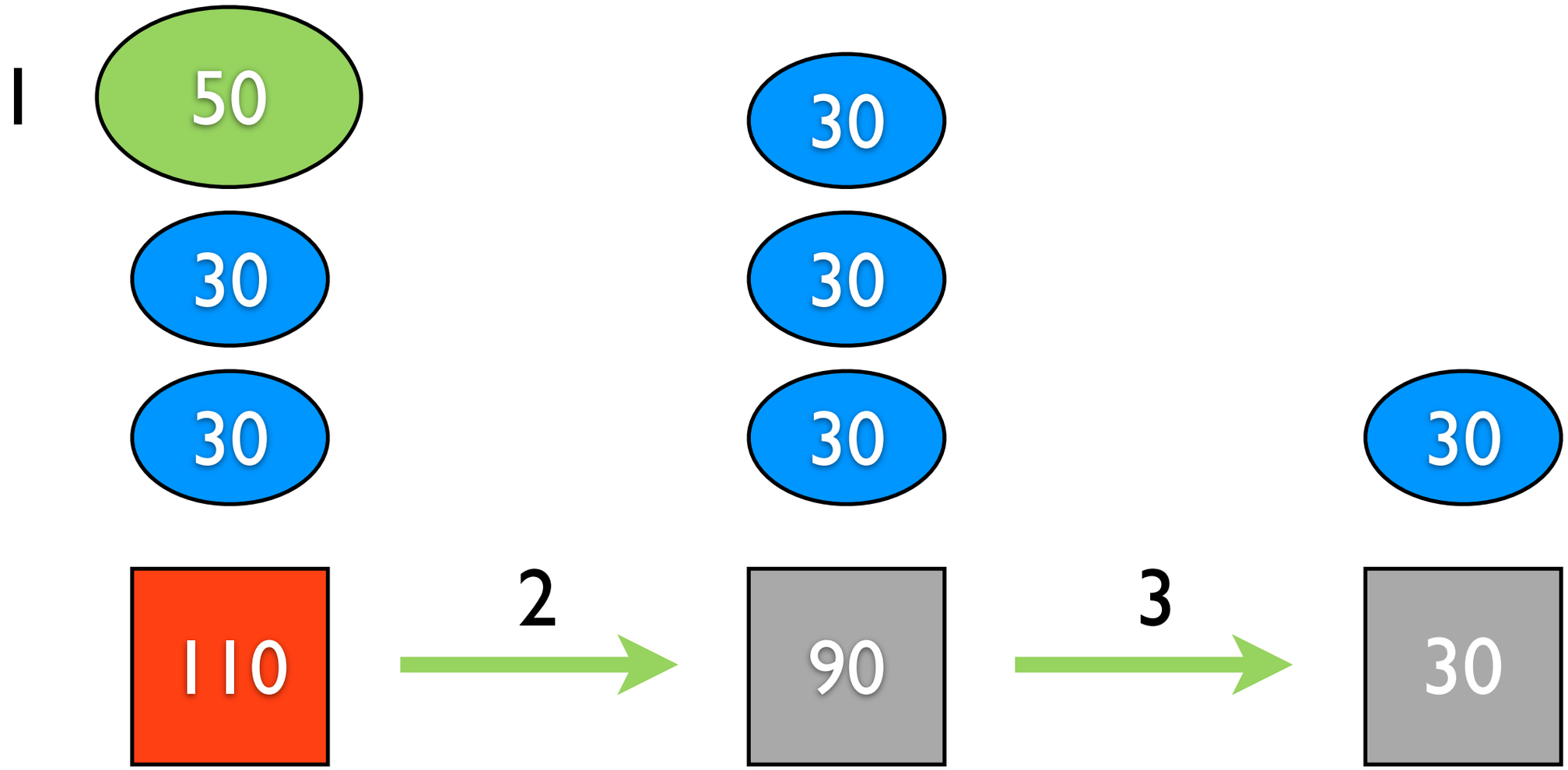
Example 1: overloaded event



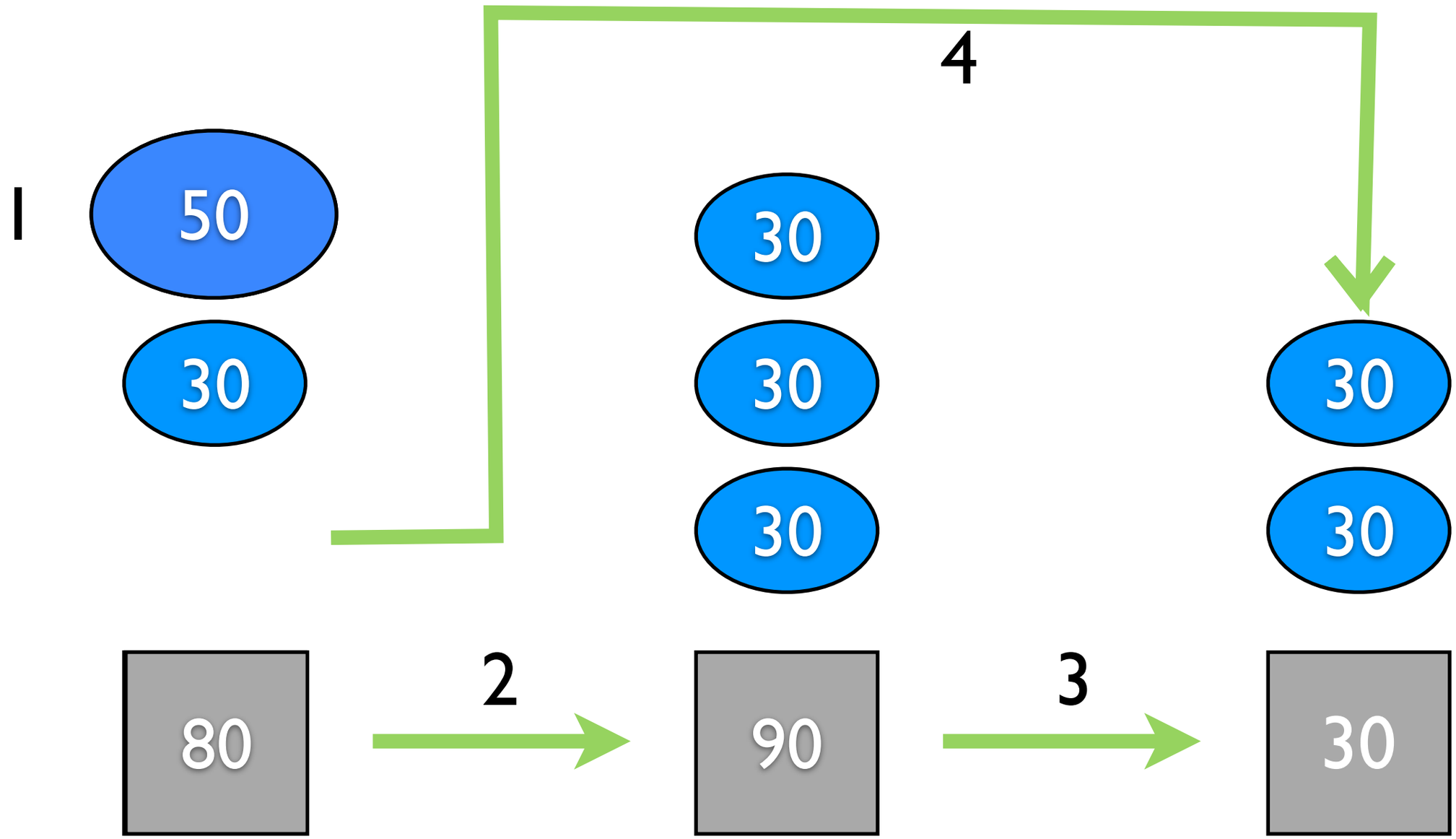
Example 1: overloaded event



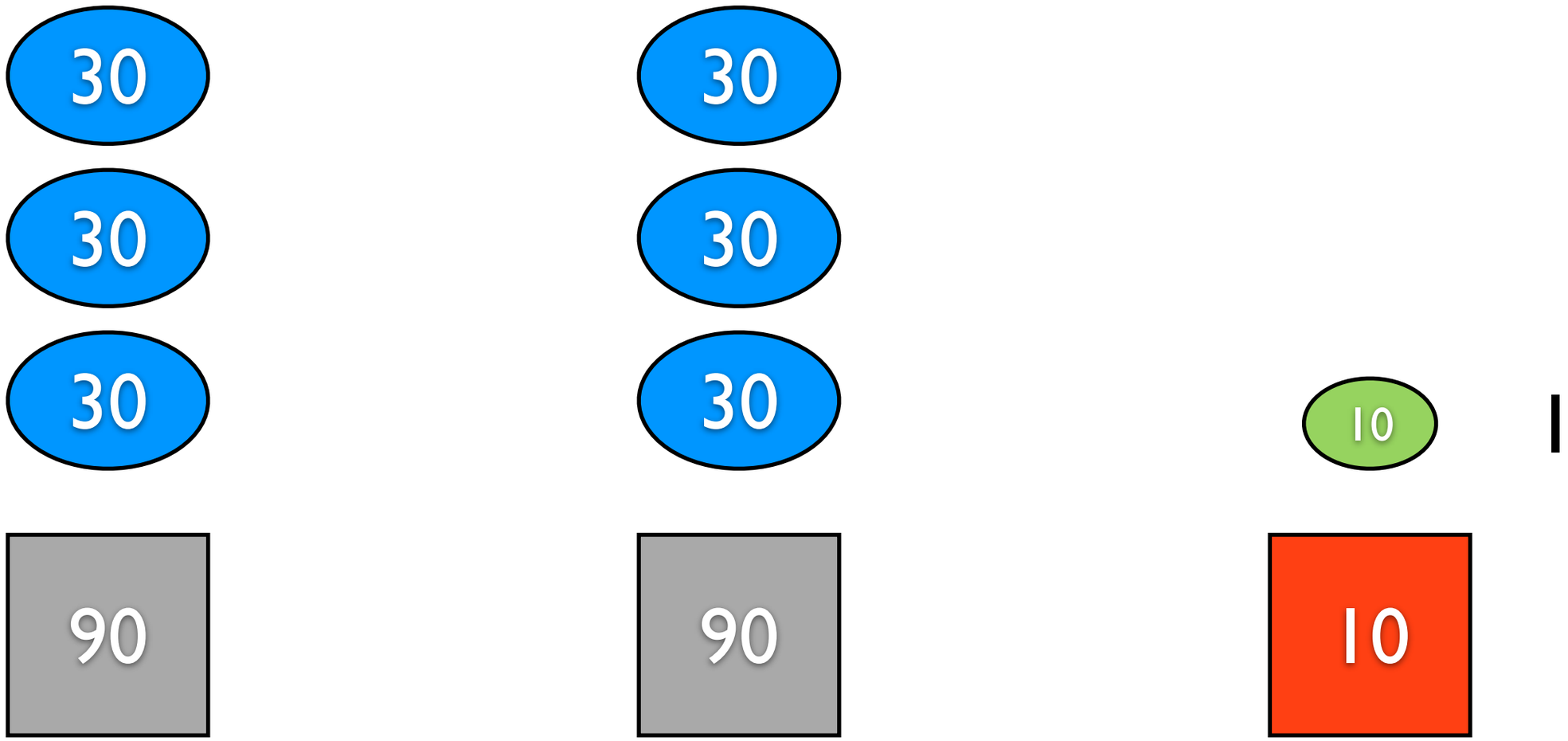
Example 1: overloaded event



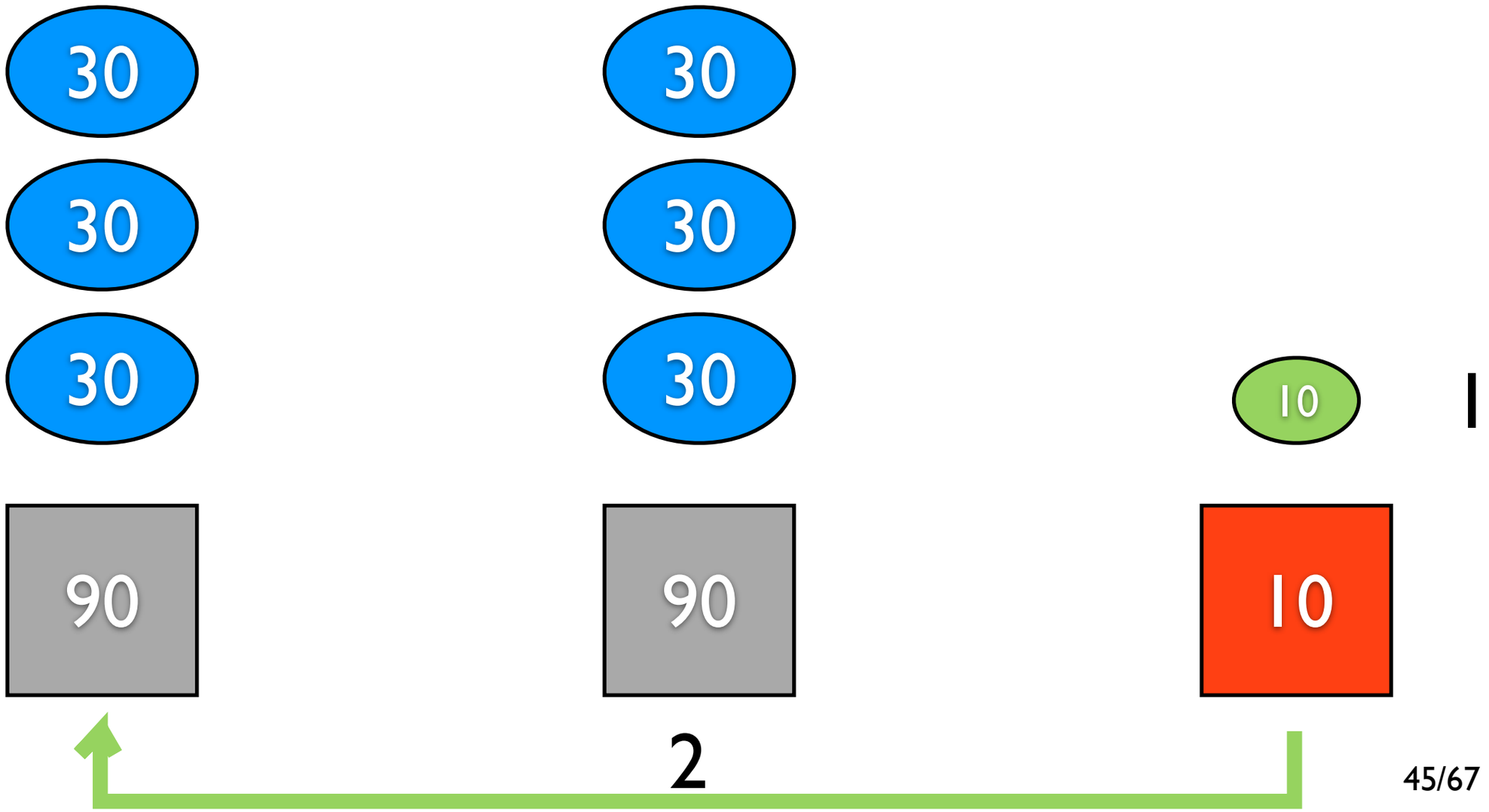
Example 1: overloaded event



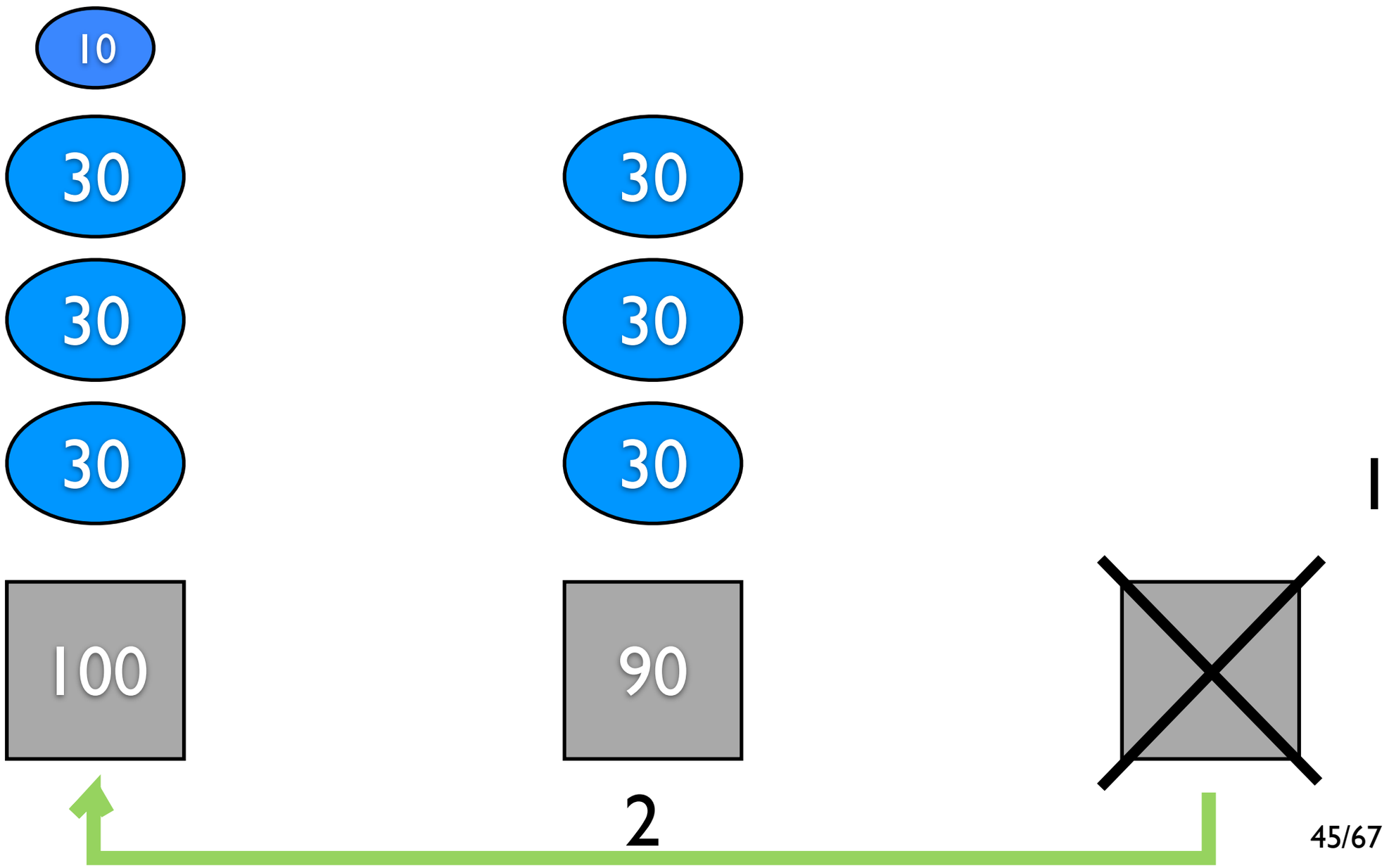
Example 2: underloaded event



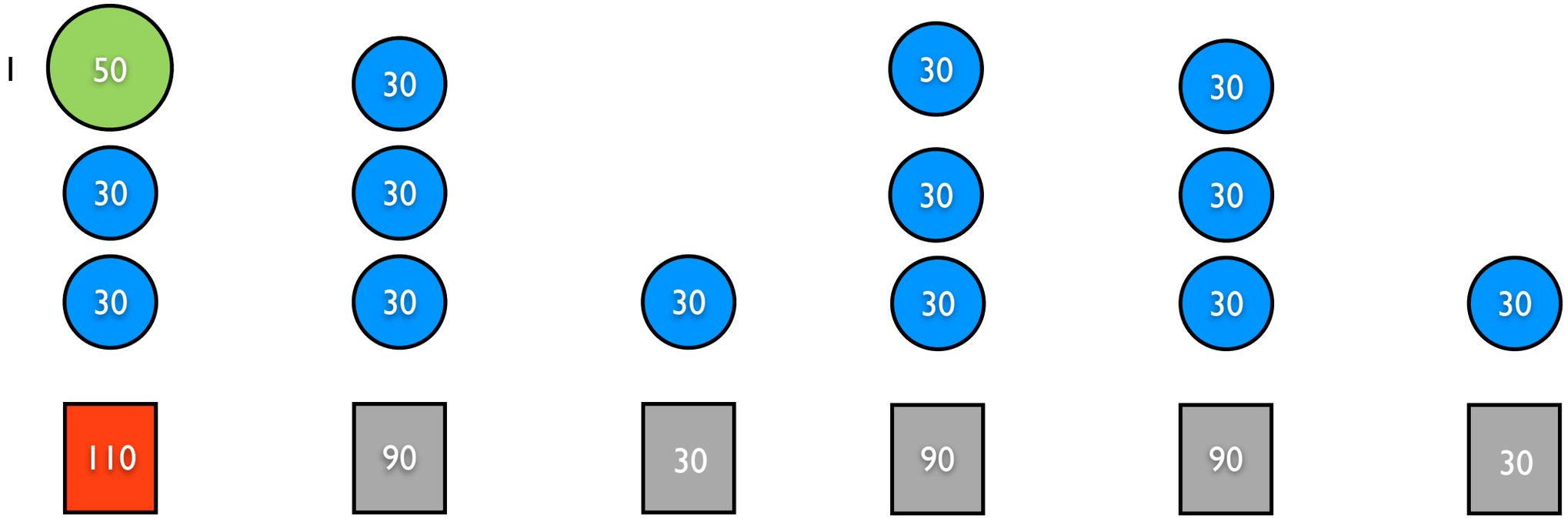
Example 2: underloaded event



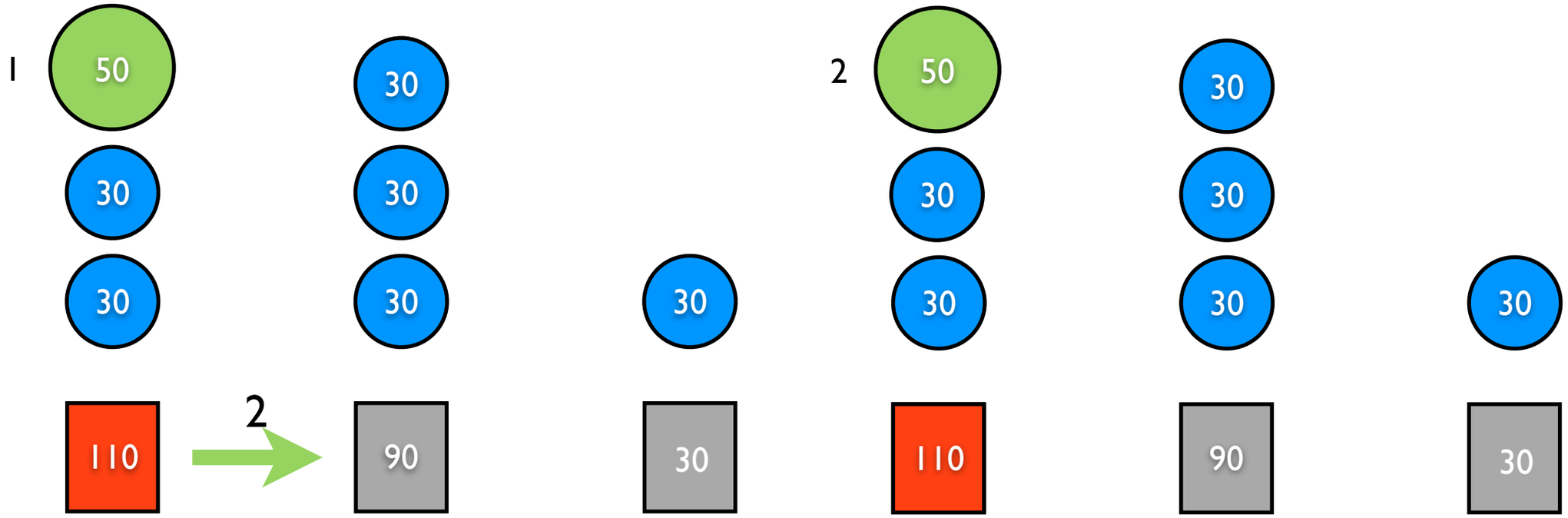
Example 2: underloaded event



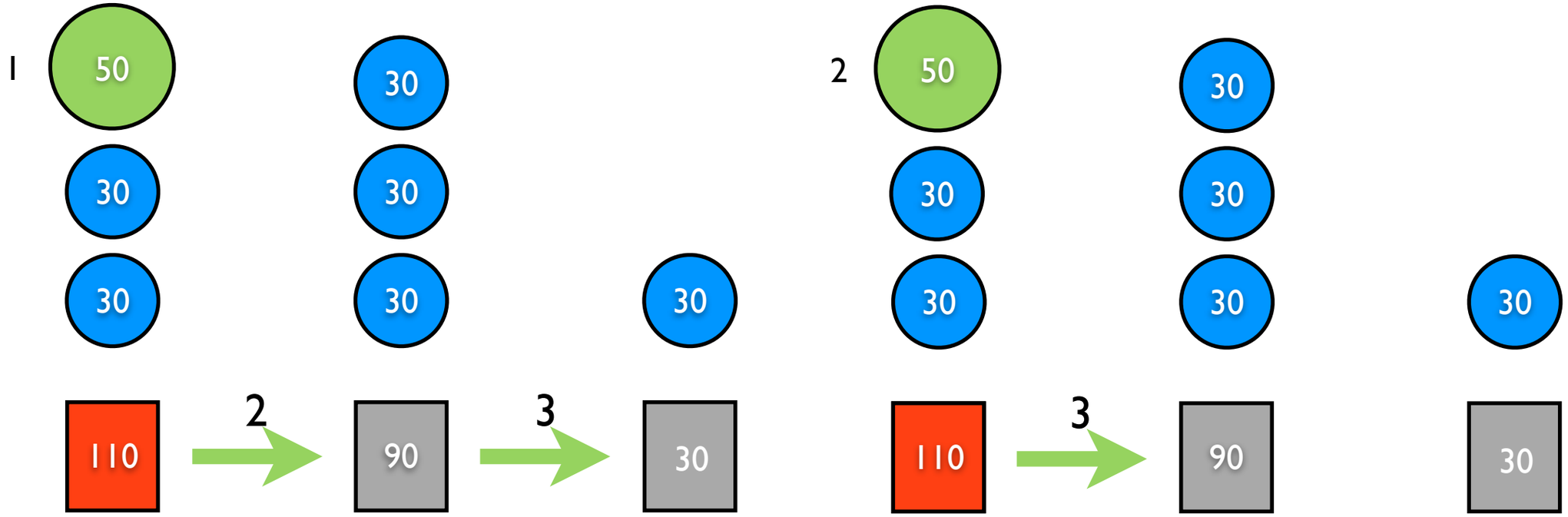
Example 3: shifted overloaded events



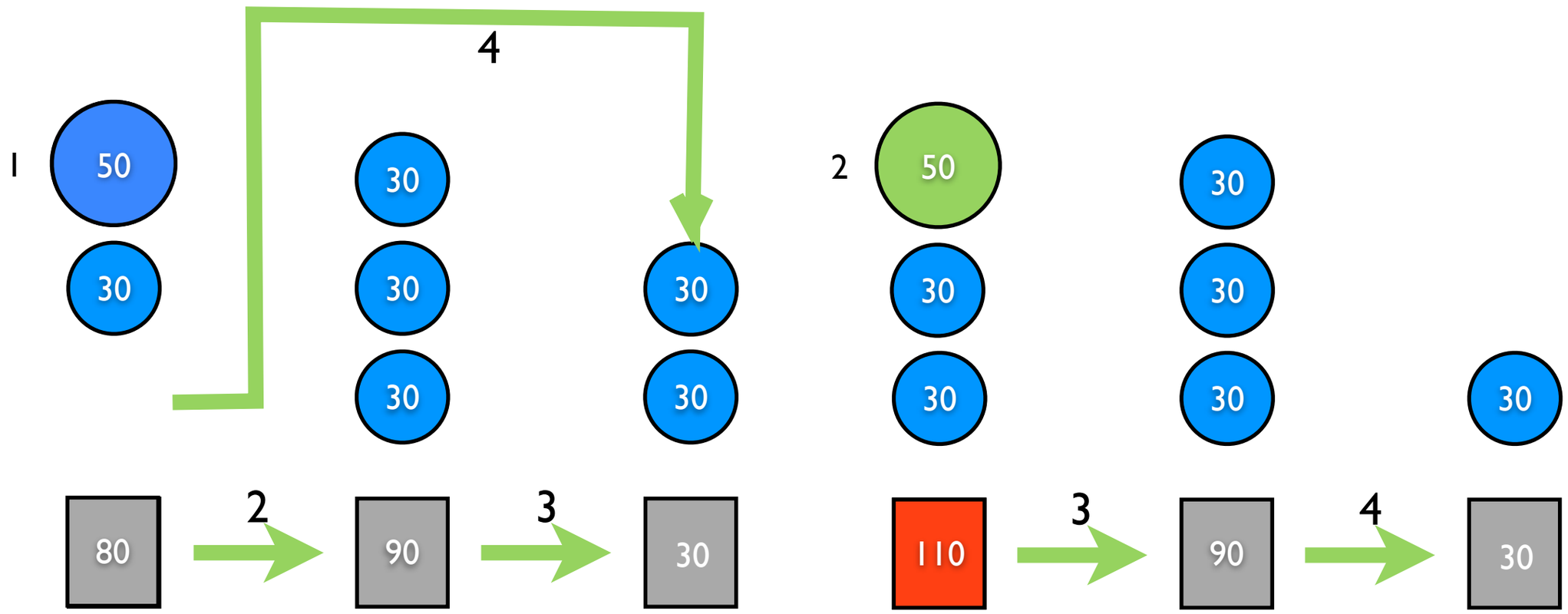
Example 3: shifted overloaded events



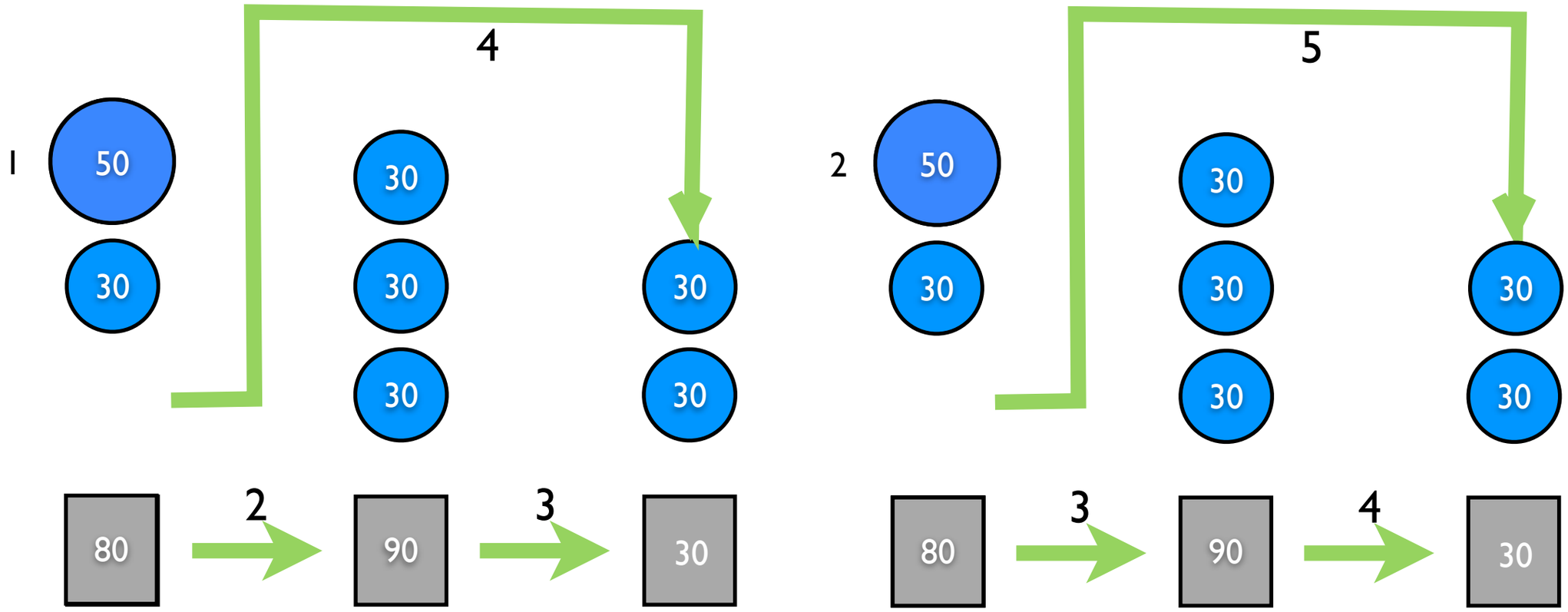
Example 3: shifted overloaded events



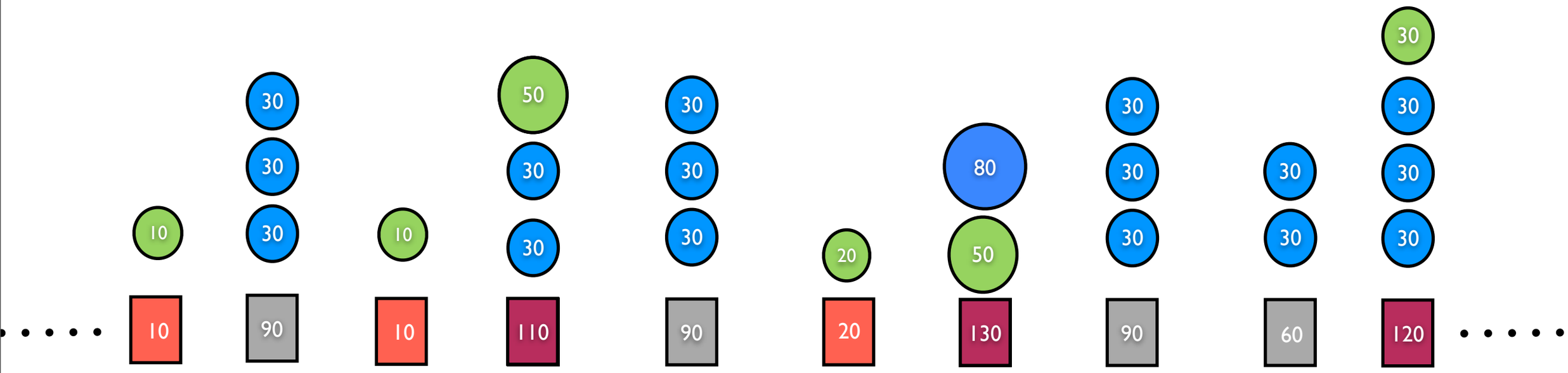
Example 3: shifted overloaded events



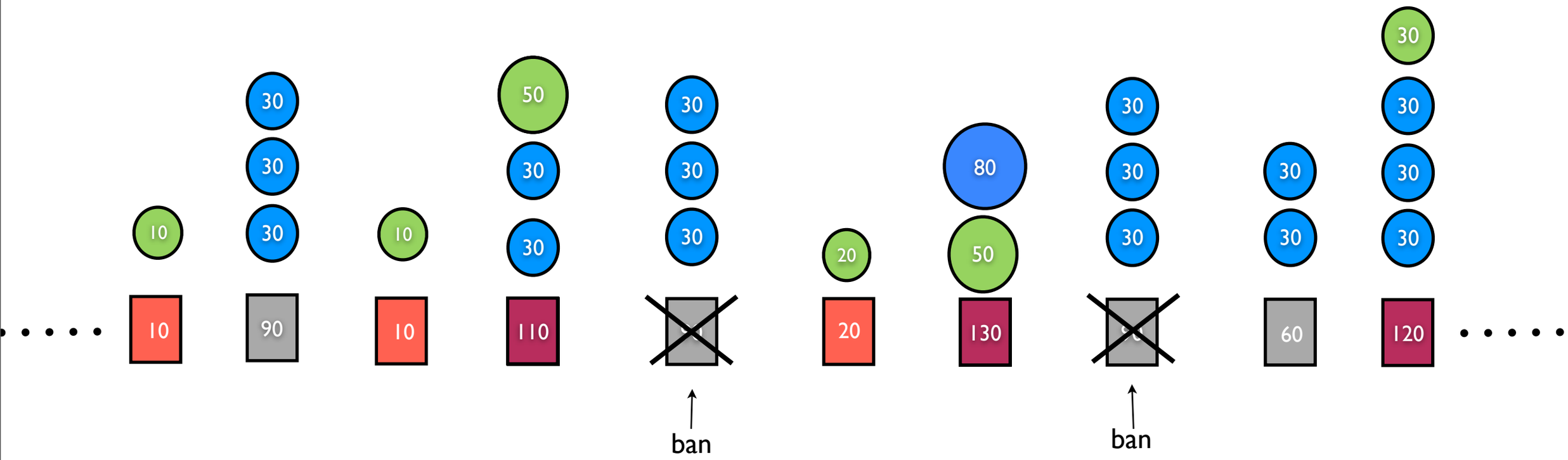
Example 3: shifted overloaded events



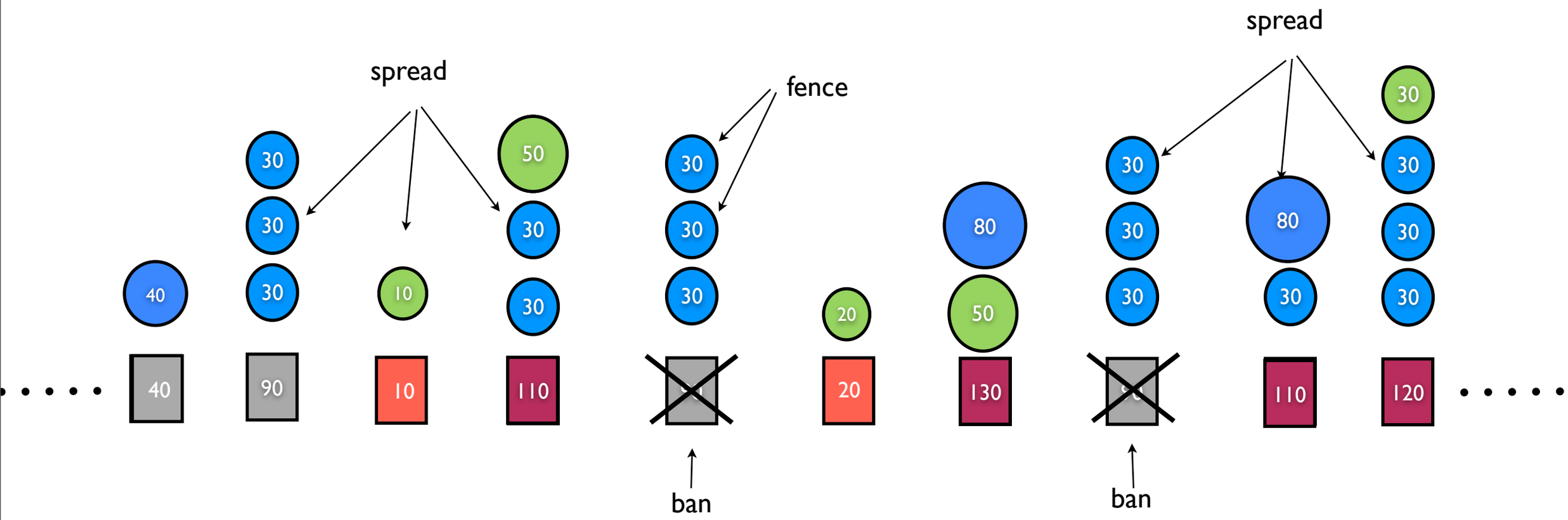
Example 4: your turn ?



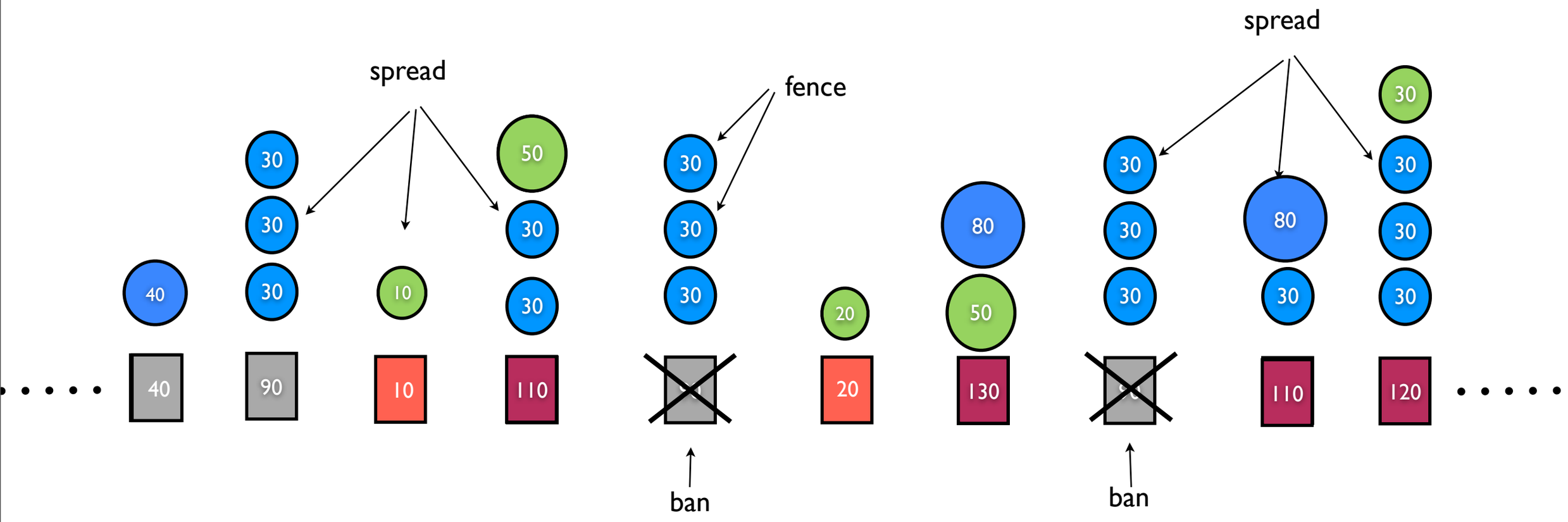
Example 4: your turn ?



Example 4: your turn ?



Example 4: your turn ?

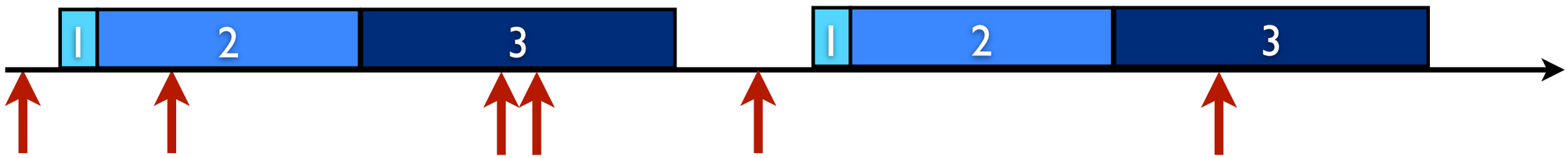


Only CPU is considered in this simple example

The LUC OS - VEs Scheduling

- POC (leveraging Entropy to solve non viable configuration)
- 100K VMs / 10K PMs (simulation using the SimGrid framework)
- Load injector

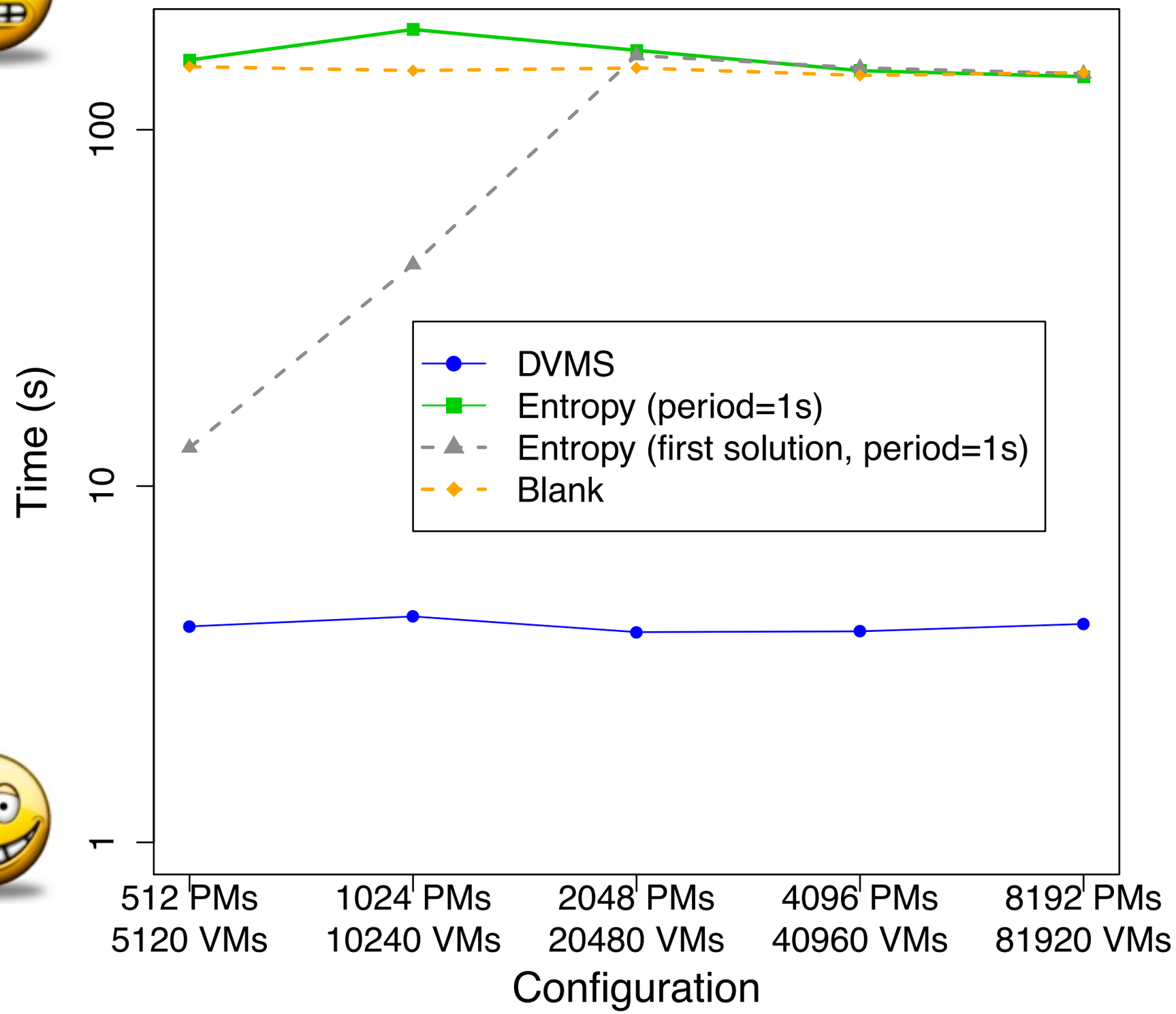
  Events based on an Exponential law
  (λ = Nb VMs / 300 sec)



  VM Load based on a Gaussian Law
  (μ = 70 / σ = 30)

- Simulation confirmed through in vivo experiments (leveraging a JAVA POC and manipulating10K VMs / 512 PMs on G5K)

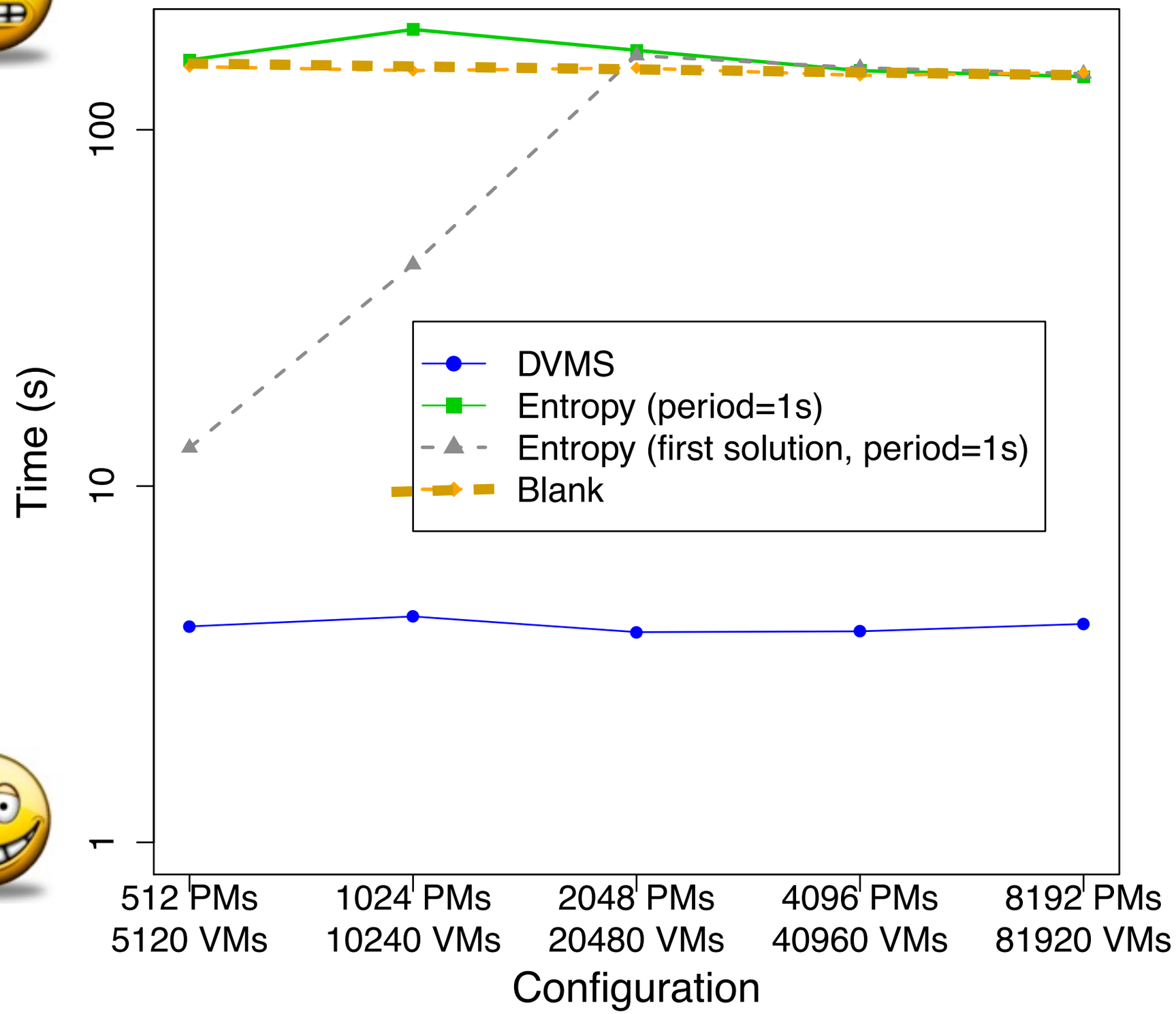
Violation Time Per Node



AVG Cluster Load
85 %

Simulation Time
3600 sec

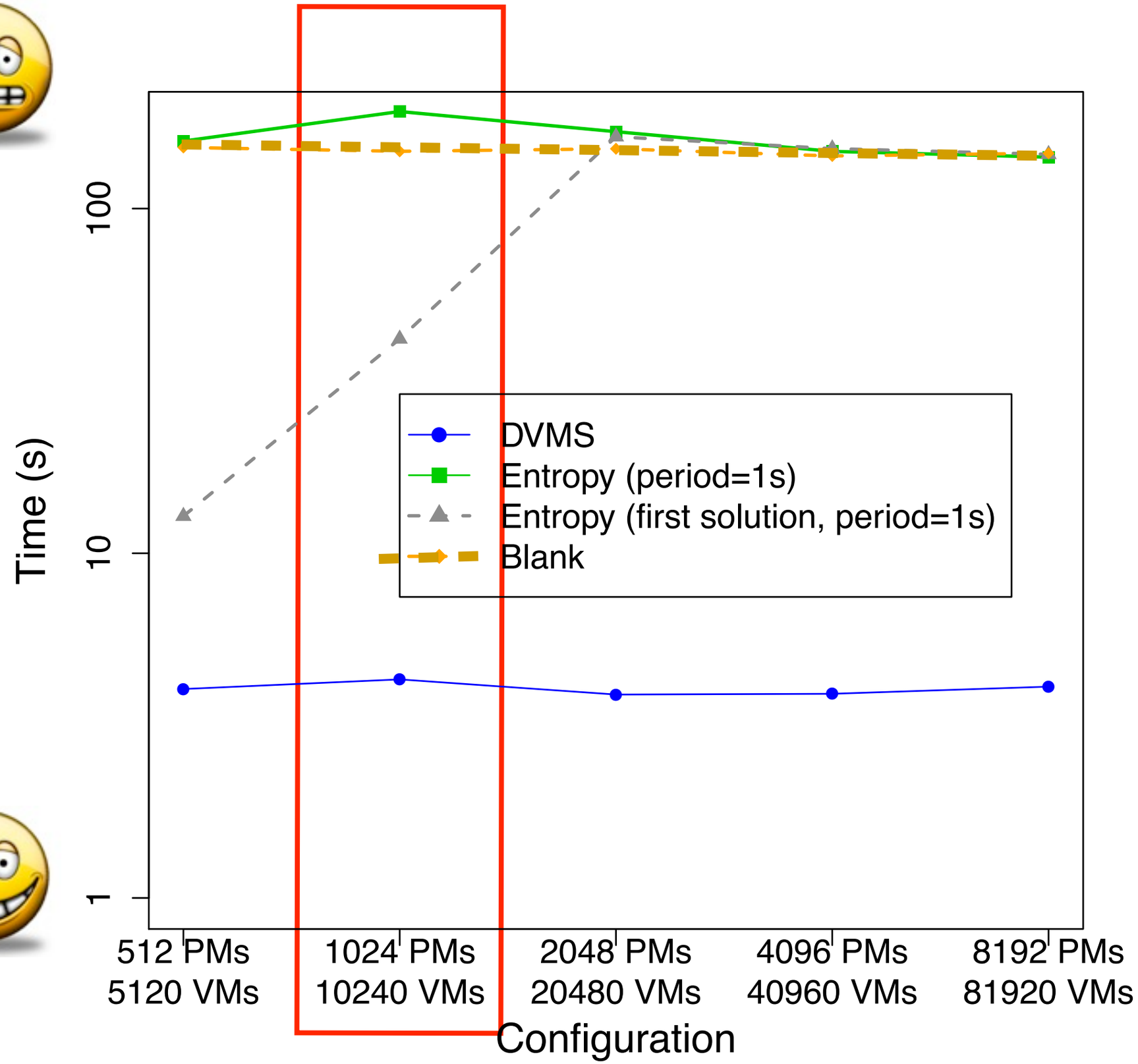
Violation Time Per Node



AVG Cluster Load
85 %

Simulation Time
3600 sec

Violation Time Per Node

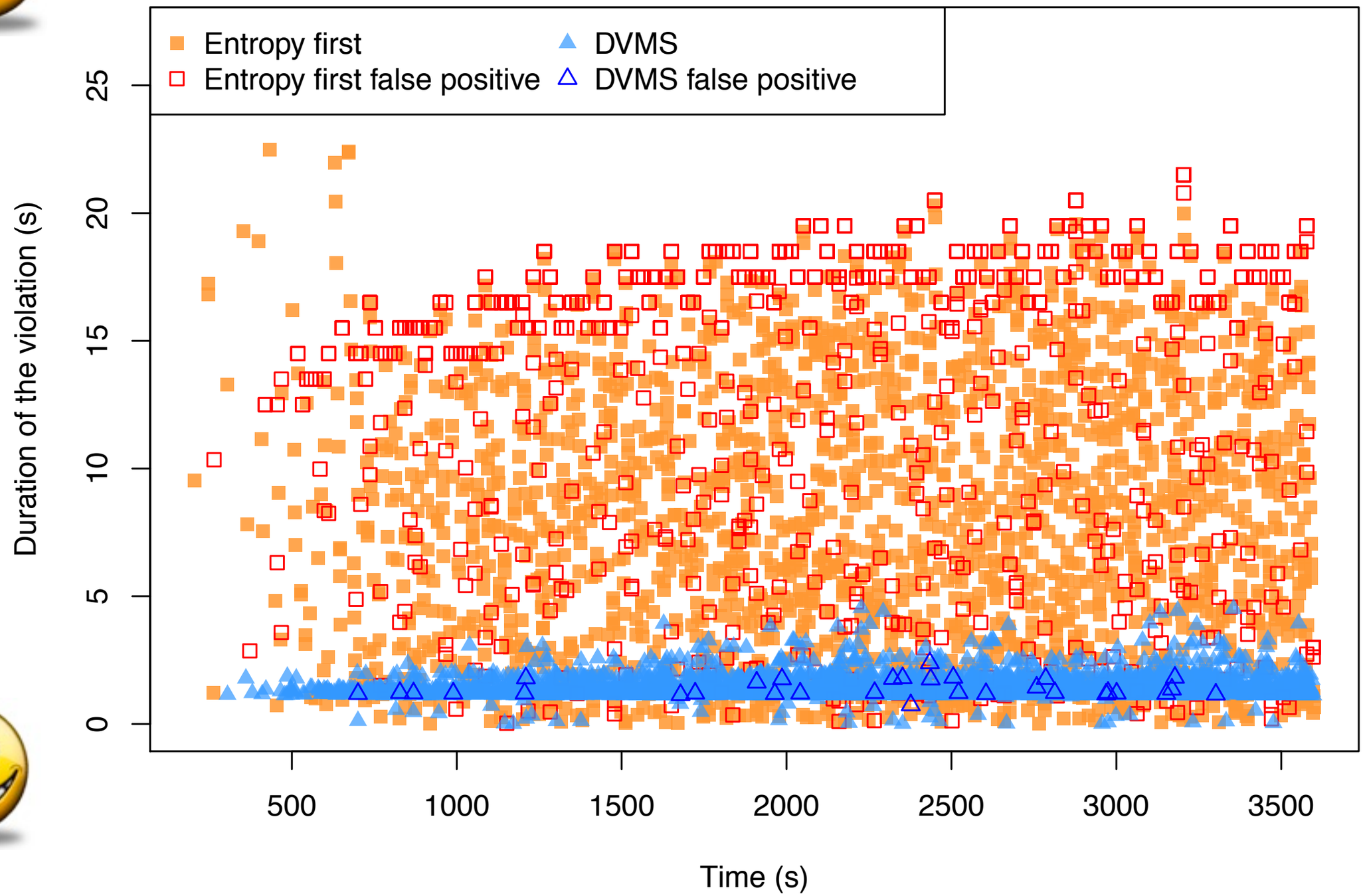


AVG Cluster Load
85 %

Simulation Time
3600 sec

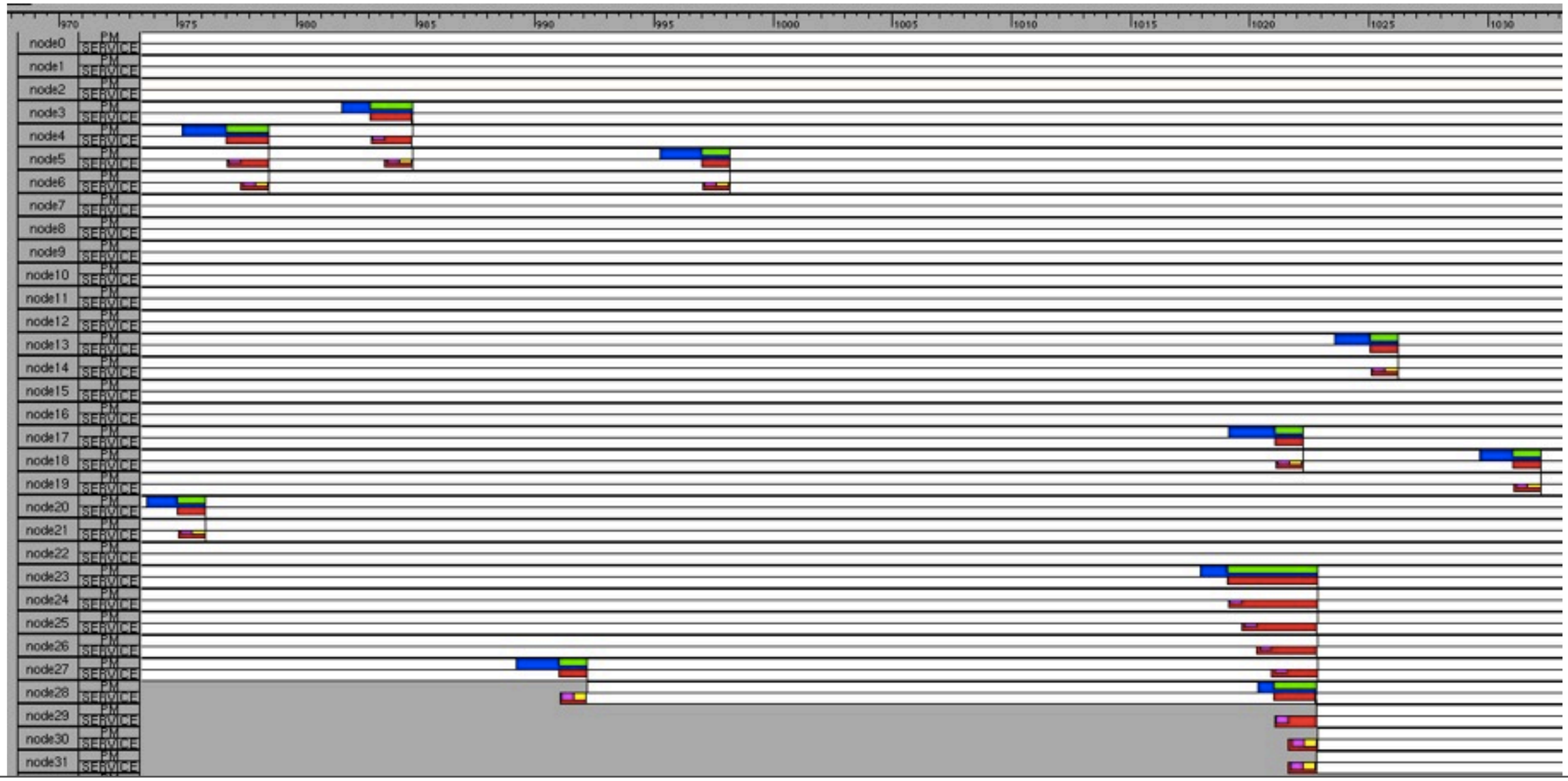
SimGrid `PJDUMP` - Devil is in details

10240 VMs / 1024 Nodes



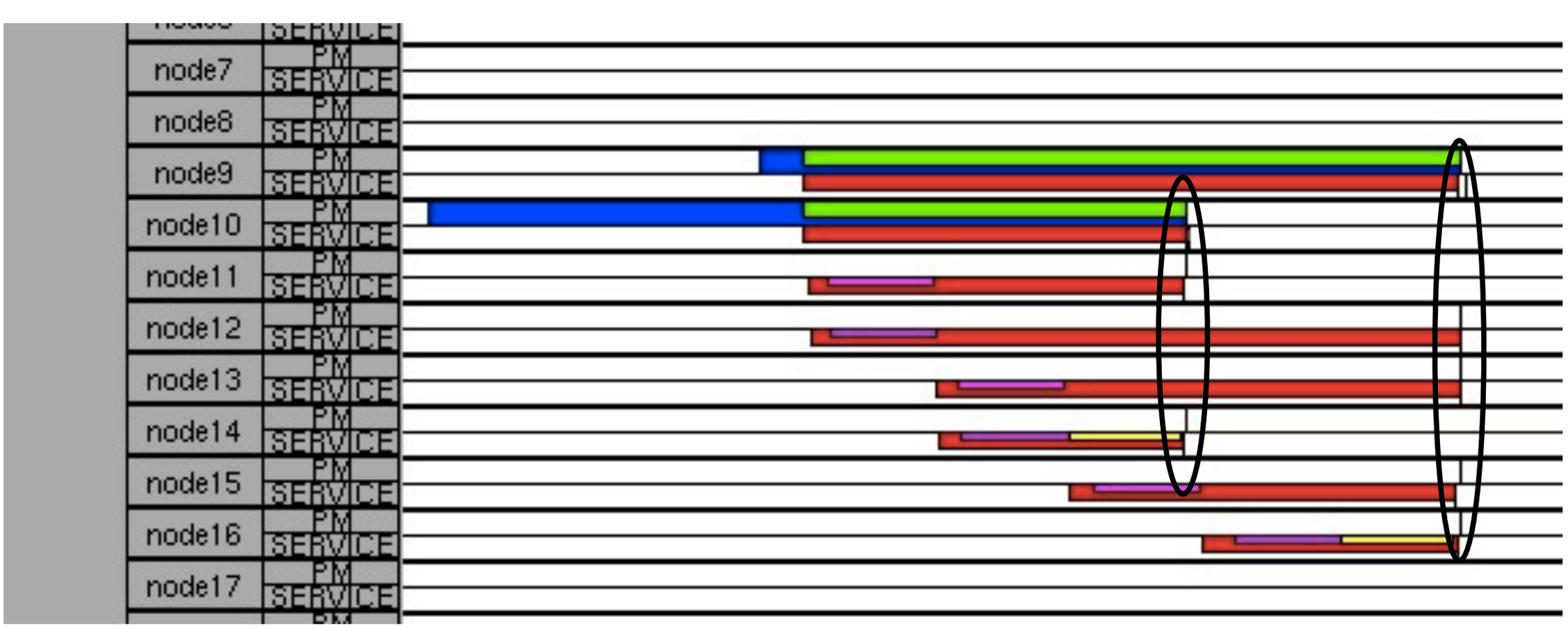
The LUC OS - VEs Scheduling

- Devil is in details
- Paje Traces (collecting during the SimGrid simulation)



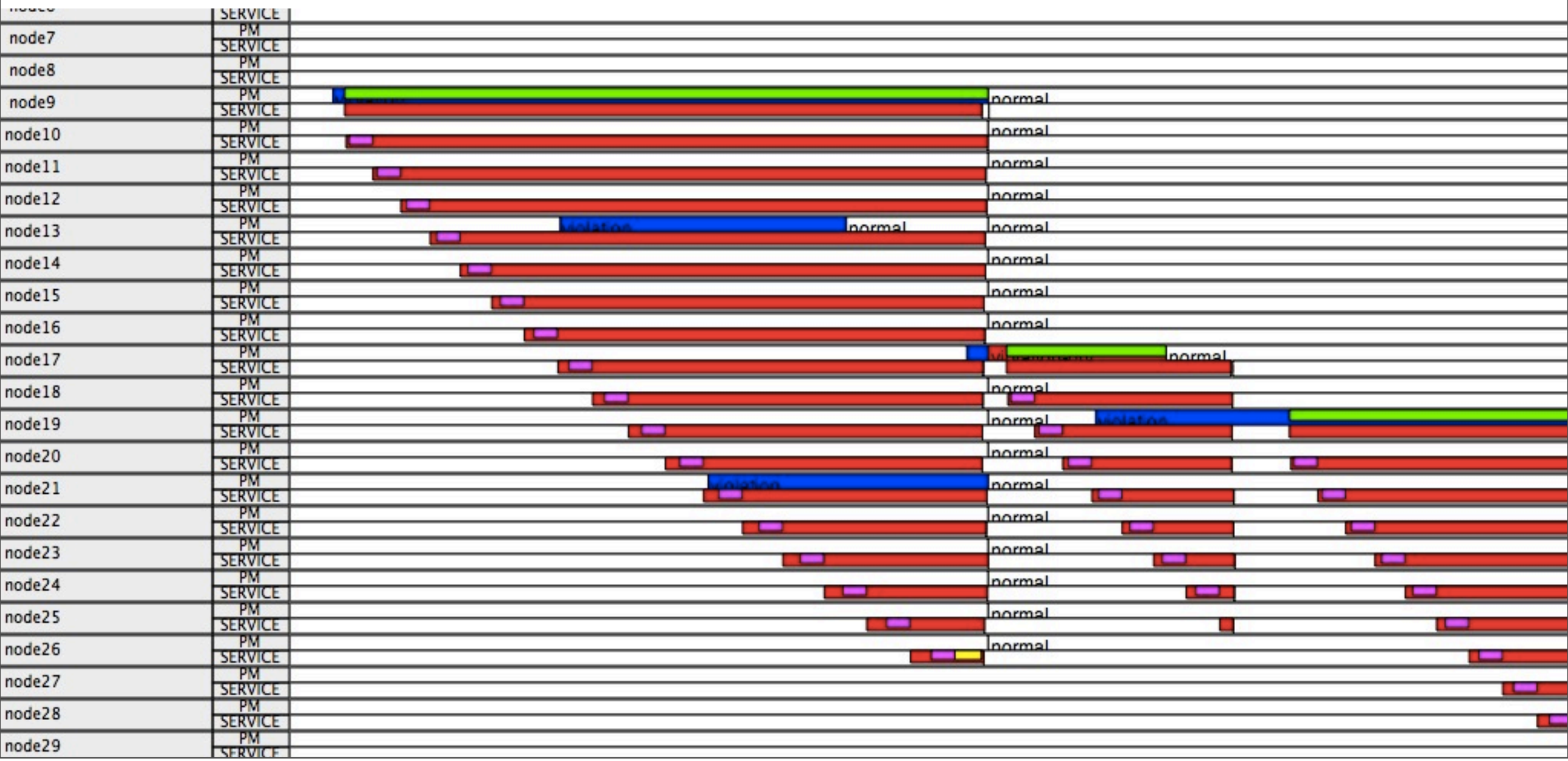
The LUC OS - VEs Scheduling

- Devil is in details
- Paje Traces (collecting during the SimGrid simulation)



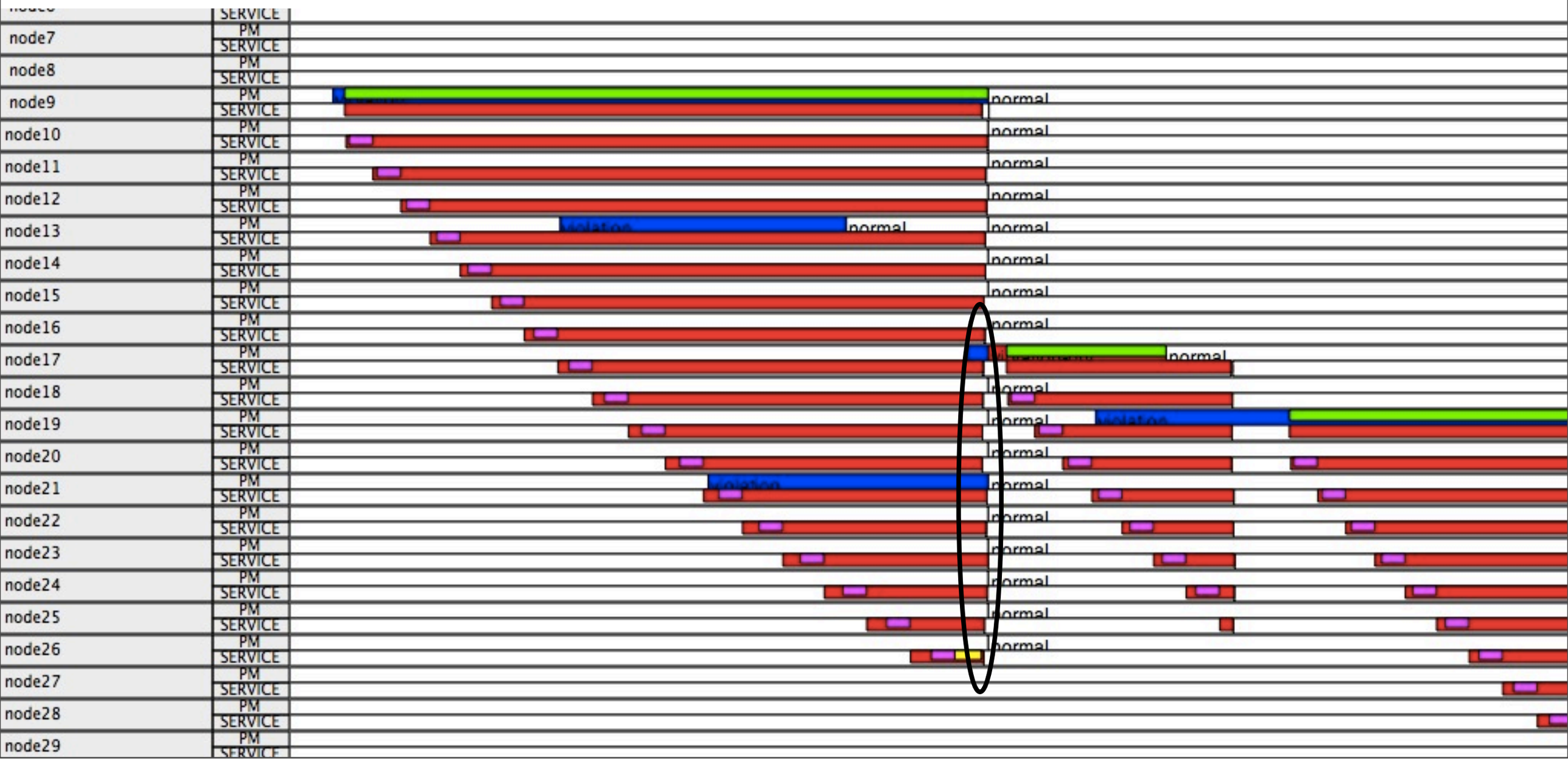
The LUC OS - VEs Scheduling

- Devil is in details
- Paje Traces (collecting during the SimGrid simulation)



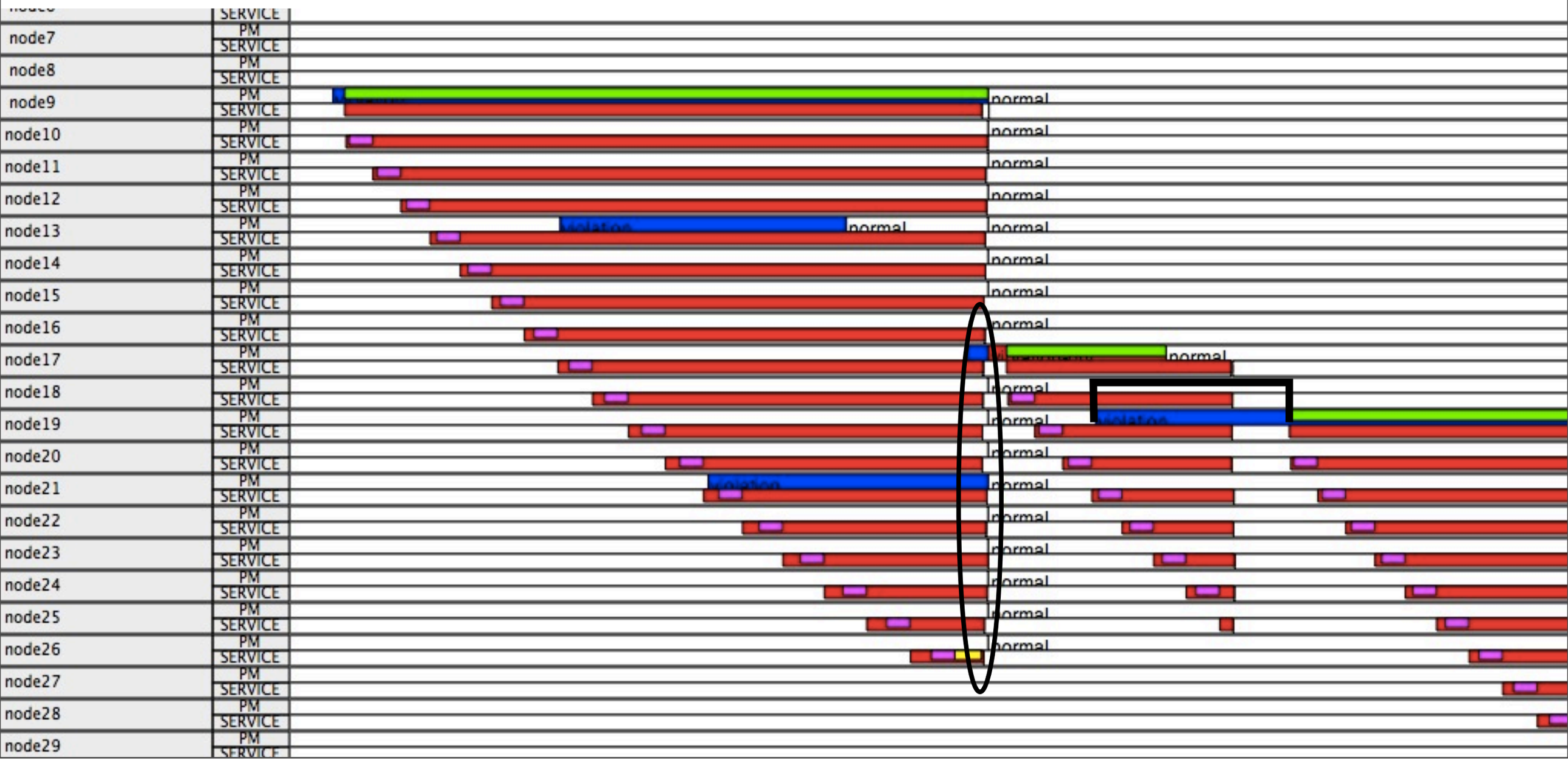
The LUC OS - VEs Scheduling

- Devil is in details
- Paje Traces (collecting during the SimGrid simulation)



The LUC OS - VEs Scheduling

- Devil is in details
- Paje Traces (collecting during the SimGrid simulation)



The LUC OS - DVMS as a first LRT

- DIStributed and COoperative approach to schedule VEs

  Reactivity/scalability
  Scheduling started when an event is generated

  Few nodes considered for scheduling
  ⇒ much faster computation

  Parallelism
  Several events can be processed simultaneously/independently

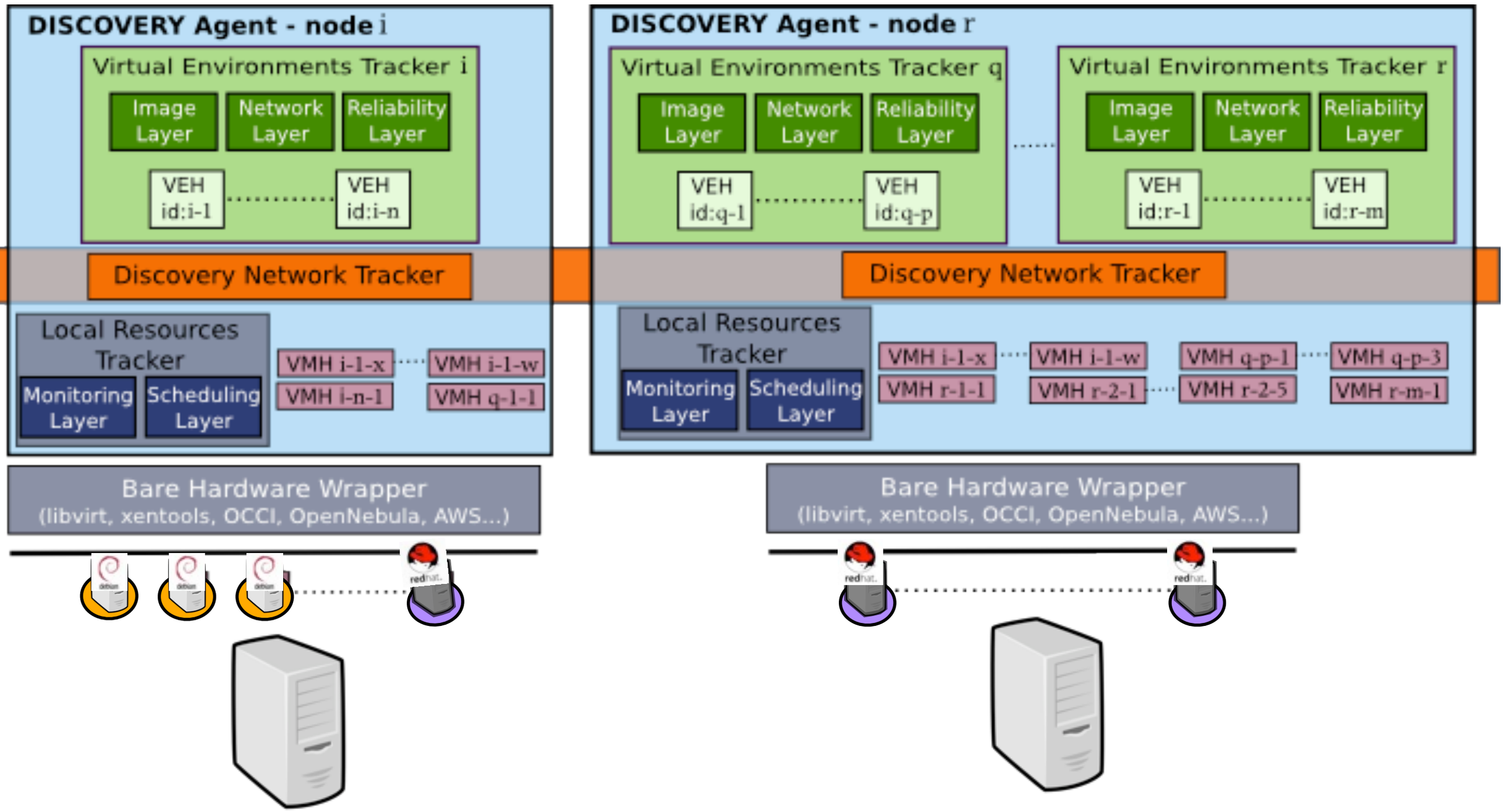
  Published in [CCPE'12, ISPA'13]

  A POC in JAVA (however no more maintained)
  A new implementation based on SCALA/Akka

  http://beyondtheclouds.github.io/

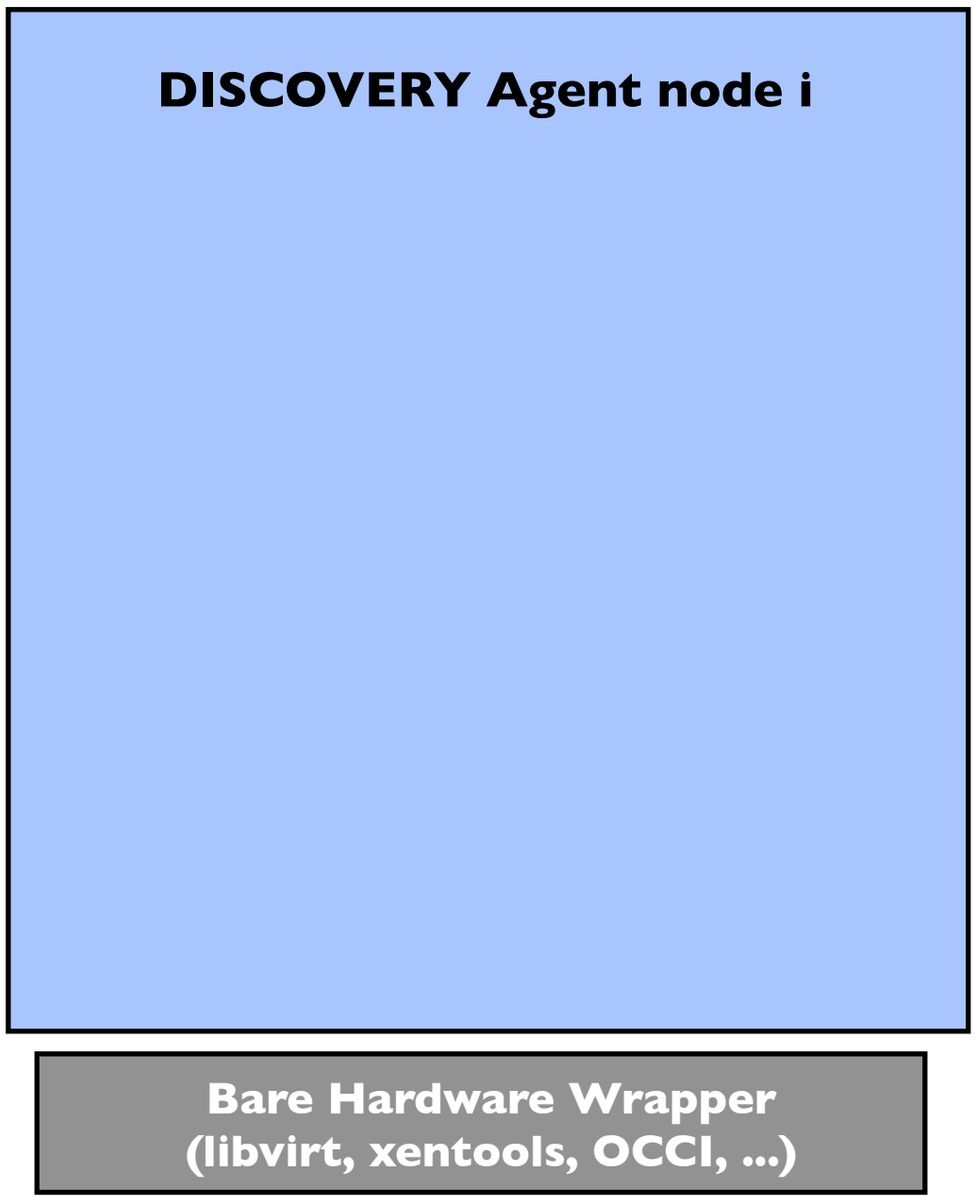
Discovery Internals in a Nutshell

Understanding the DISCOVERY Agent

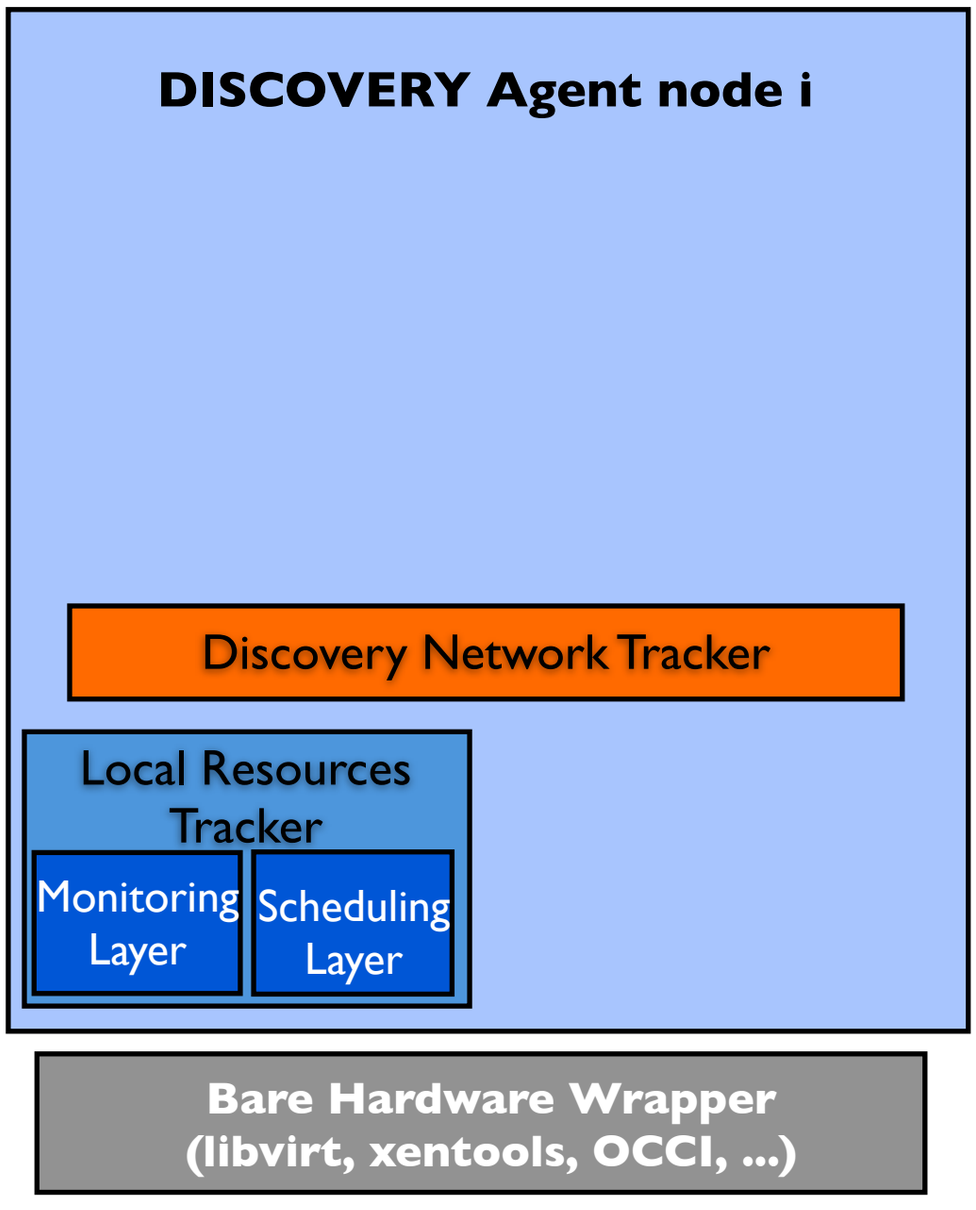


DISCOVERY - Basic Usage

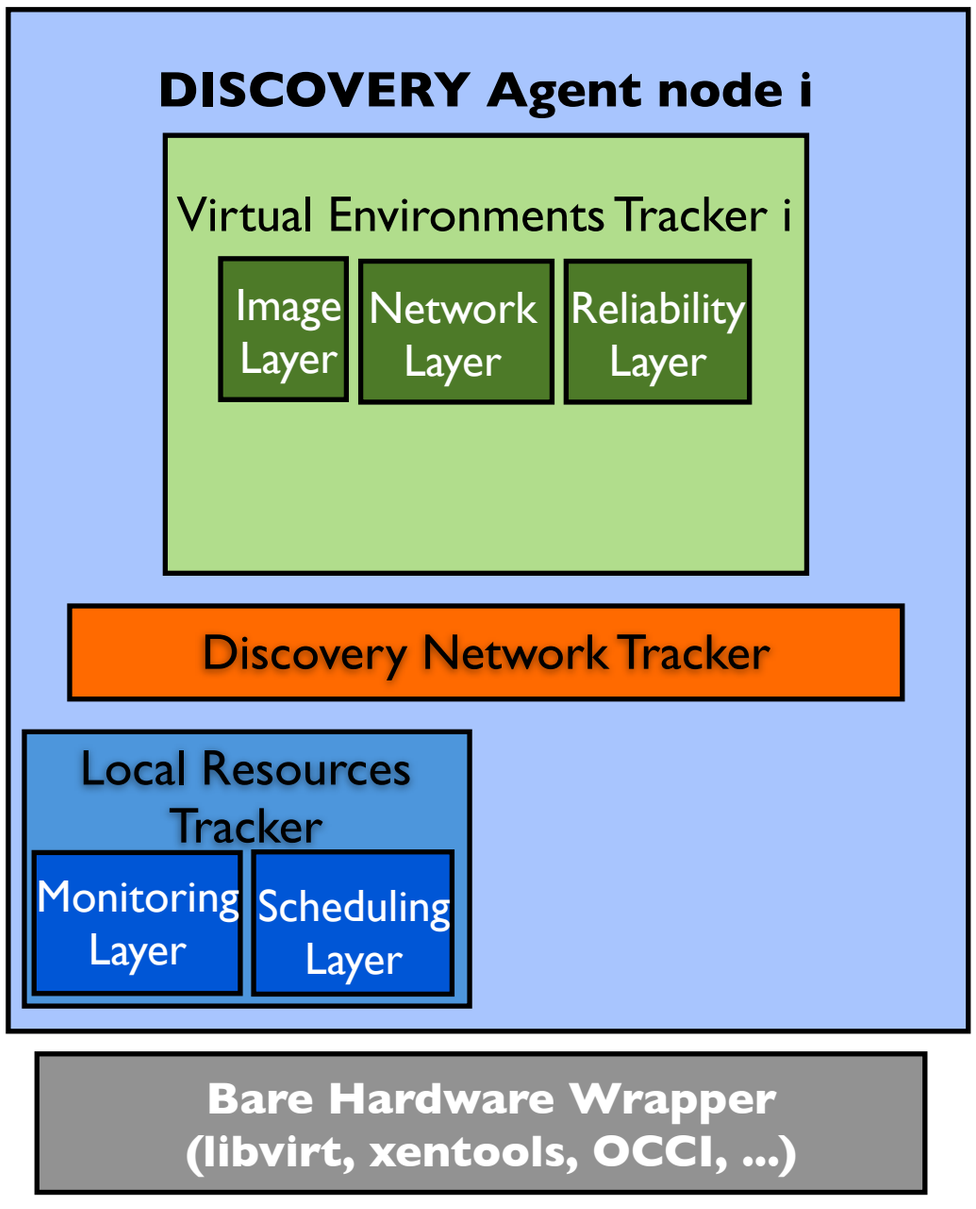
DISCOVERY - Basic Usage



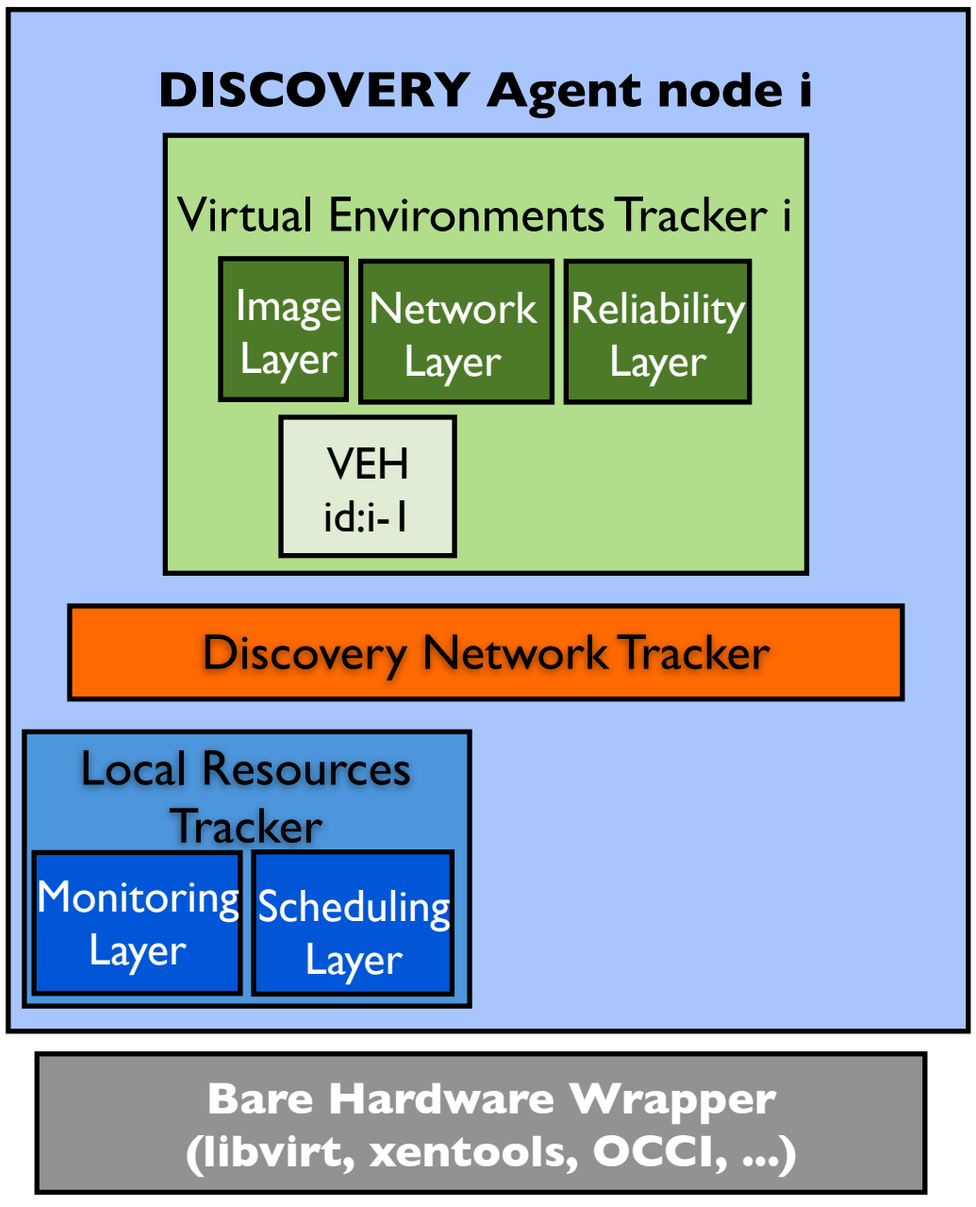
DISCOVERY - Basic Usage



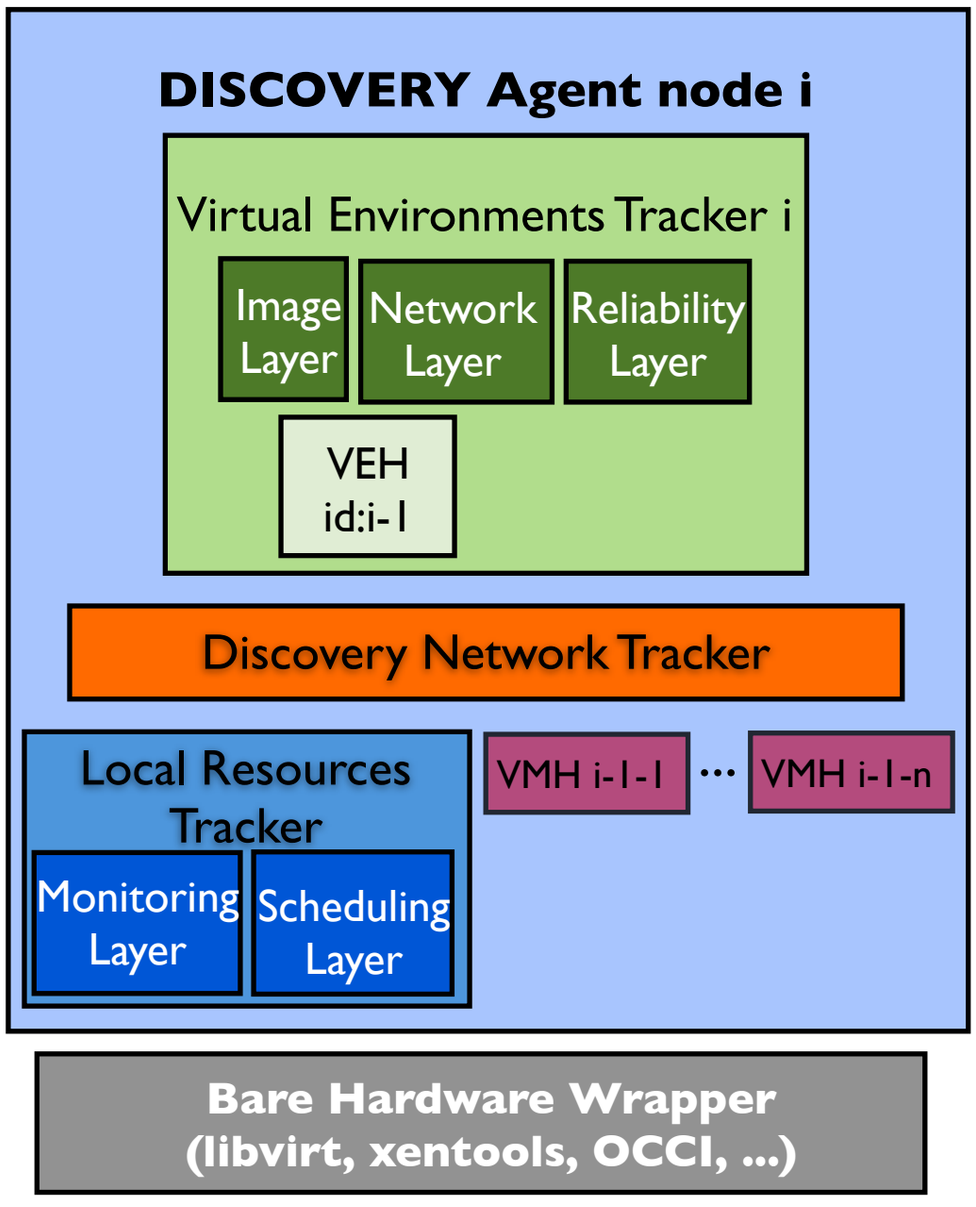
DISCOVERY - Basic Usage



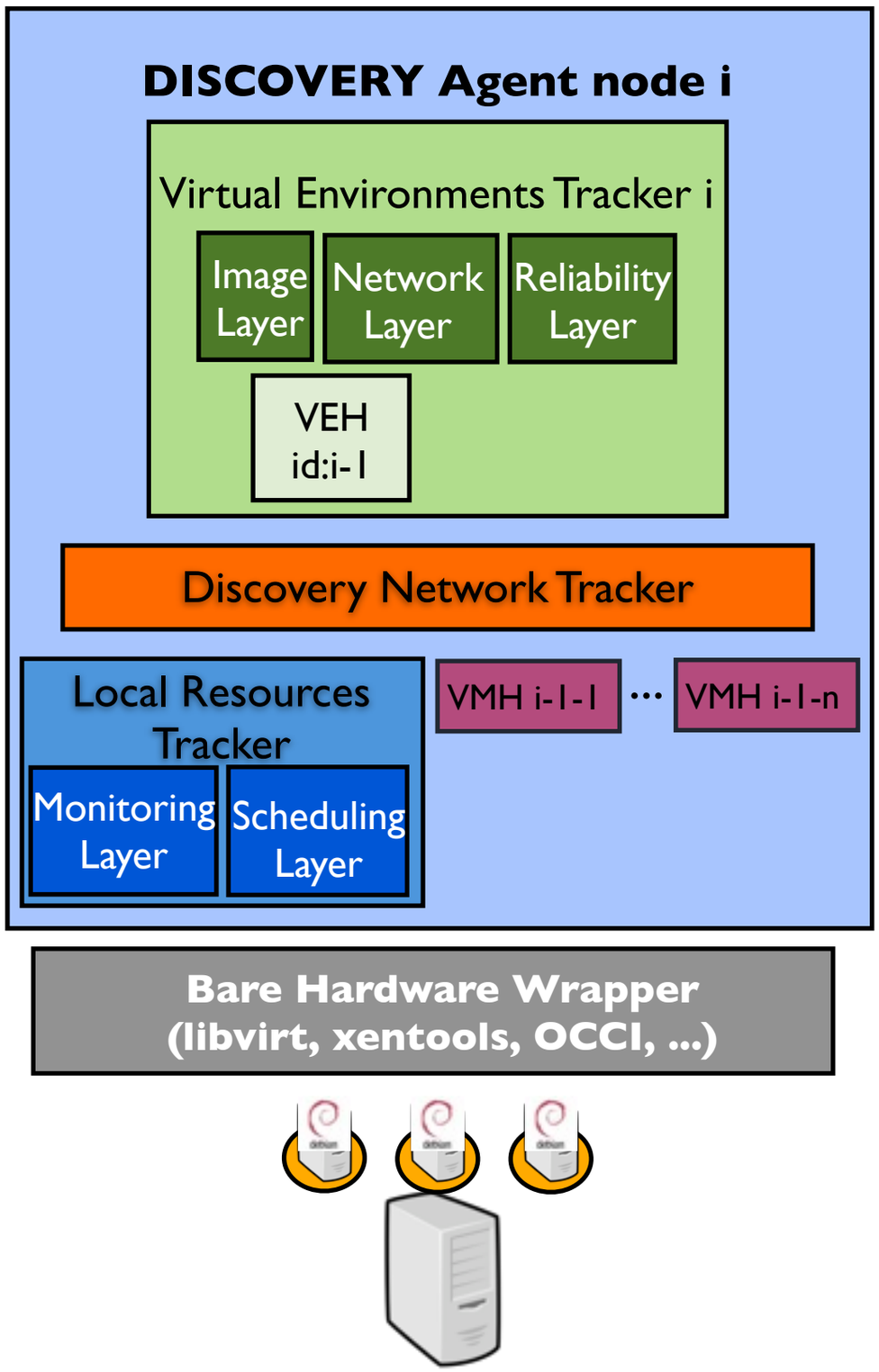
DISCOVERY - Basic Usage



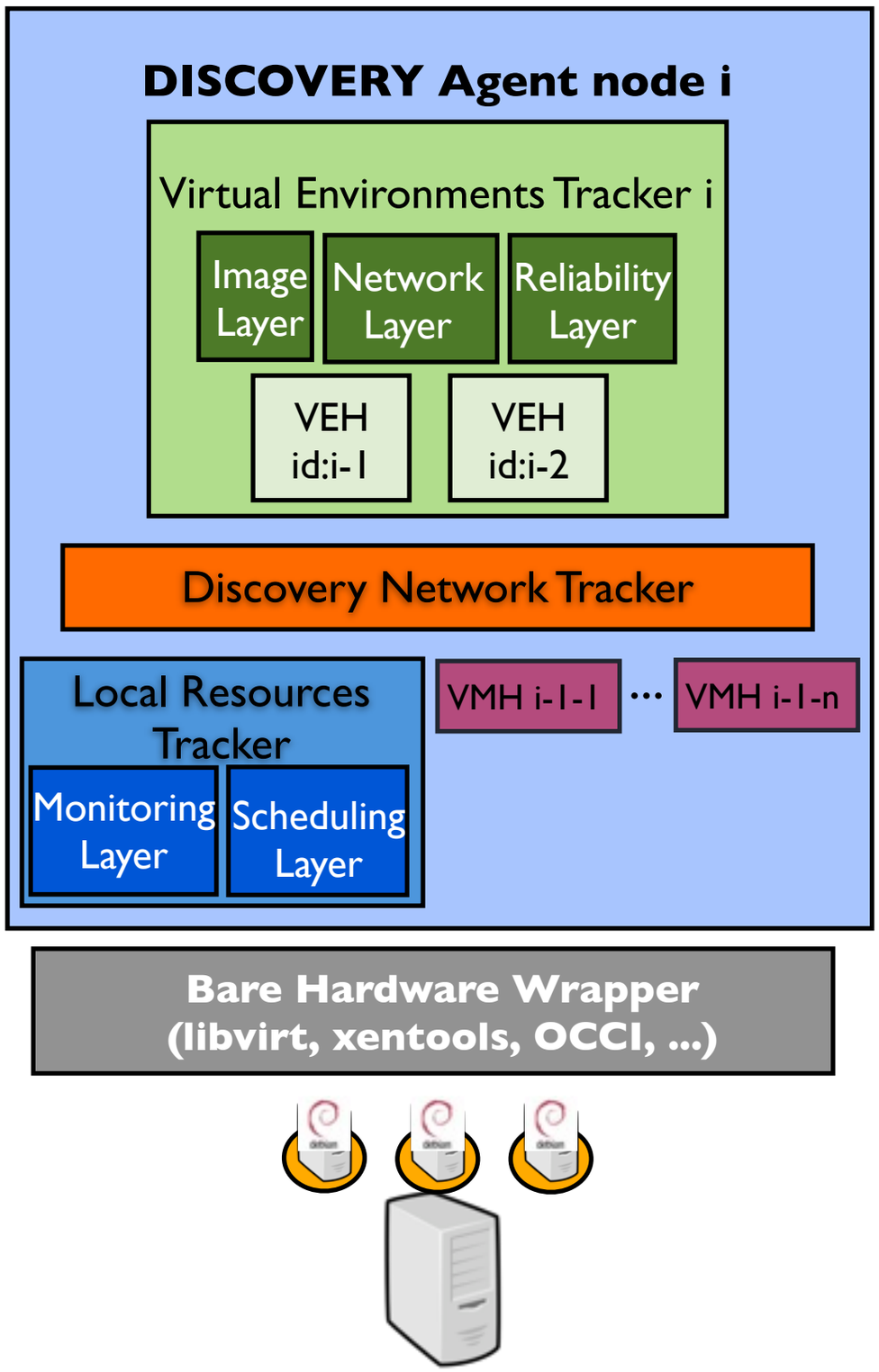
DISCOVERY - Basic Usage



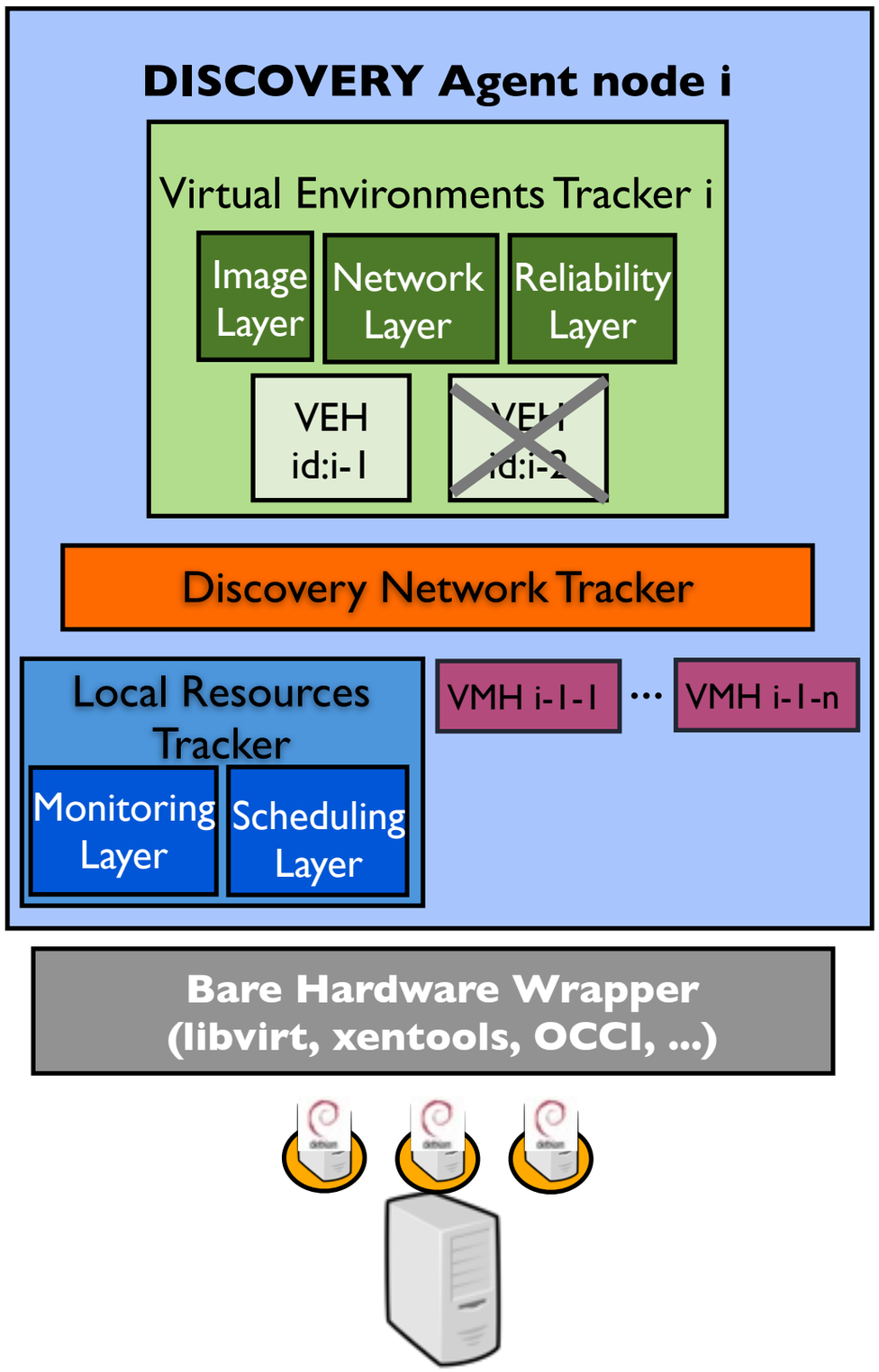
DISCOVERY - Basic Usage



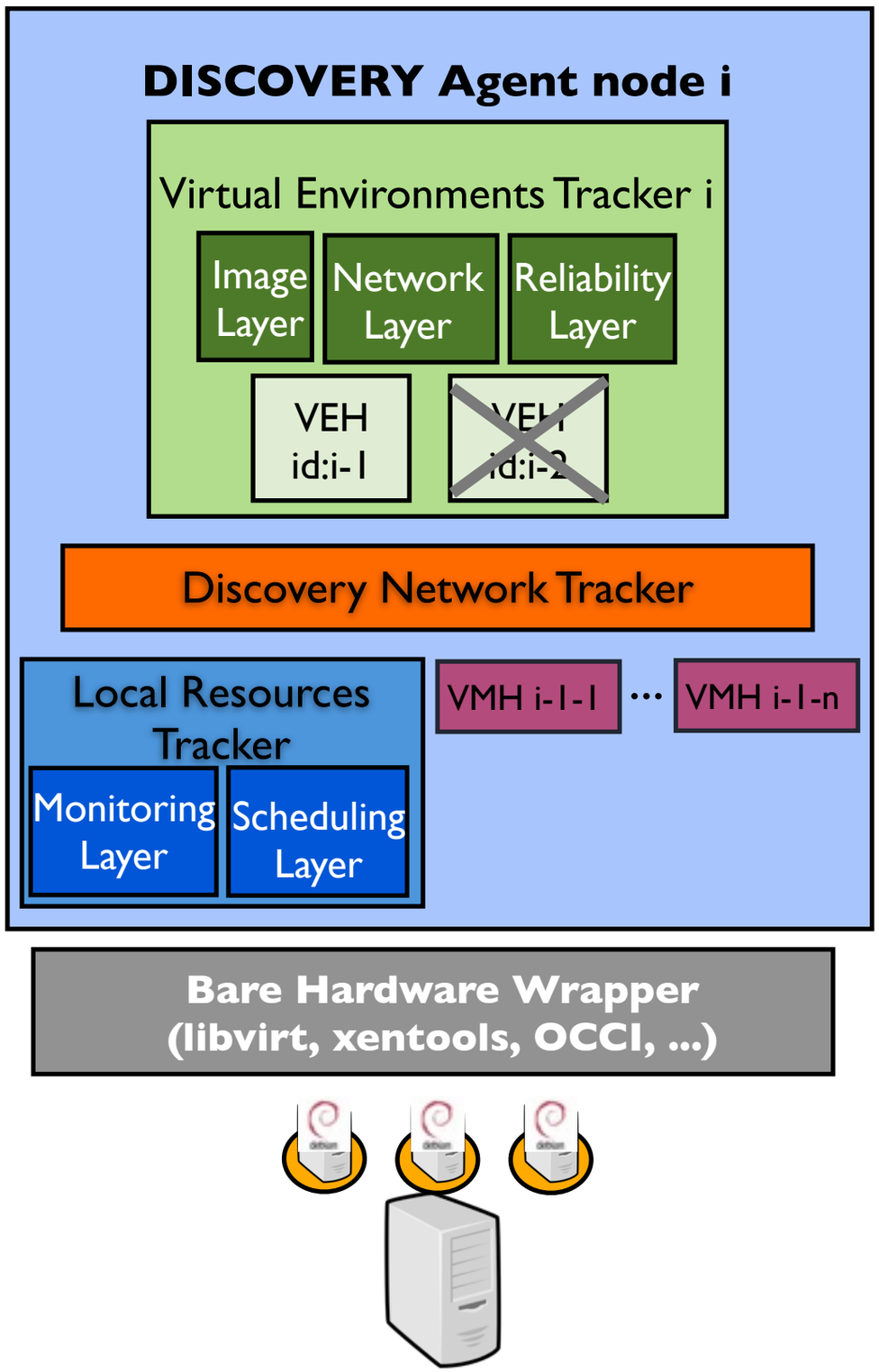
DISCOVERY - Basic Usage



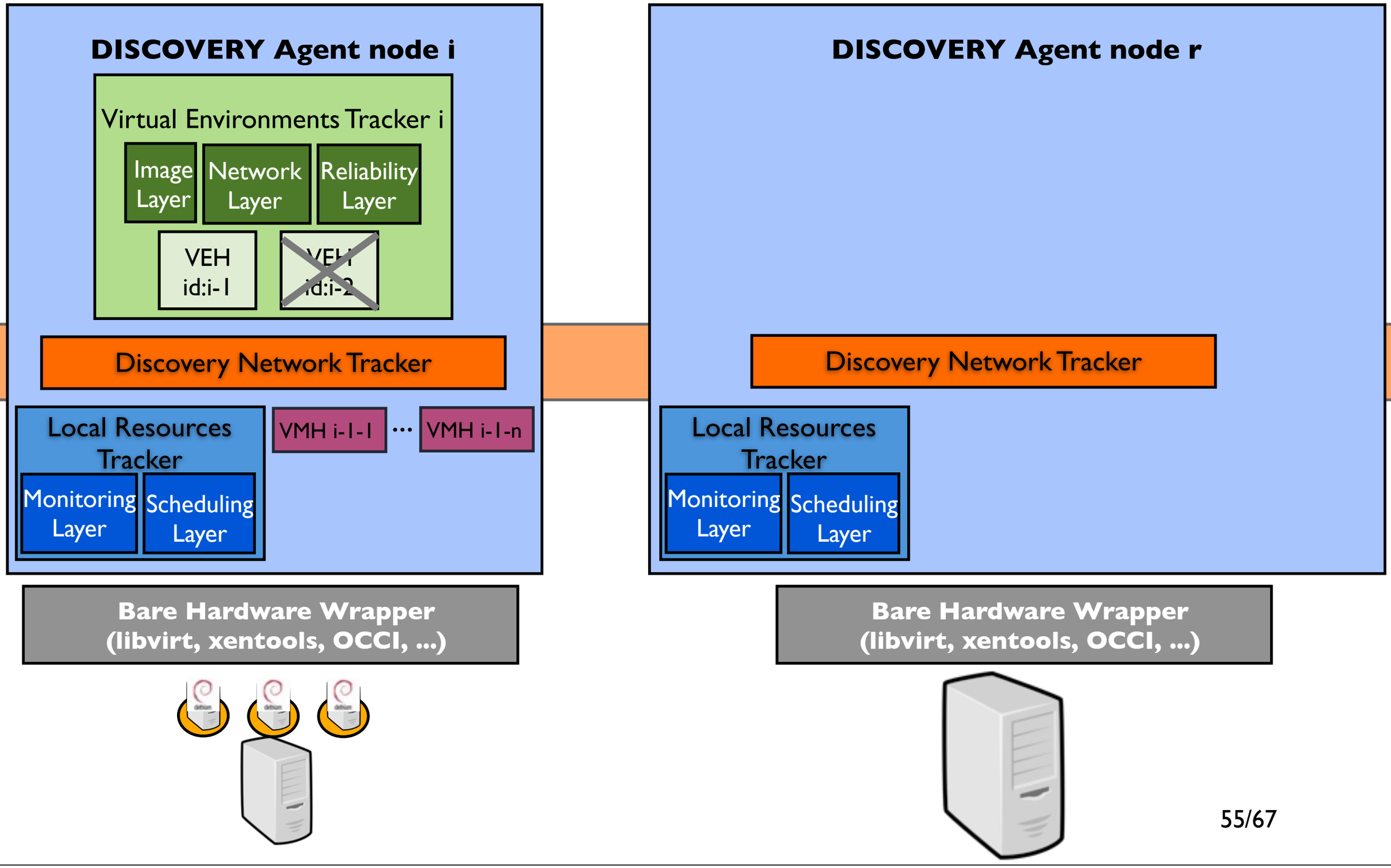
DISCOVERY - Basic Usage



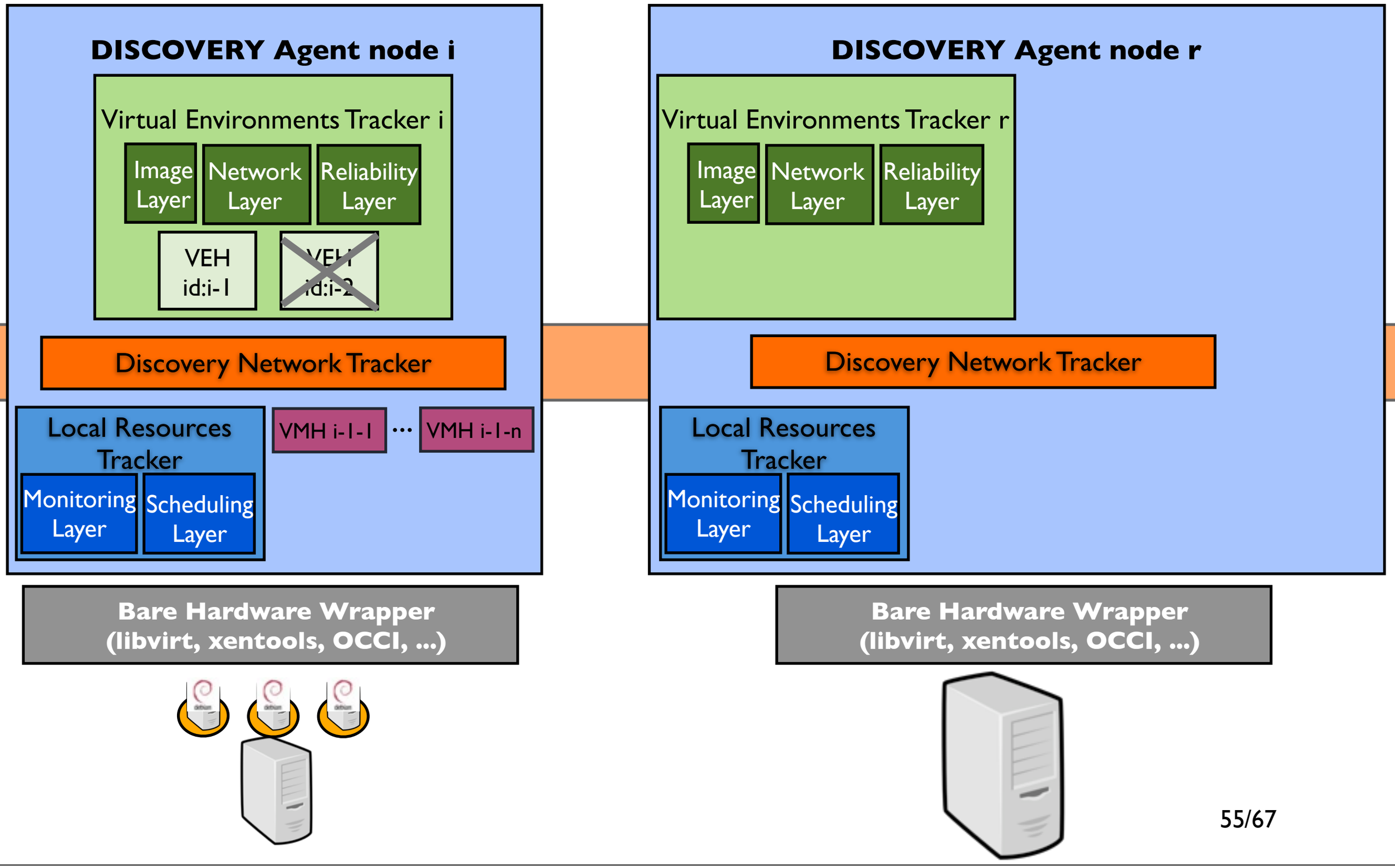
DISCOVERY - Basic Usage



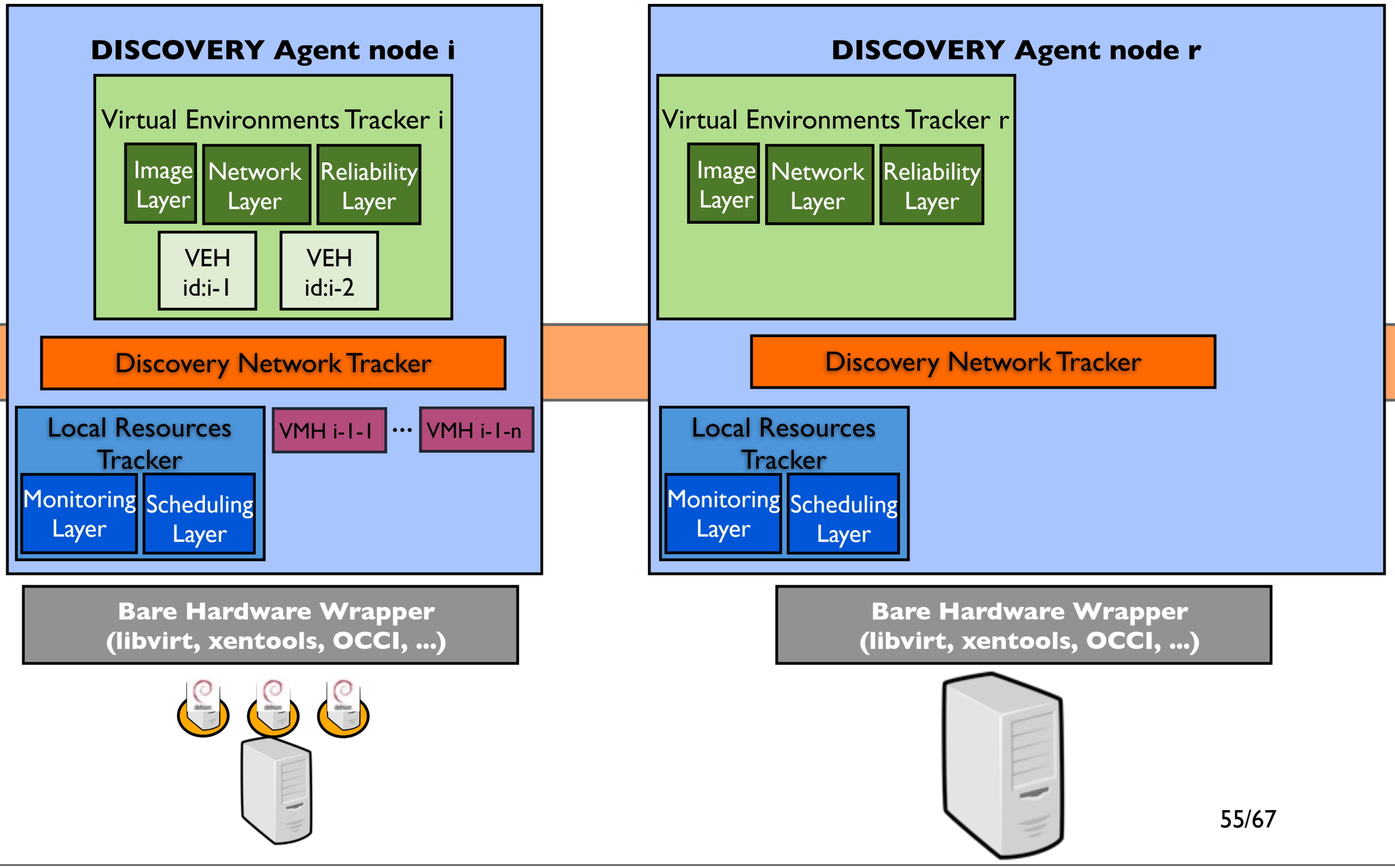
DISCOVERY - Basic Usage



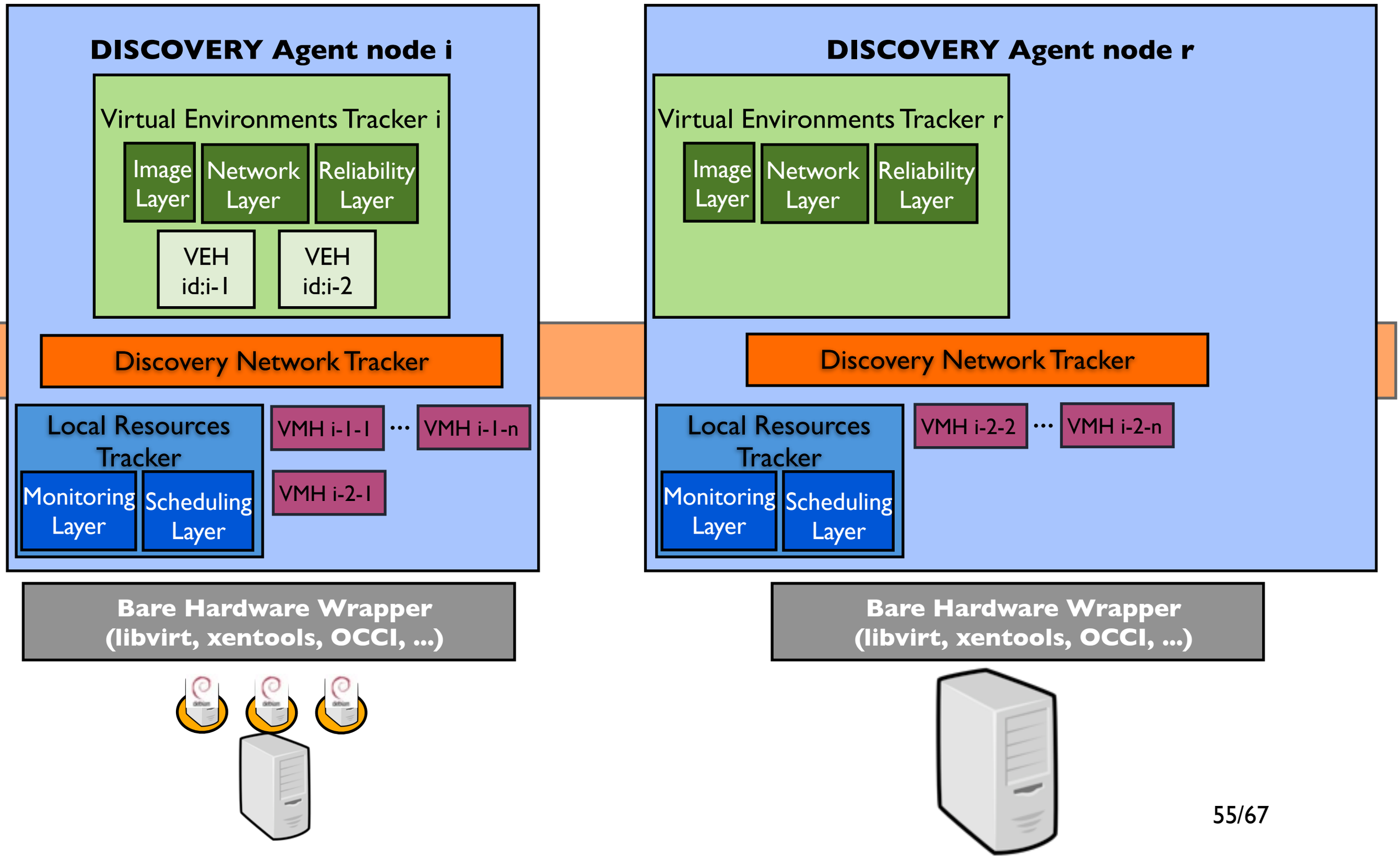
DISCOVERY - Basic Usage



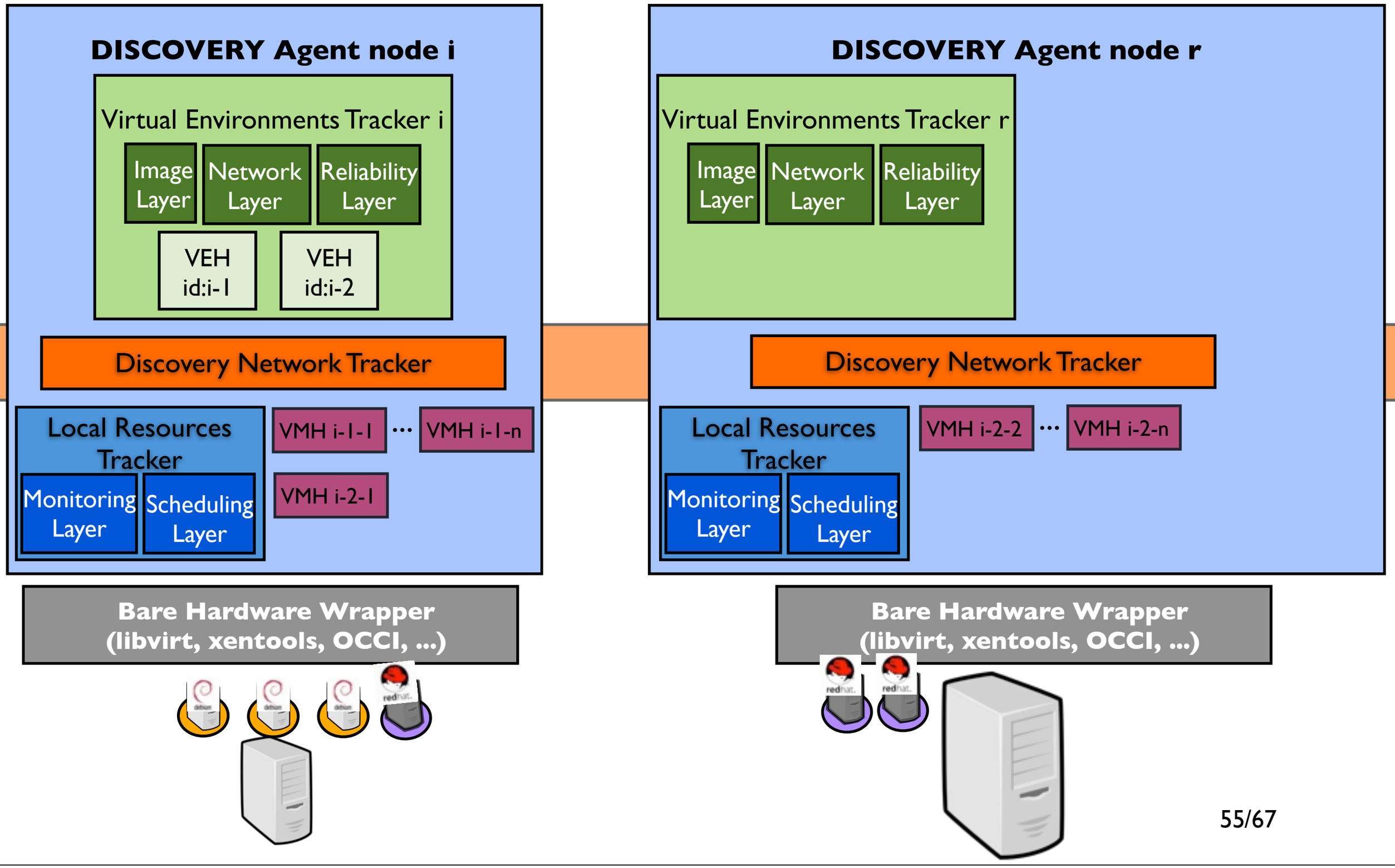
DISCOVERY - Basic Usage



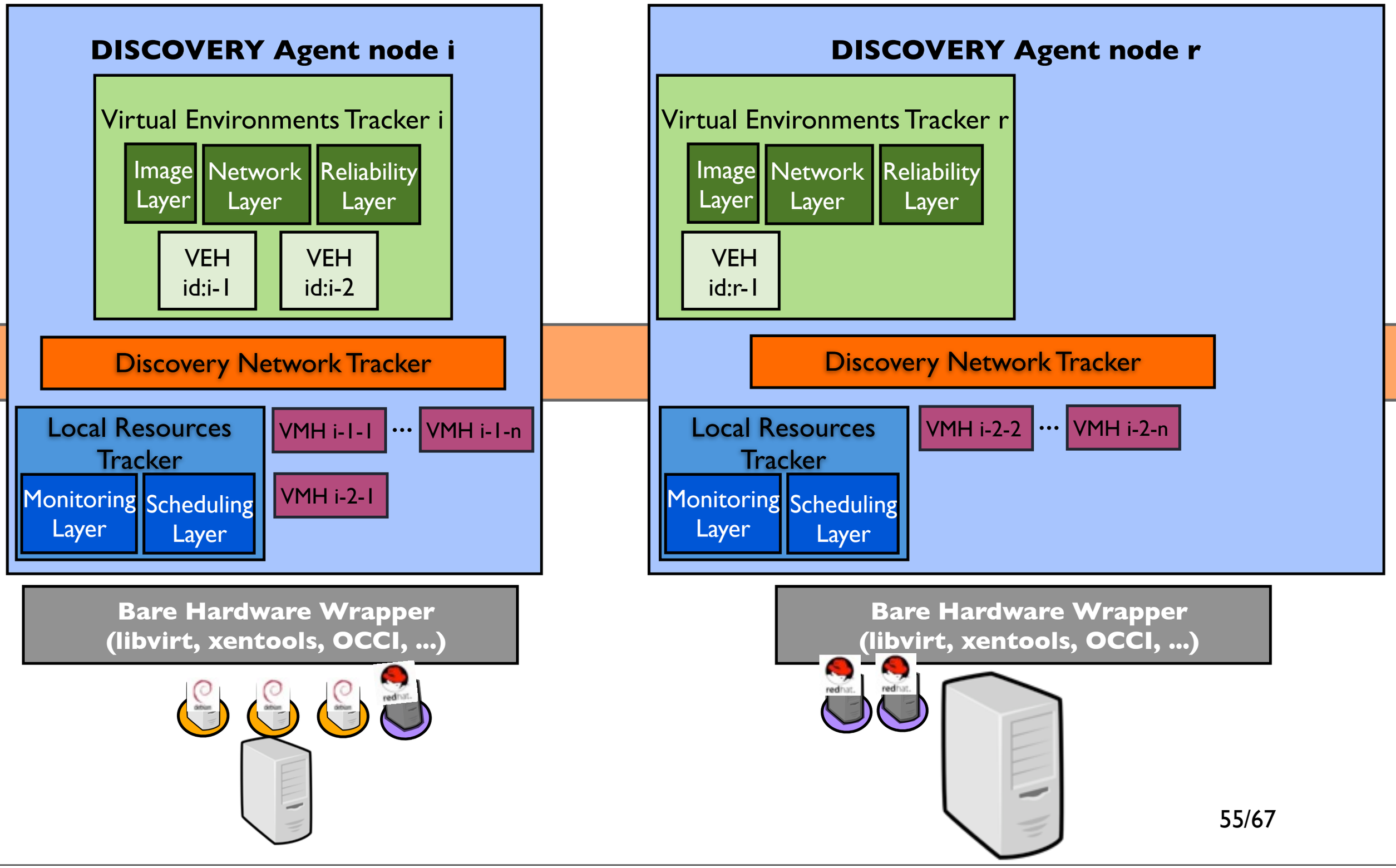
DISCOVERY - Basic Usage



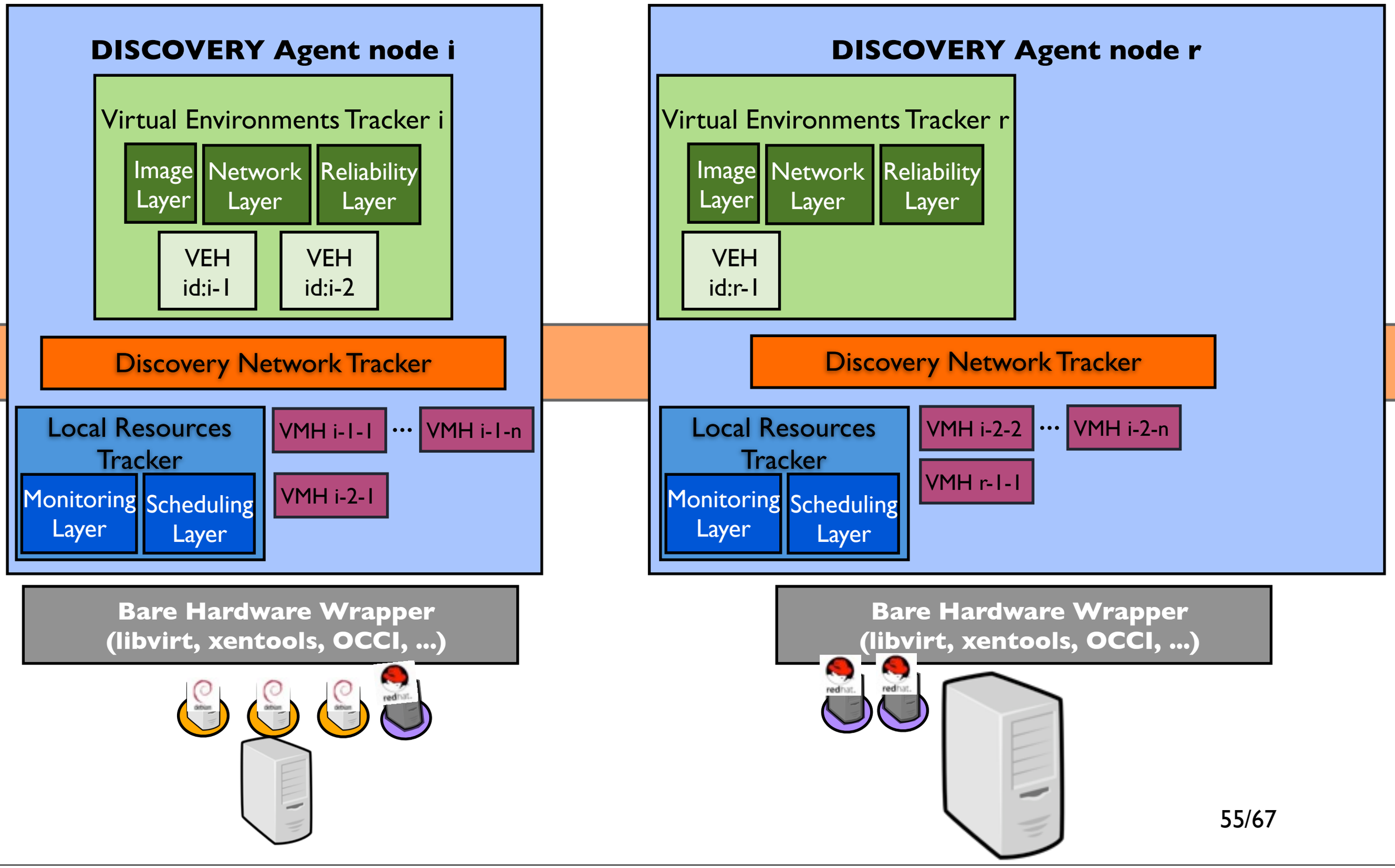
DISCOVERY - Basic Usage



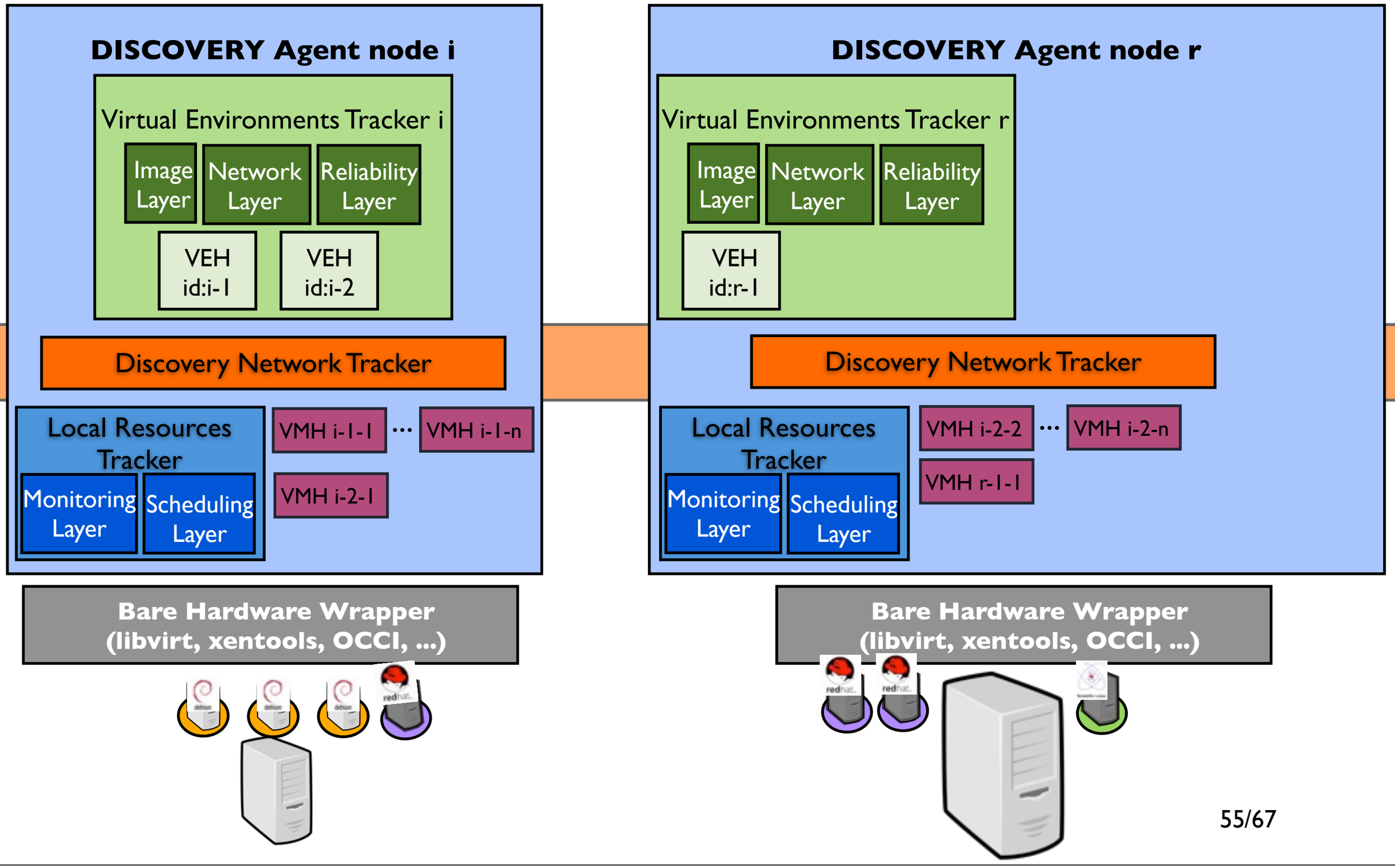
DISCOVERY - Basic Usage



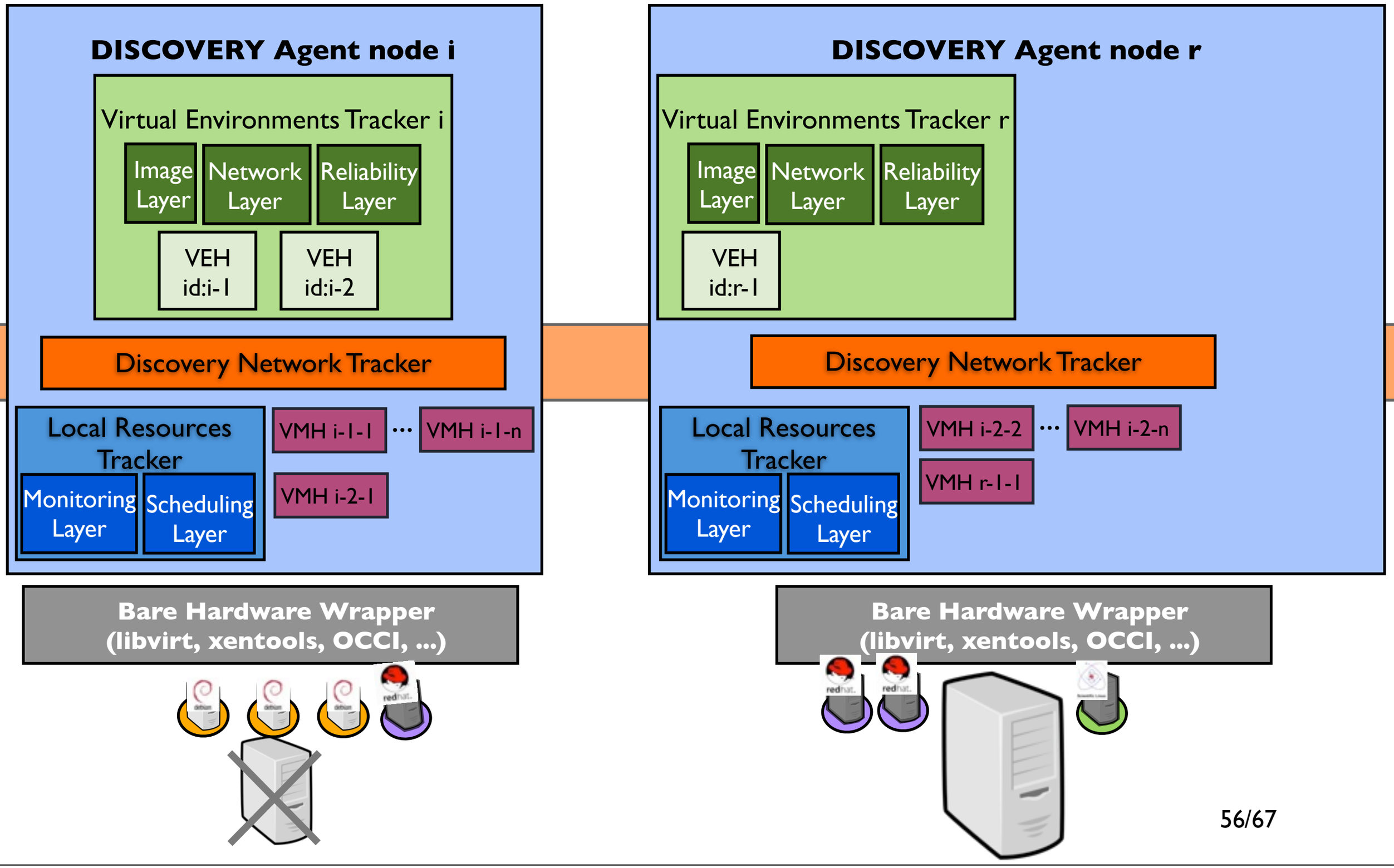
DISCOVERY - Basic Usage



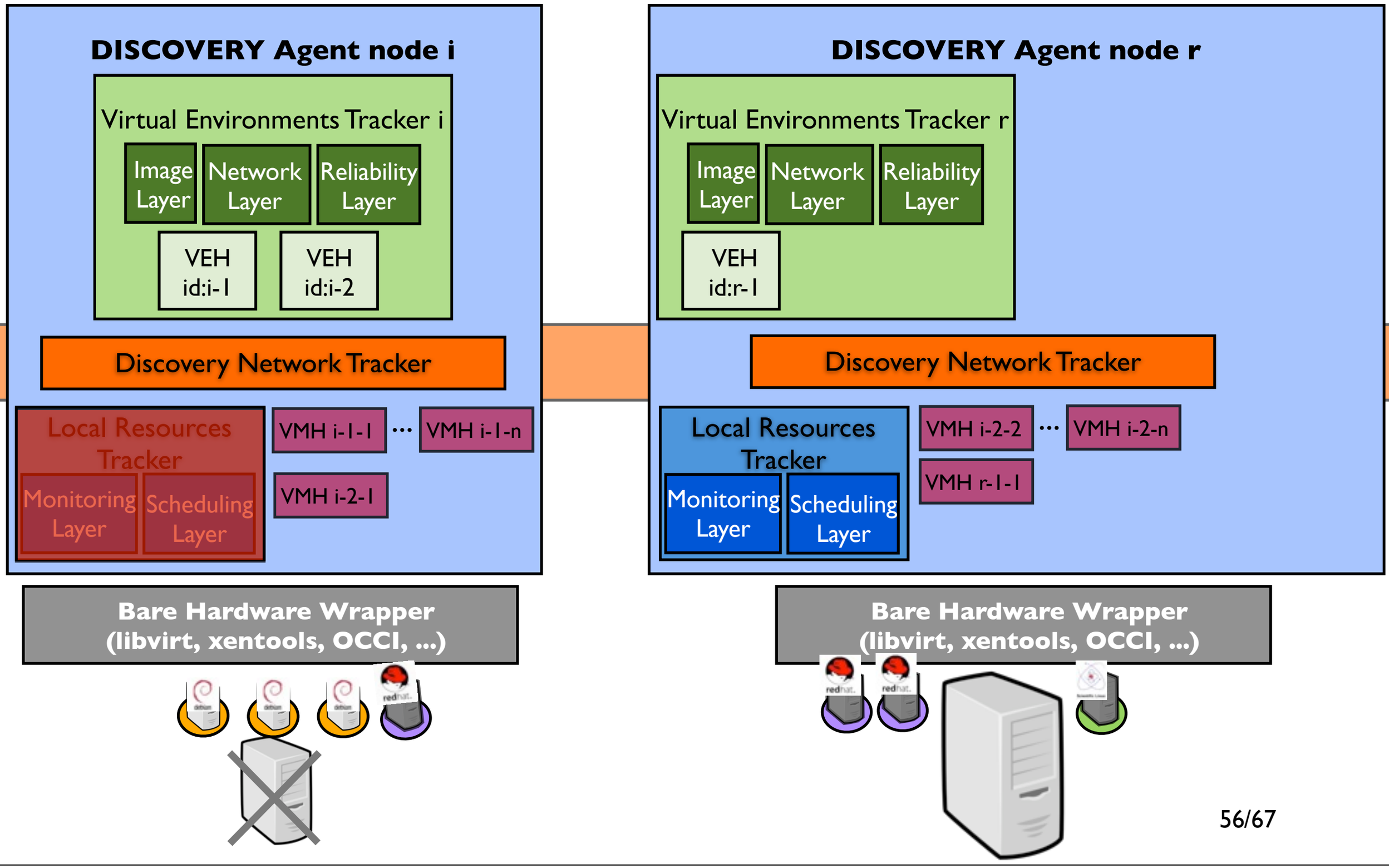
DISCOVERY - Basic Usage



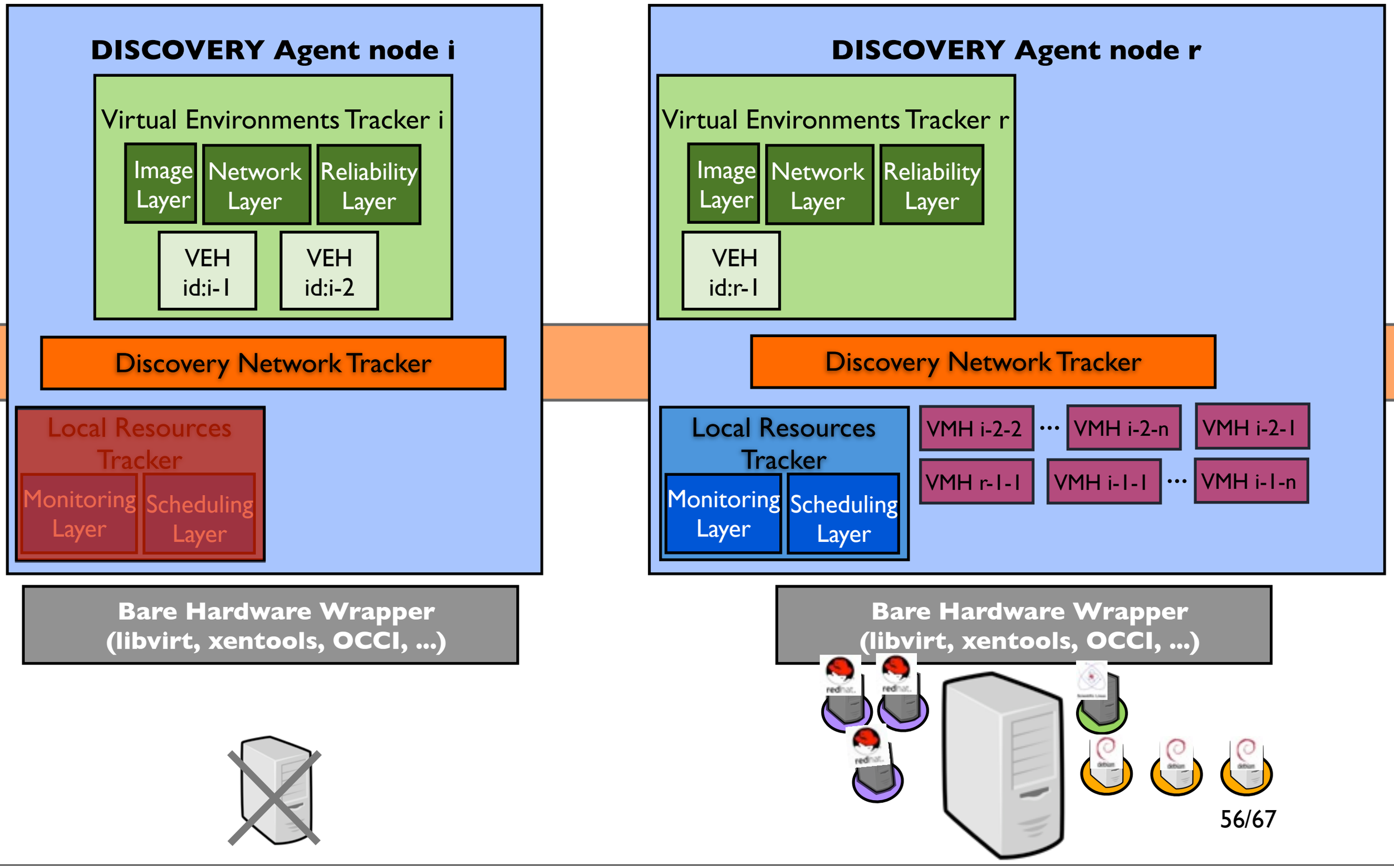
DISCOVERY - Human removals



DISCOVERY - Human removals



DISCOVERY - Human removals



DISCOVERY - Human removals

DISCOVERY Agent node i

Discovery Network Tracker

Local Resources Tracker

Monitoring Layer

Scheduling Layer

DISCOVERY Agent node r

Virtual Environments Tracker r

Image Layer

Network Layer

Reliability Layer

VEH id:r-1

Virtual Environments Tracker i

Image Layer

Network Layer

Reliability Layer

VEH id:i-1

VEH id:i-2

Discovery Network Tracker

Local Resources Tracker

Monitoring Layer

Scheduling Layer

VMH i-2-2 ··· VMH i-2-n

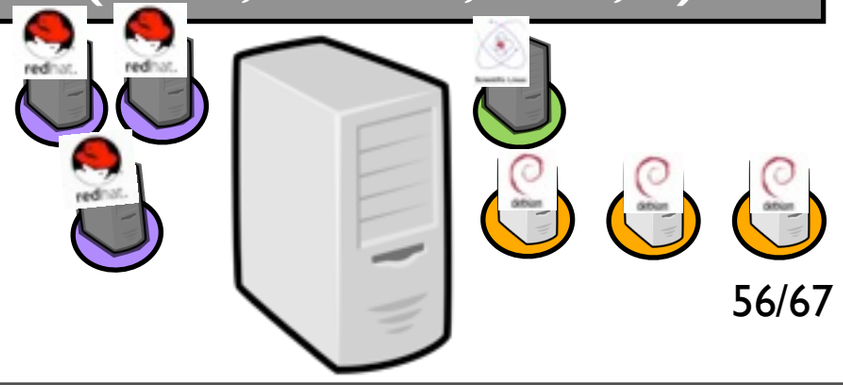
VMH i-2-1

VMH r-1-1

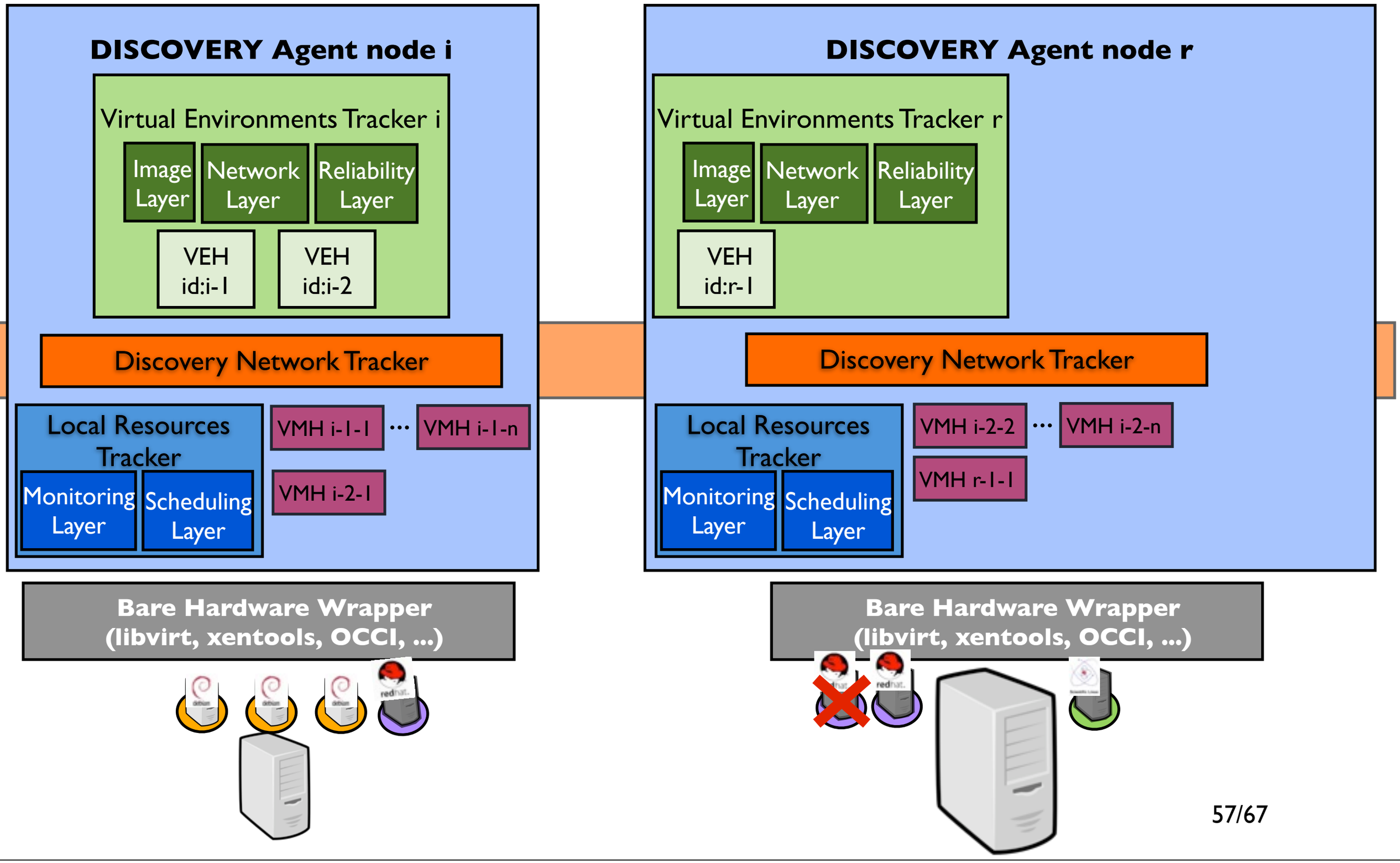
VMH i-1-1 ··· VMH i-1-n

Bare Hardware Wrapper (libvirt, xentools, OCCI, ...)

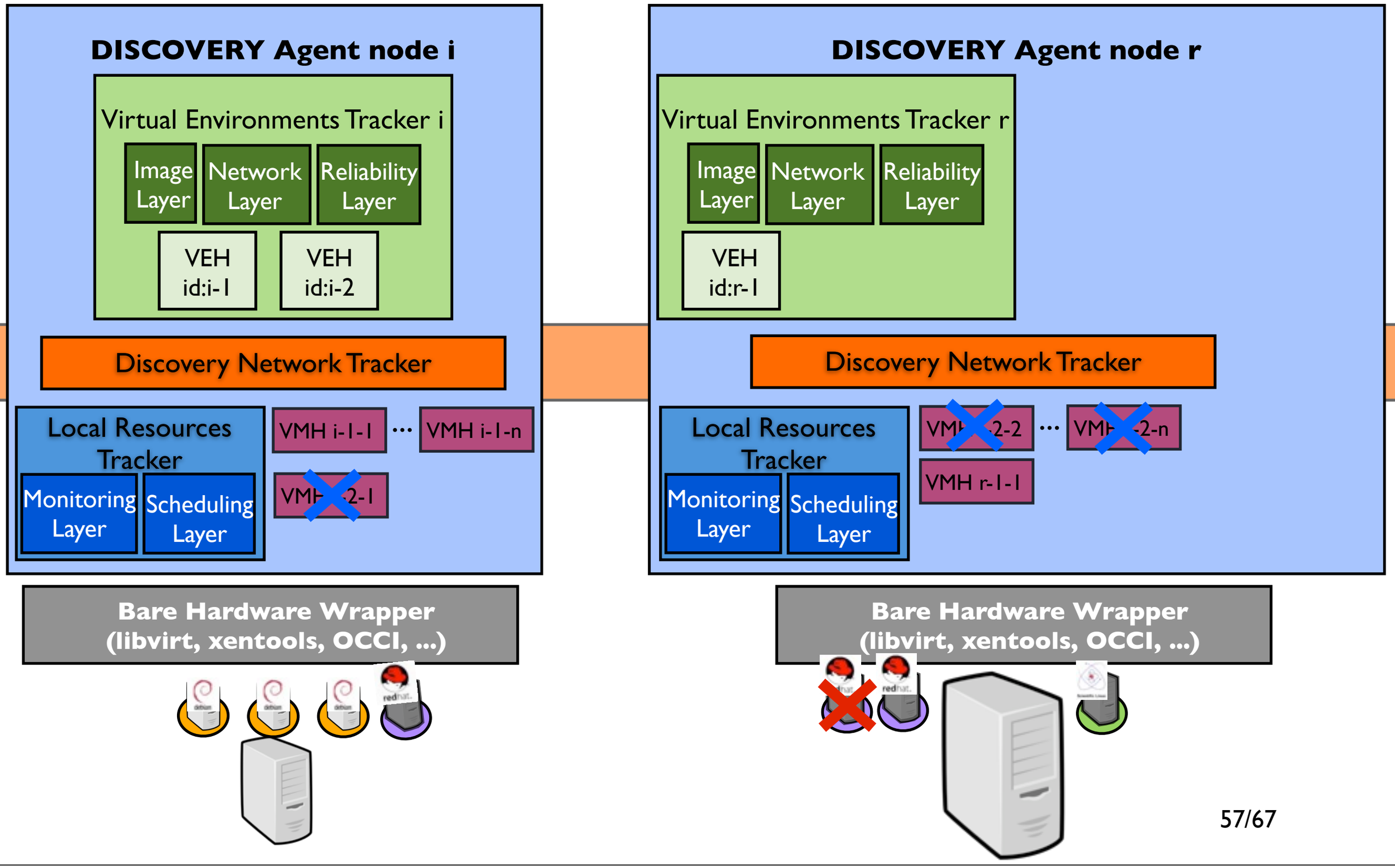
Bare Hardware Wrapper (libvirt, xentools, OCCI, ...)



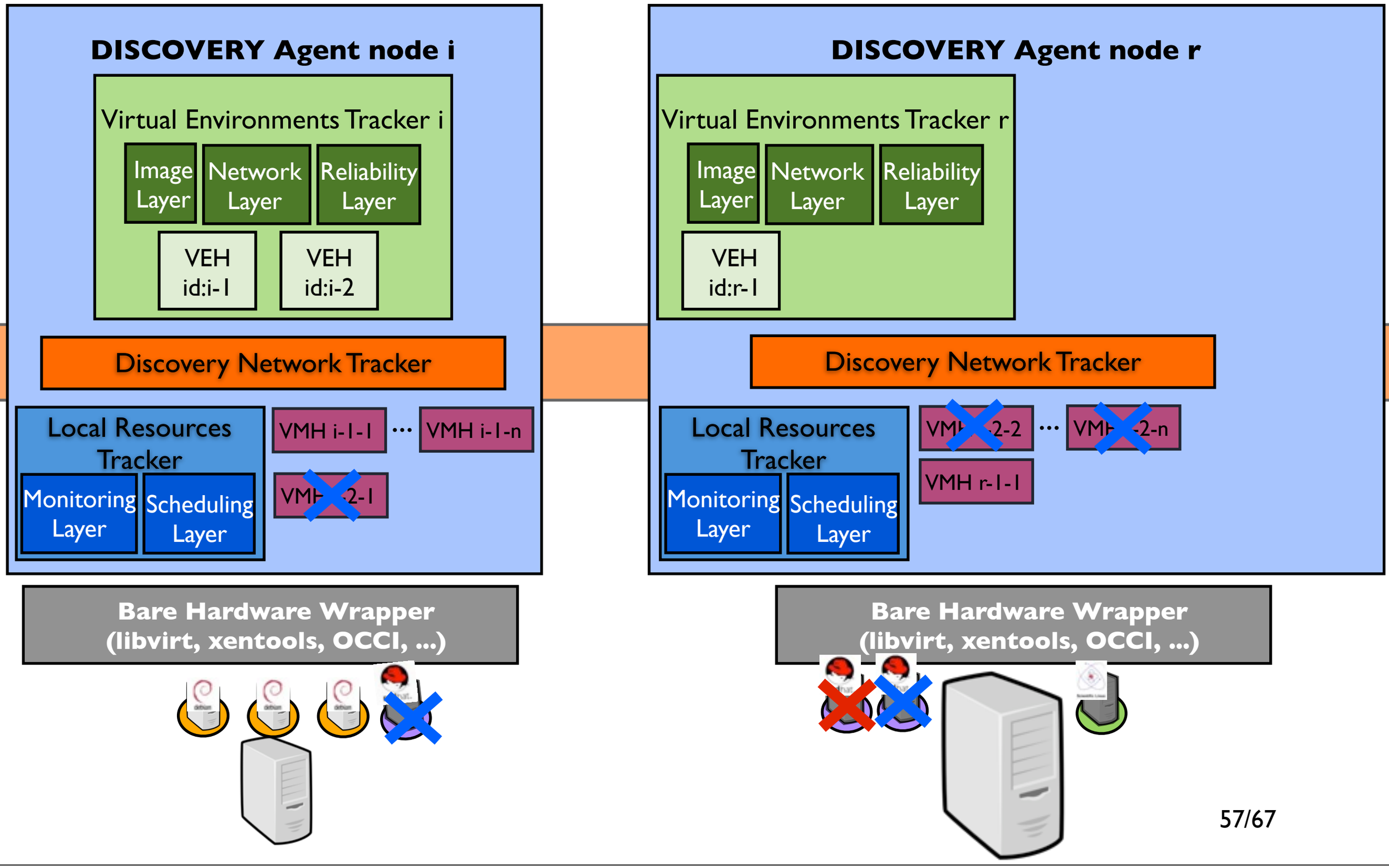
DISCOVERY - VM Crashes



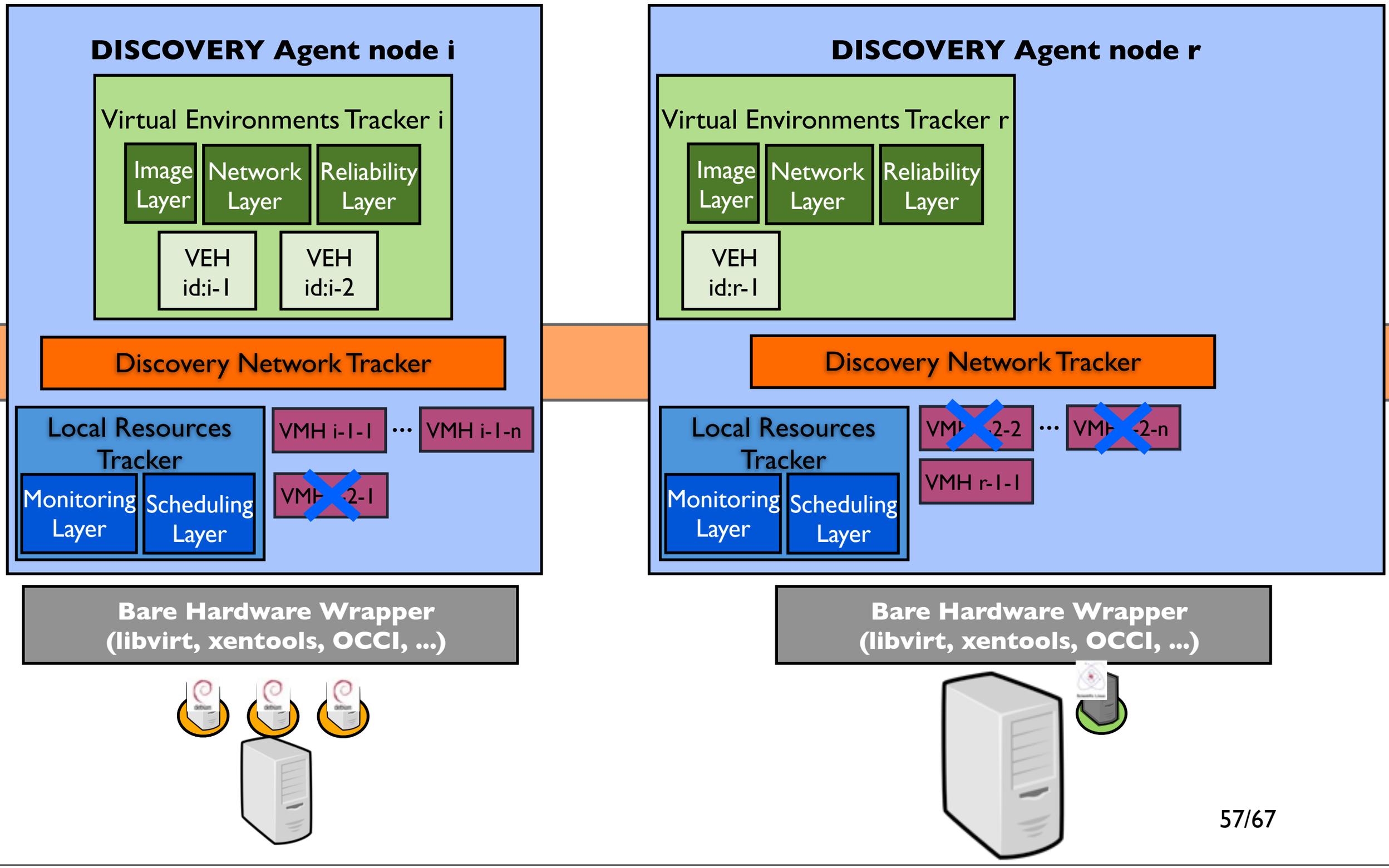
DISCOVERY - VM Crashes



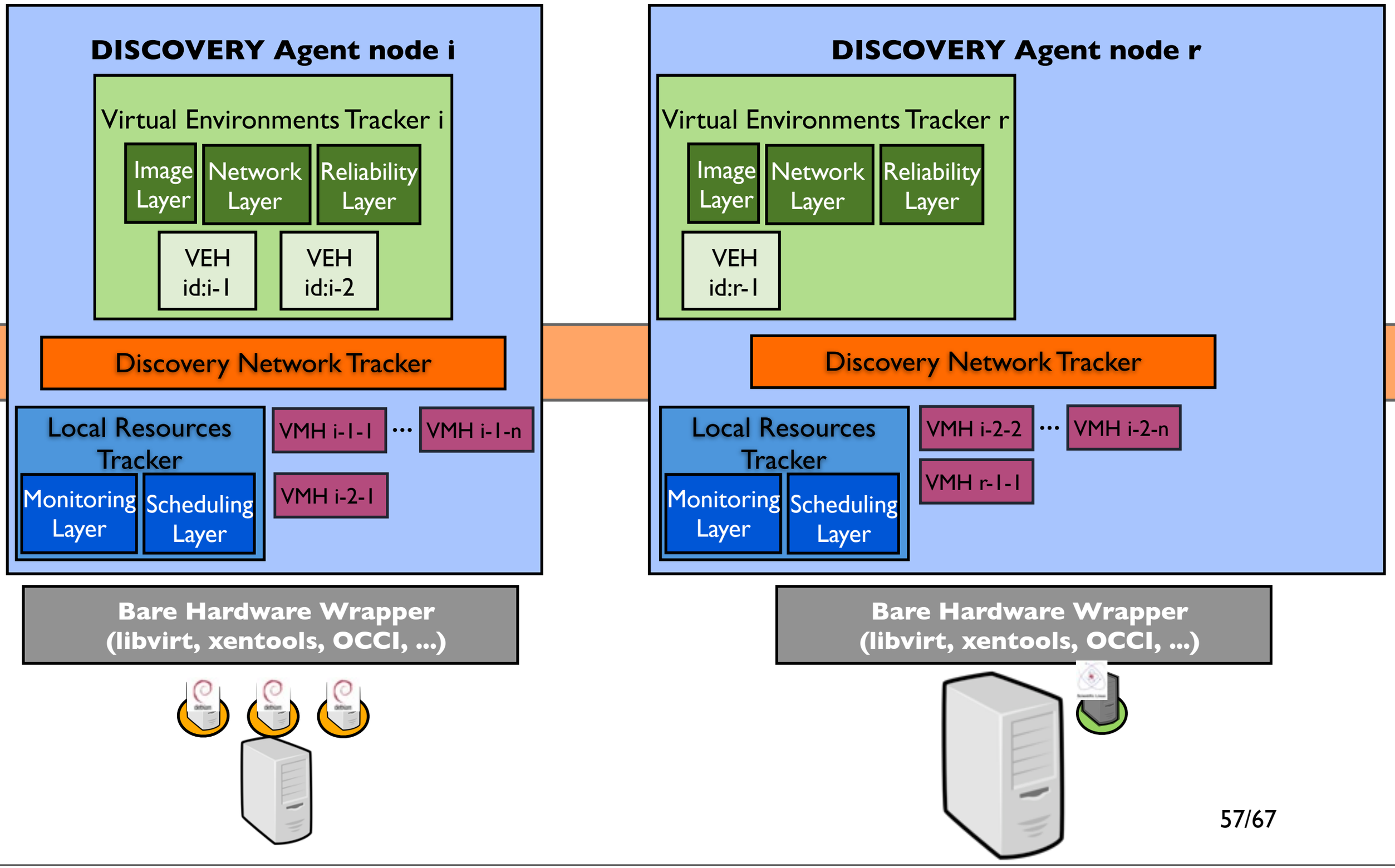
DISCOVERY - VM Crashes



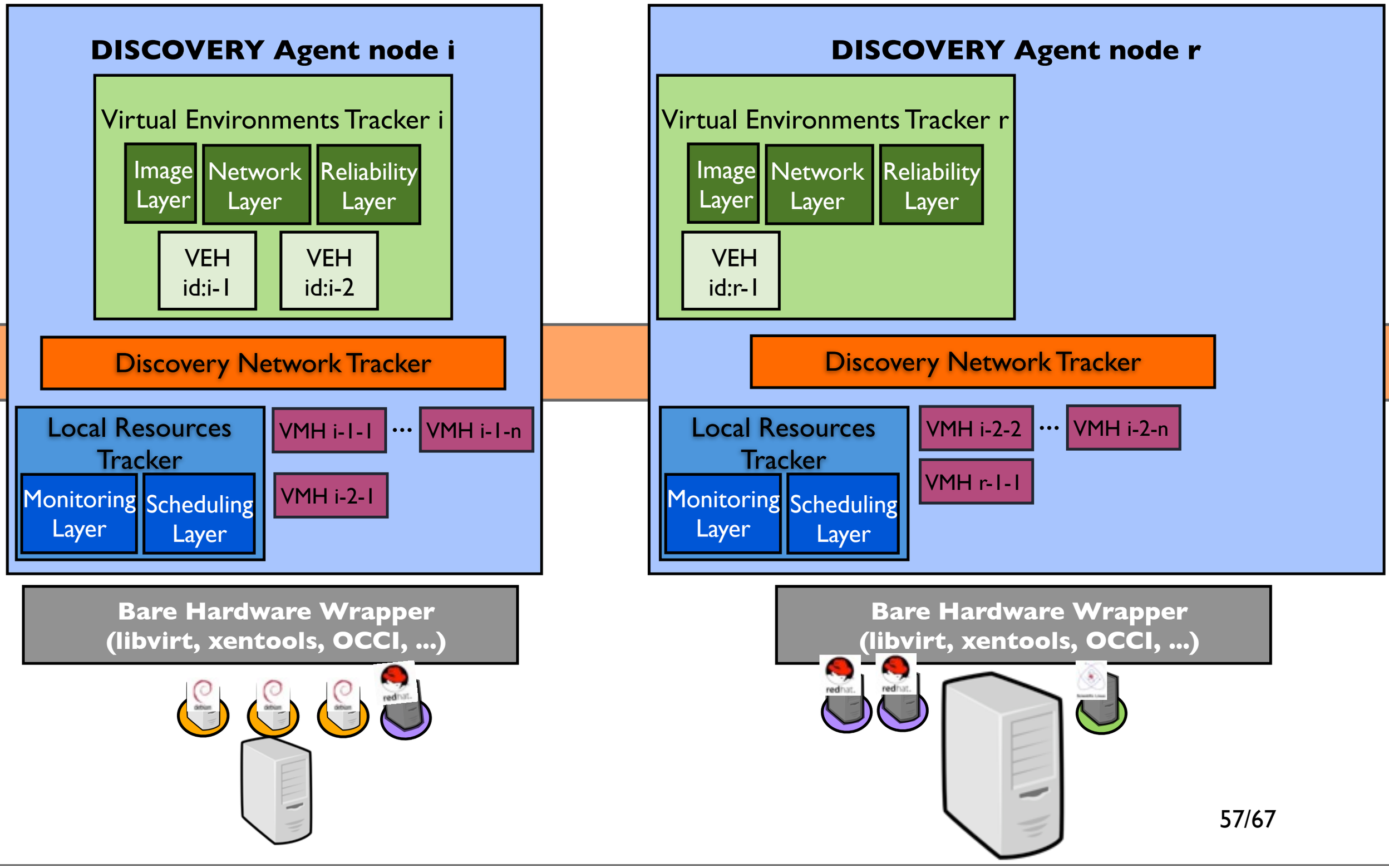
DISCOVERY - VM Crashes



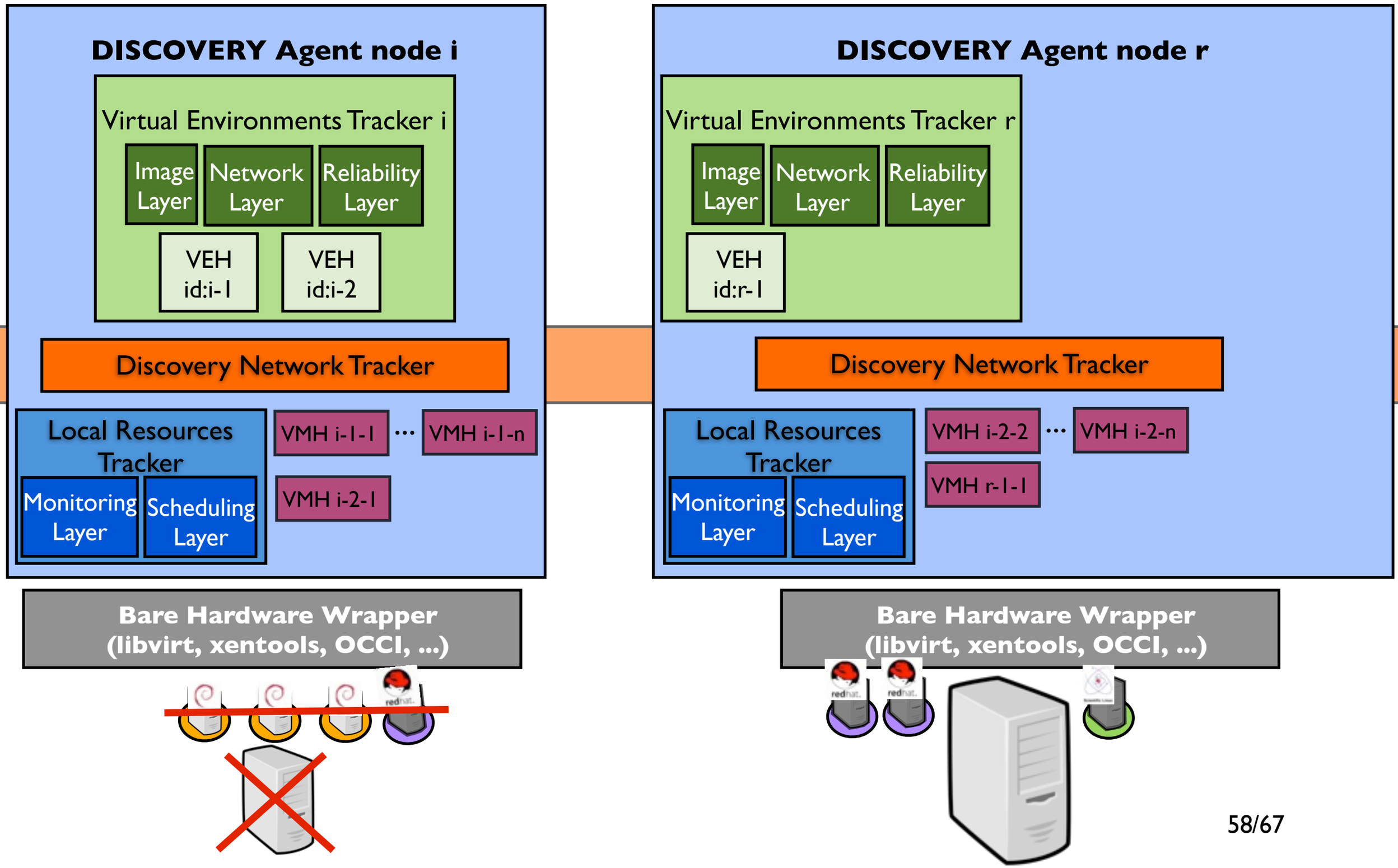
DISCOVERY - VM Crashes



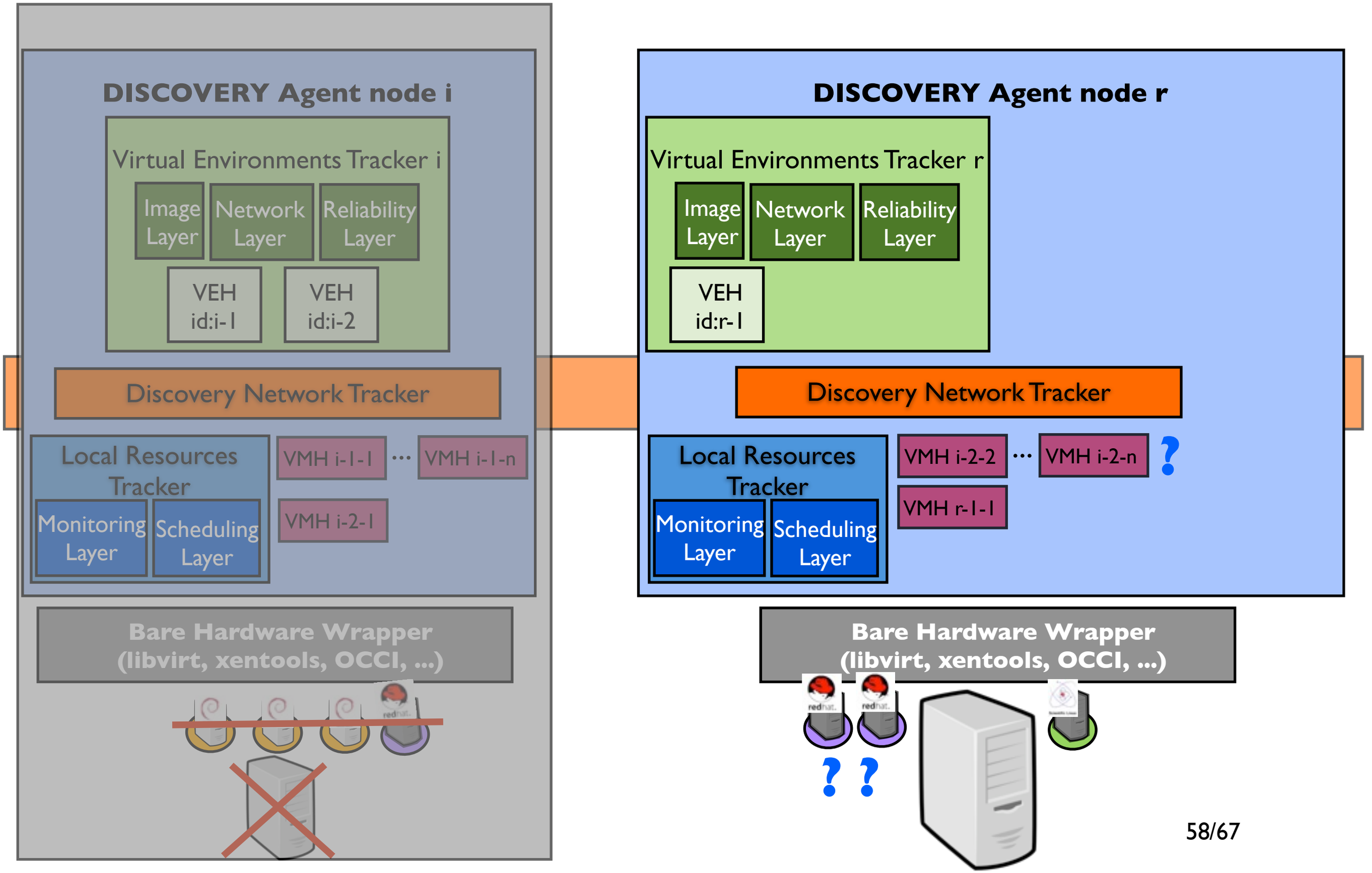
DISCOVERY - VM Crashes



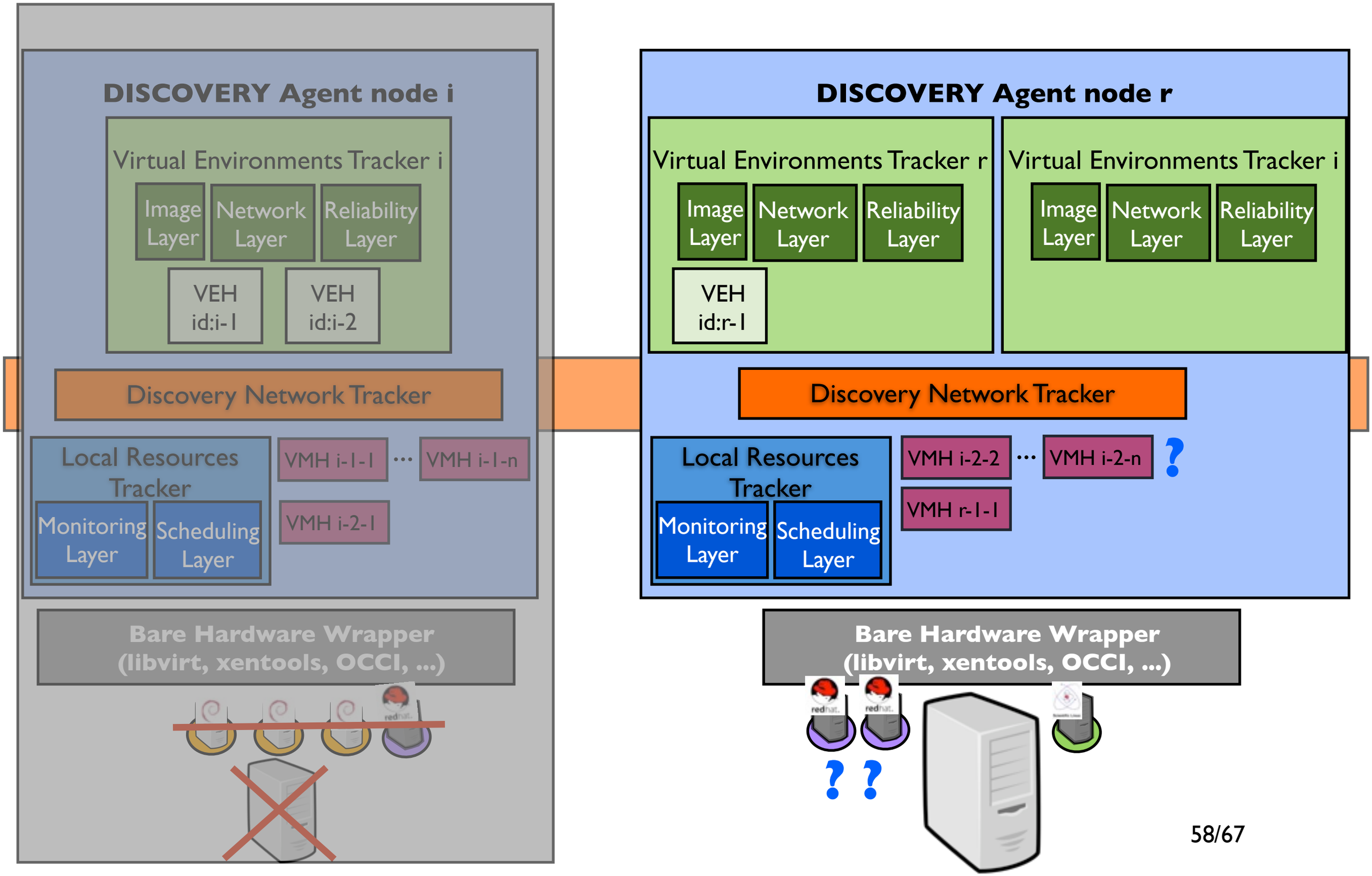
DISCOVERY - Nodes Crashes



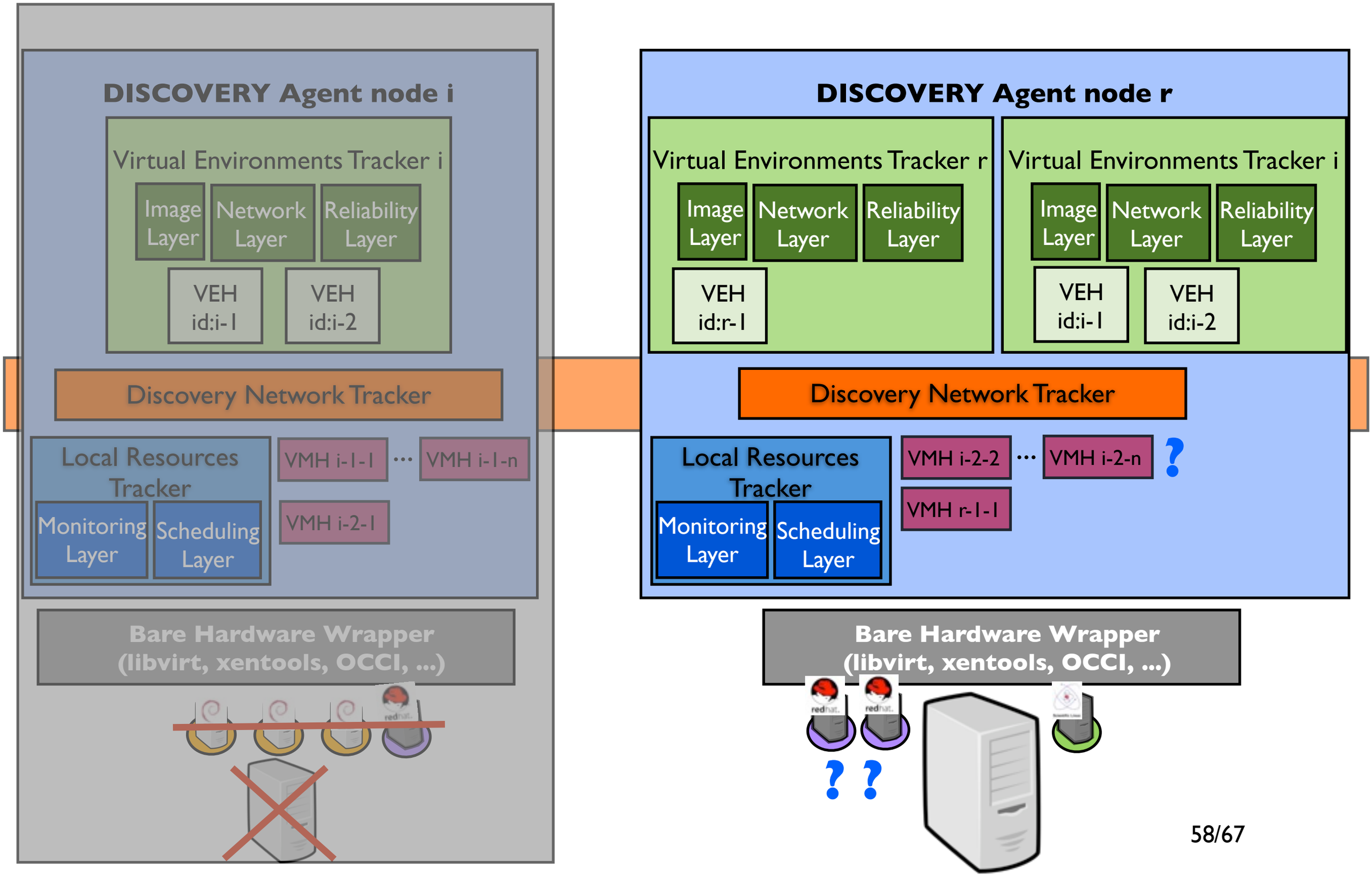
DISCOVERY - Nodes Crashes



DISCOVERY - Nodes Crashes



DISCOVERY - Nodes Crashes



DISCOVERY - Nodes Crashes

DISCOVERY Agent node i

Virtual Environments Tracker i

Image Layer | Network Layer | Reliability Layer

VEH id:i-1 | VEH id:i-2

Discovery Network Tracker

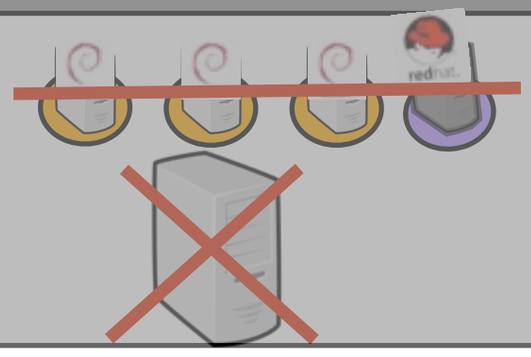
Local Resources Tracker

Monitoring Layer | Scheduling Layer

VMH i-1-1 ··· VMH i-1-n

VMH i-2-1

Bare Hardware Wrapper (libvirt, xentools, OCCI, ...)



DISCOVERY Agent node r

Virtual Environments Tracker r

Image Layer | Network Layer | Reliability Layer

VEH id:r-1

Virtual Environments Tracker i

Image Layer | Network Layer | Reliability Layer

VEH id:i-1 | VEH id:i-2

Discovery Network Tracker

Local Resources Tracker

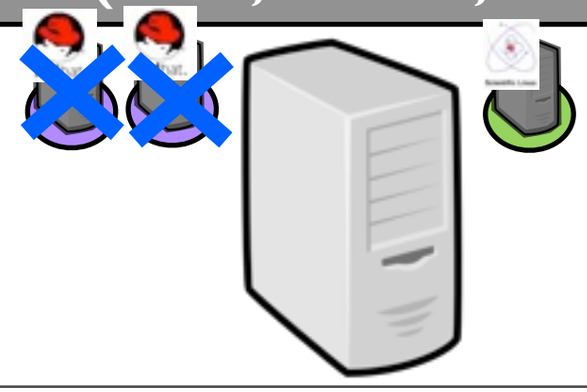
Monitoring Layer | Scheduling Layer

VMH i-2-2 ··· VMH i-2-n

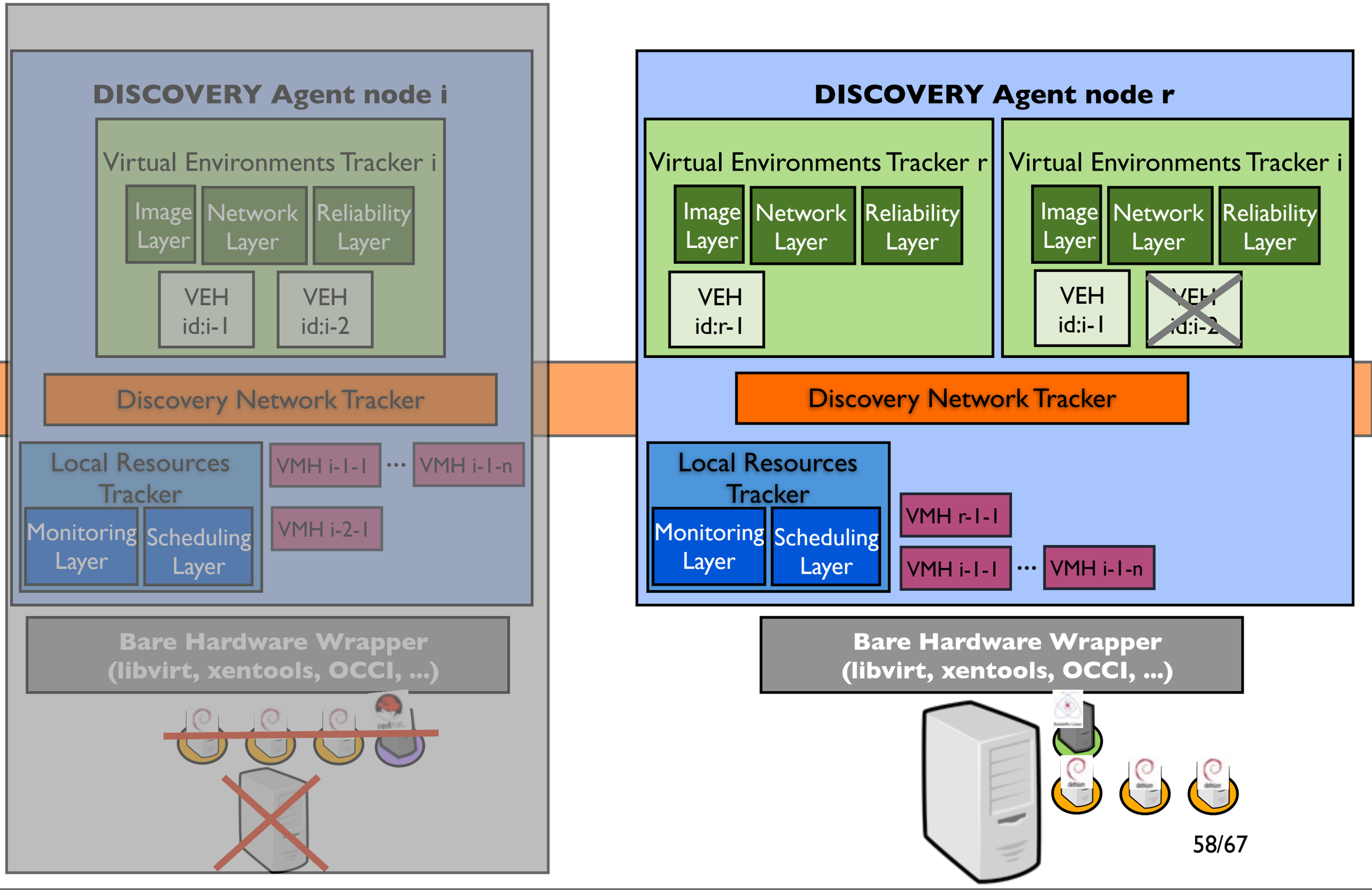
VMH r-1-1

VMH i-1-1 ··· VMH i-1-n

Bare Hardware Wrapper (libvirt, xentools, OCCI, ...)



DISCOVERY - Nodes Crashes



Discovery from the Software Programming Point of View

Background - Actor Model

- Model for concurrent computation
- Actors are the primitive for parallel computing
- Actors communicate with messages
- Actors process only one message at a time

- No shared state between actors

  No lock for data (no barriers)
  scalability

- “Let it crash” pattern

  Erlang
  Fault tolerance

Supervisor and Peer concept

- PeerActor: primitive for developing an application over an overlay (abstraction of the network concerns)

  OverlayActor: Implement the overlay
  NotificationActor: Event bus (overlay modifications)

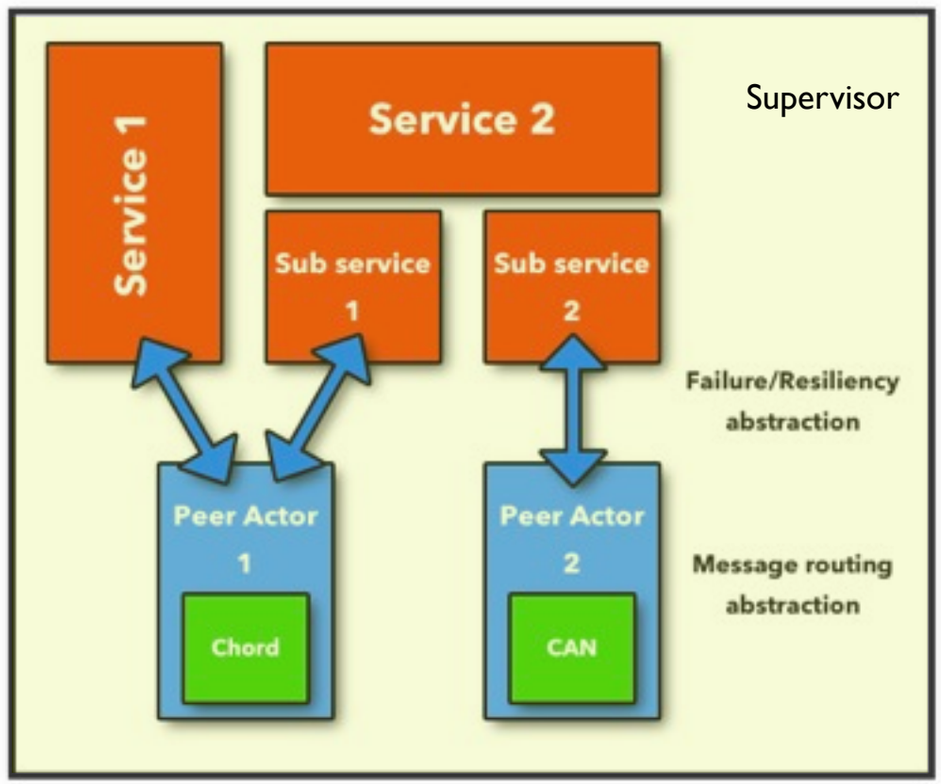
- SupervisorActor: monitoring of sub actors

Analyze failures causes, decides what to do, ...

Credits:J. Pastor - The PeerActor Model, 2013

The Discovery Software Programming Model

- Discovery system will be composed of several services
- Each service will use one or more PeerActor
- Each service will be monitored by a SupervisorActor
- Resiliency will be handled by SupervisorActor



The Discovery Software Programming

- AkkaArc

  PeerActor + SupervisorActor definitions (Scala/Akka)

  A Chord Implementation

- Reimplementation of the DVMS proposal with fault tolerant mechanisms in less than 3 months

- Implementation of distributed IPOP like system (on-going work)

Conclusion

- System Virtualization provides mature "techniques for organizing computer systems resources to provide extraordinary system flexibility" Golberg, 1974
- Leveraging previous works is mandatory
- System Virtualization leads to new concerns (which have not been addressed in this talk)

  Storage (management of VM images/VM migrations through distinct sites)
  Network management (IP assignment, VLAN configurations, suspend/resume)
  Coordination between users and administrators expectations

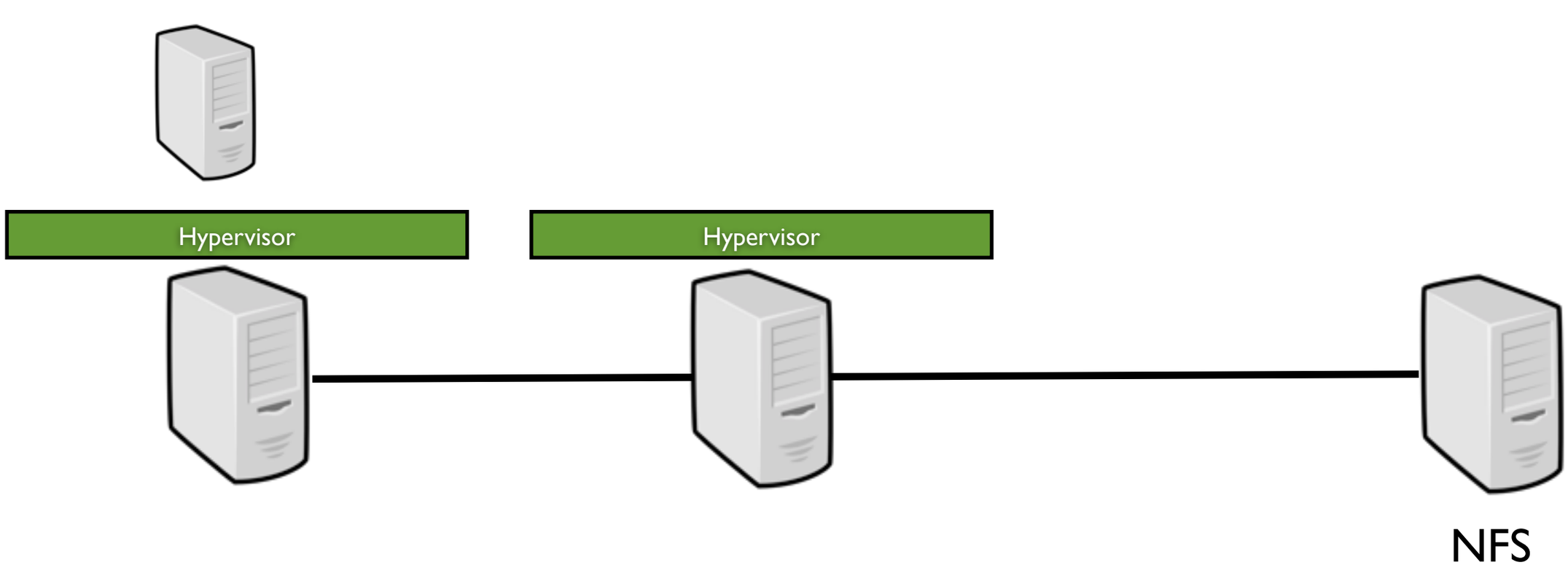
Conclusion

- VM : volatile states vs persistent ones

  VM images (AMI Amazon Machine Image)

- VM migration requires efficient storage mechanisms

  Exploit a distributed file system (NAS/SAN)
  (only volatile memory is transferred)



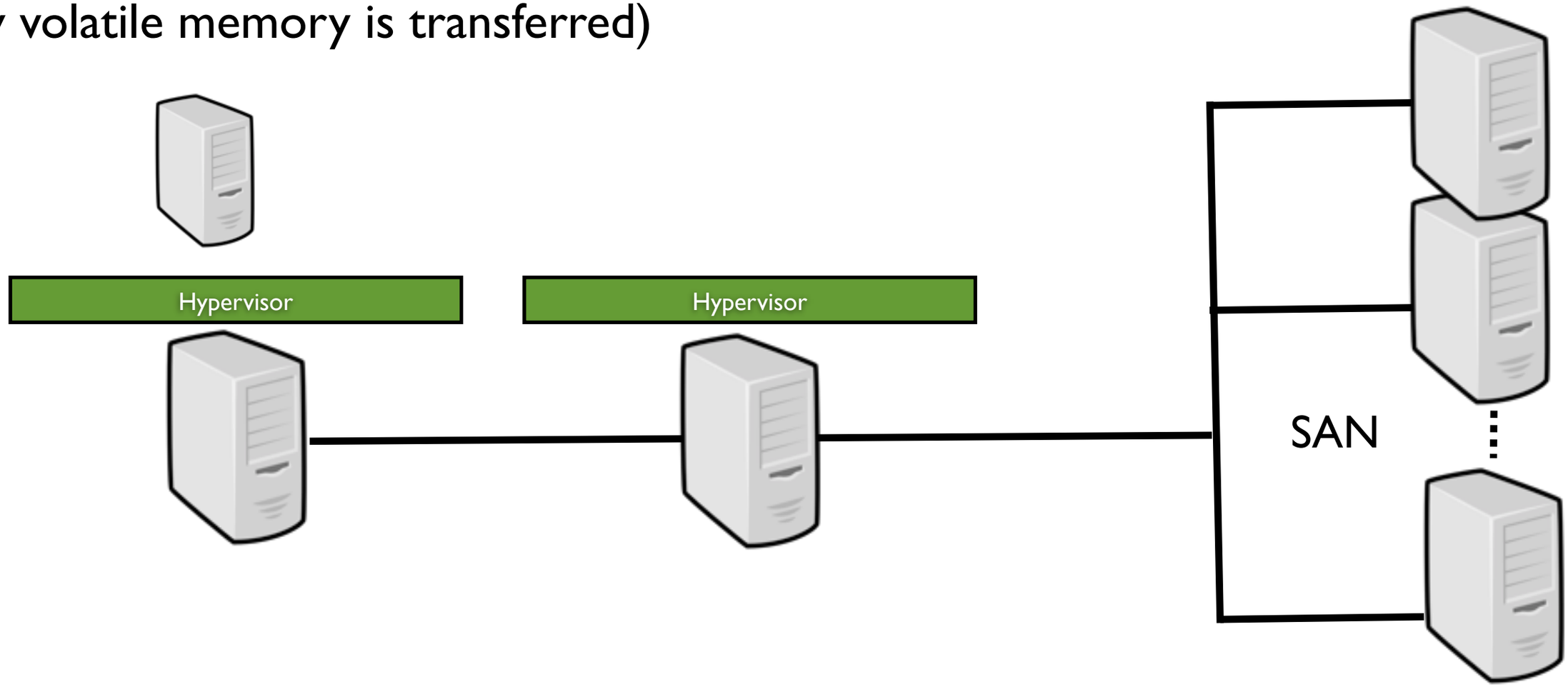
Conclusion

- VM : volatile states vs persistent ones

  VM images (AMI Amazon Machine Image)

- VM migration requires efficient storage mechanisms

  Exploit a distributed file system (NAS/SAN)
  (only volatile memory is transferred)



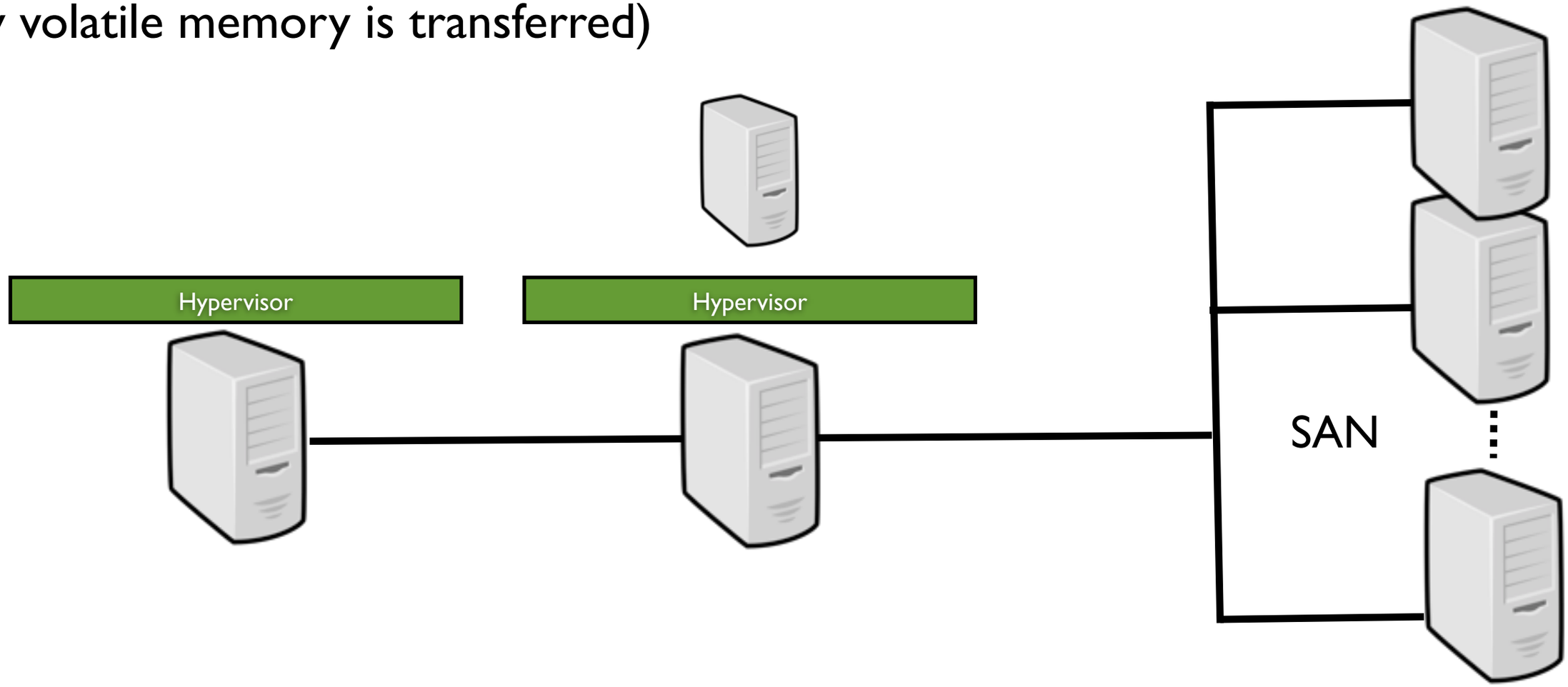
Conclusion

- VM : volatile states vs persistent ones

  VM images (AMI Amazon Machine Image)

- VM migration requires efficient storage mechanisms

  Exploit a distributed file system (NAS/SAN)
  (only volatile memory is transferred)



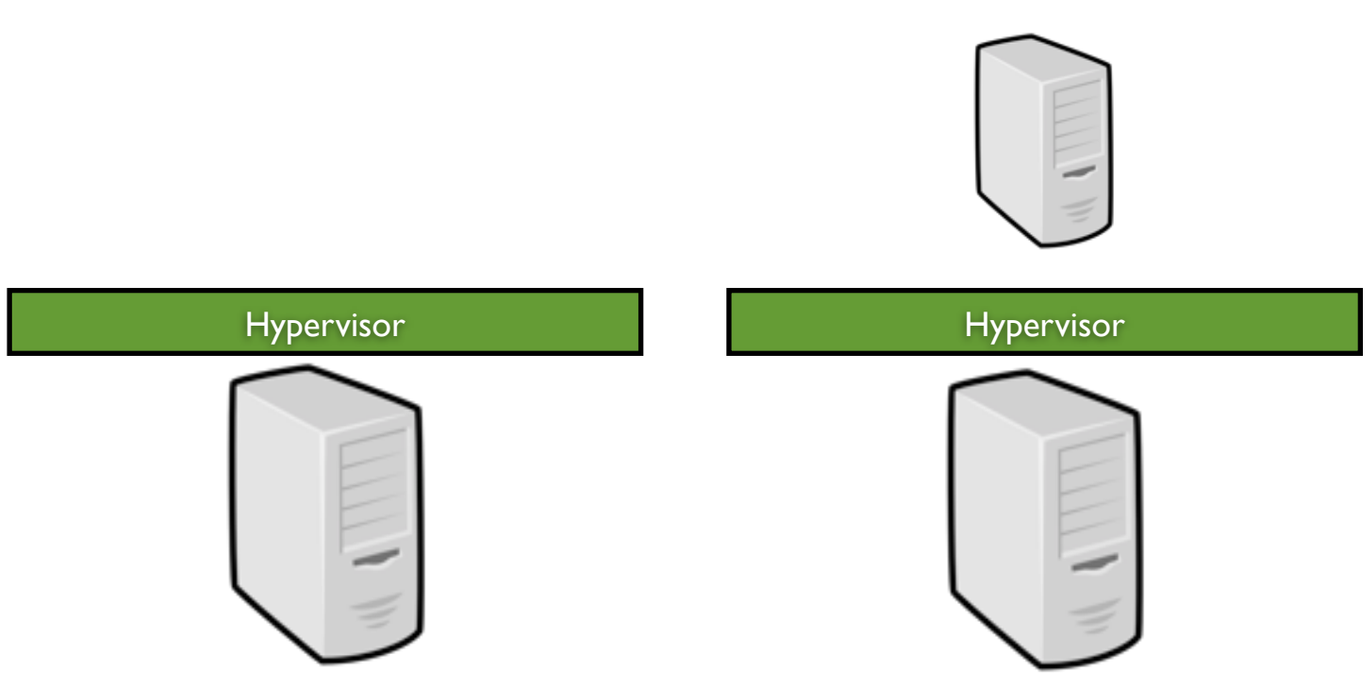
Conclusion

- VM : volatile states vs persistent ones

  VM images (AMI Amazon Machine Image)

- VM migration requires efficient storage mechanisms

  Exploit a distributed file system (NAS/SAN)
  (only volatile memory is transferred)



Copy HDD image from the source to the destination node

Portable but expensive (some contributions such as qcow/backing file solutions, background pre-copy, ..)

Conclusion

- Cloud Computing technology is changing every day

  New features, new requirements

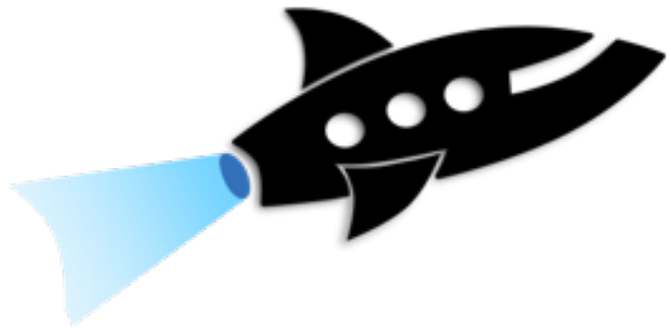
  The main challenge of the Discovery Initiative is to ensure that such new features/mechanisms can run in a distributed manner.

- But Distributed Cloud Computing is happening !

  Dist. CC workshop (collocated with IEEE/ACM UCC 20 13)
  FOG Computing workshop (collocated with IEEE ICC 2013)

Thanks

- The Discovery Initiative
  Past and on-going contributors



Paolo Anedda, Marin Bertier, Frédéric Desprez, Massimo Gaggero, Fabien Hermenier, Flavien Quesnel, Jean-Marc Menaud, Jonathan Pastor, Rémi Pottier, Etienne Riviere, Thomas Ropars, Jonathan Rouzaud-Cornabas, Cédric Tedeschi, Gianluigi Zanetti...

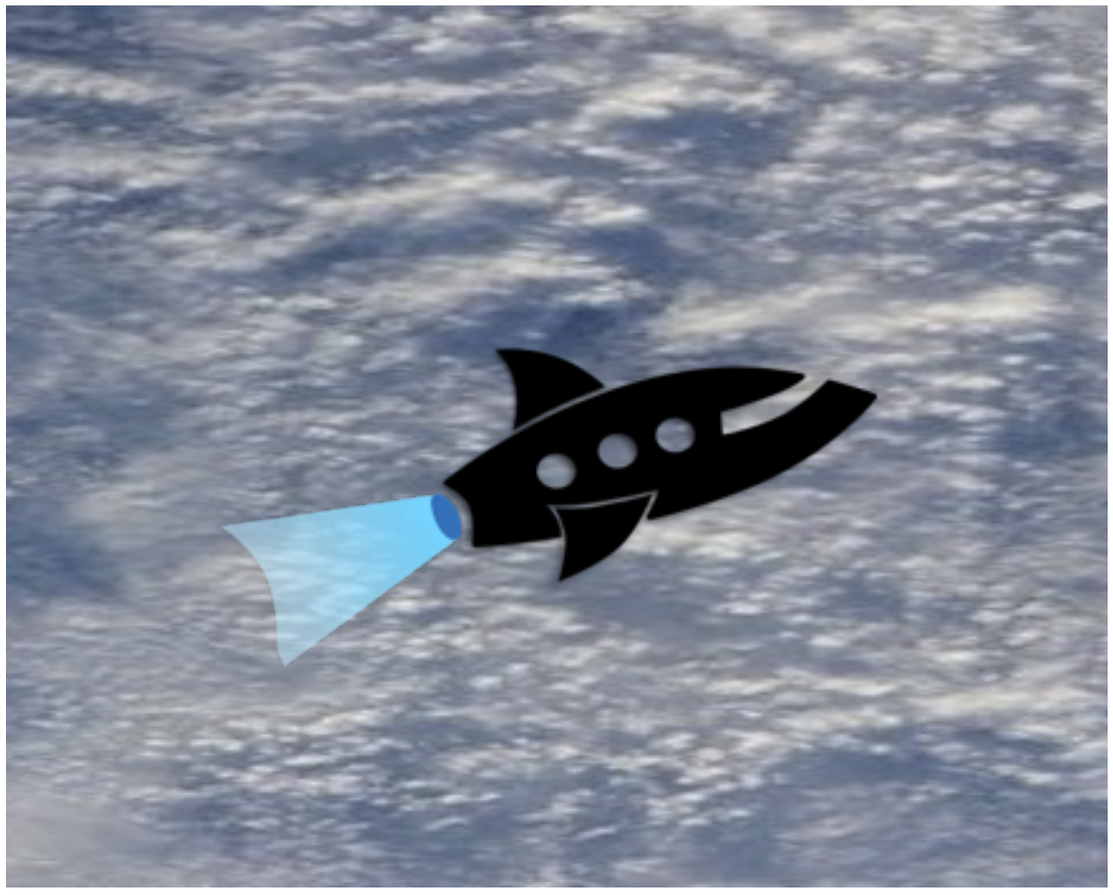
http://beyondtheclouds.github.io/

xxx as Utility

"If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry"

John McCarthy 1961

Beyond the Clouds, the DISCOVERY Initiative



Localization is a key element to deliver
efficient as well as sustainable Utility Computing Solutions

INSTITUT Mines-Télécom

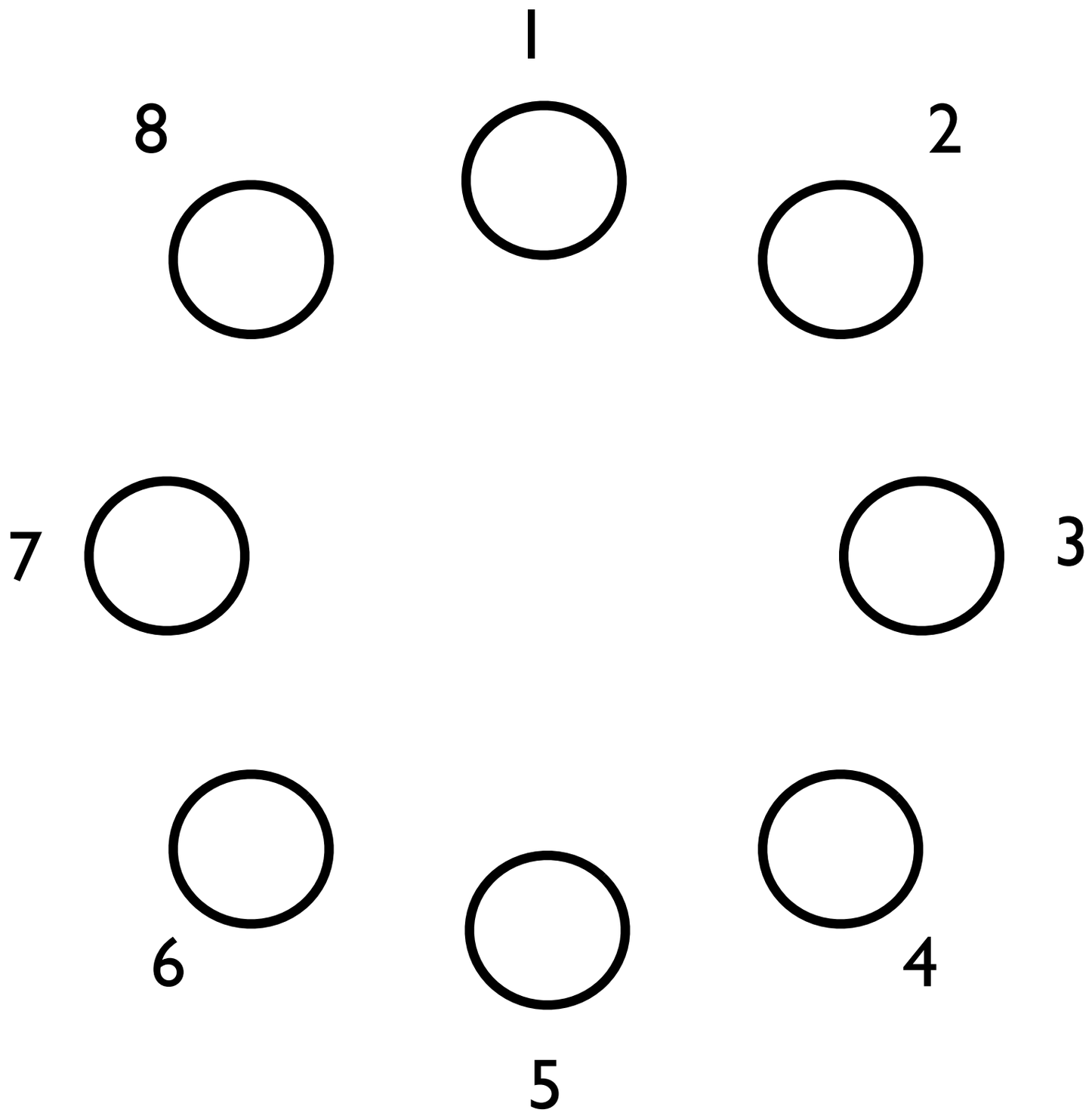
Adrien Lèbre / Ascola Project Team

MINES Nantes

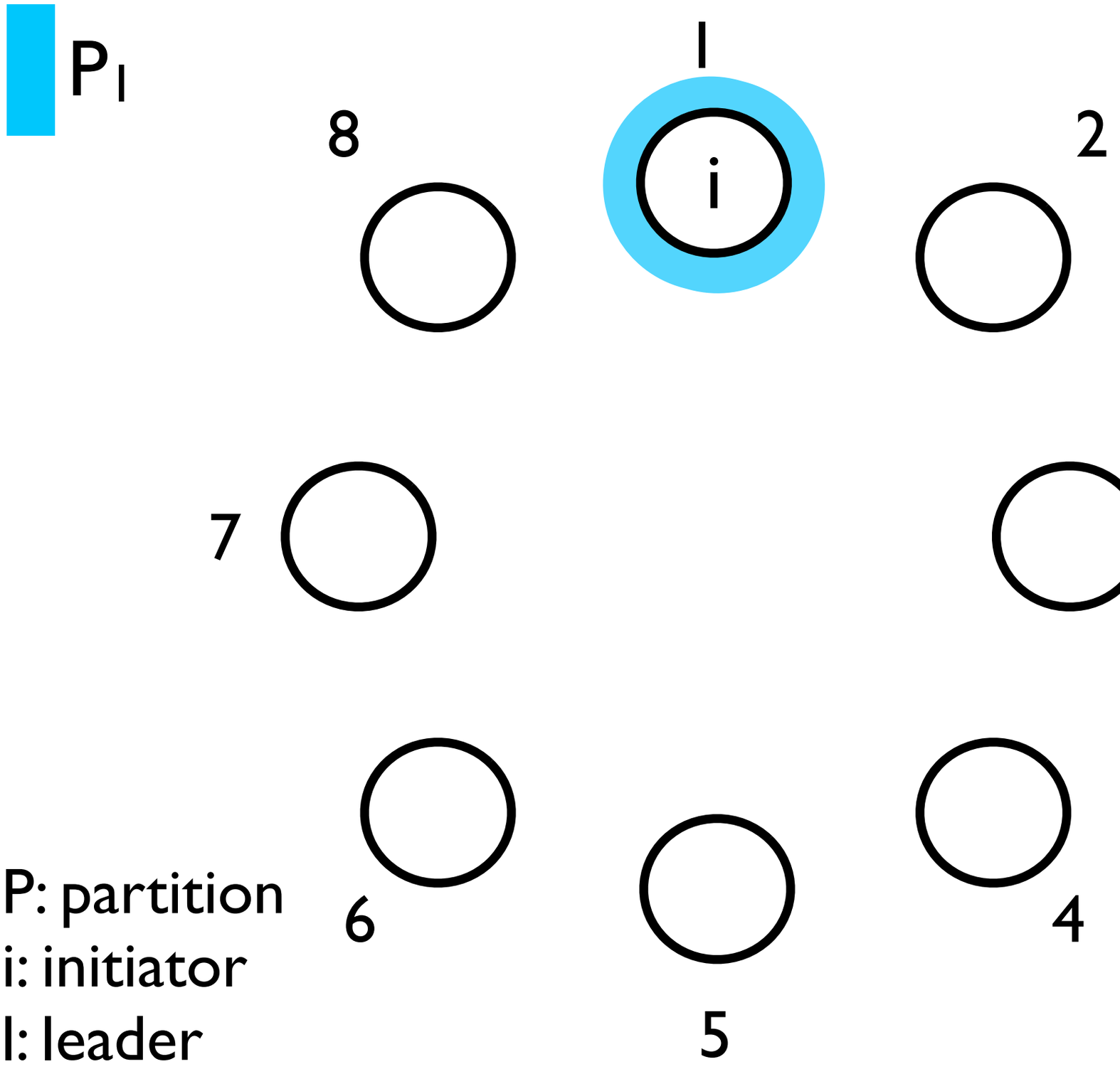
August, 2013

Backup / Advanced Informations

DVMS Proposal - Main Algorithm

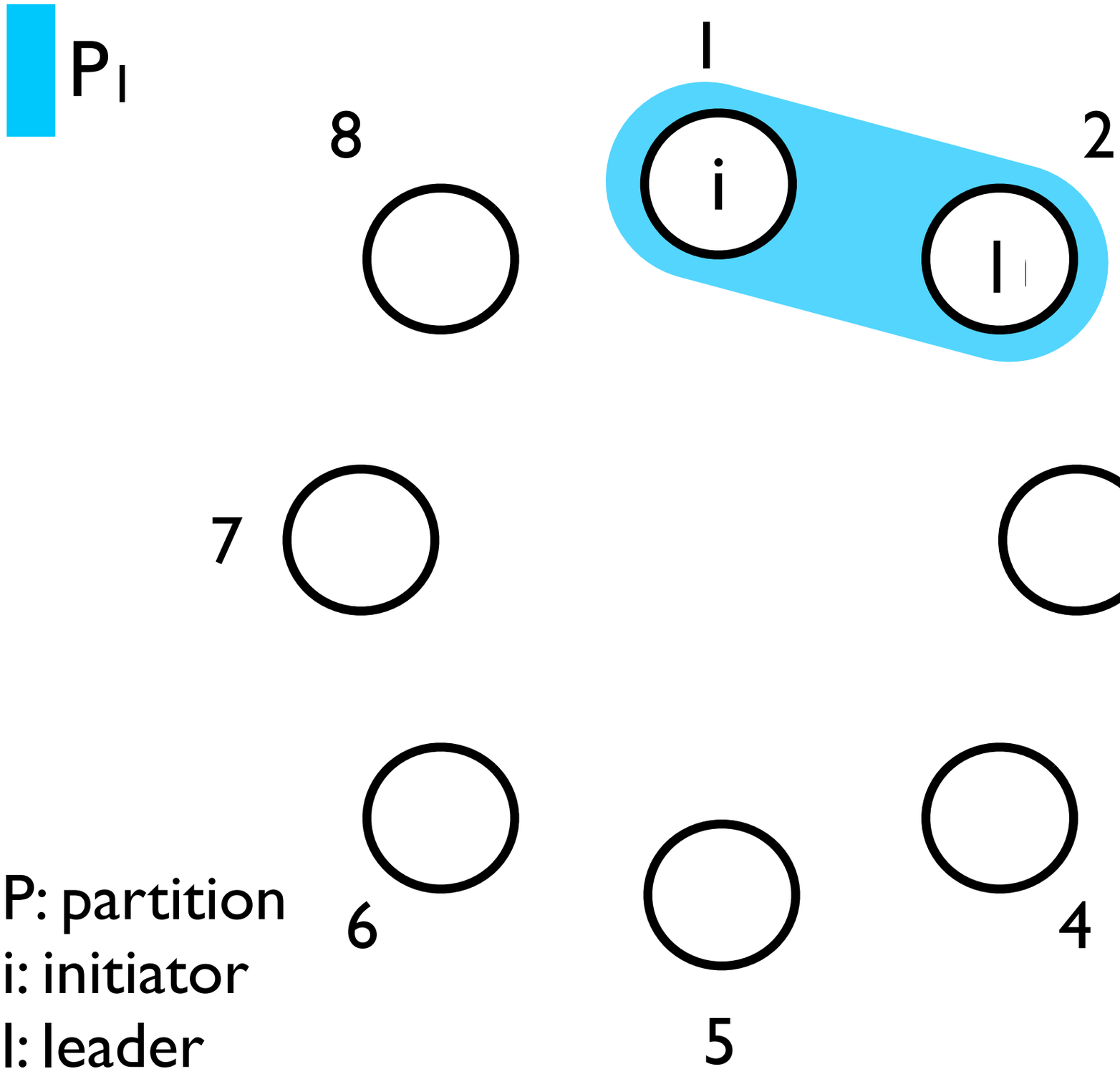


DVMS Proposal - Main Algorithm



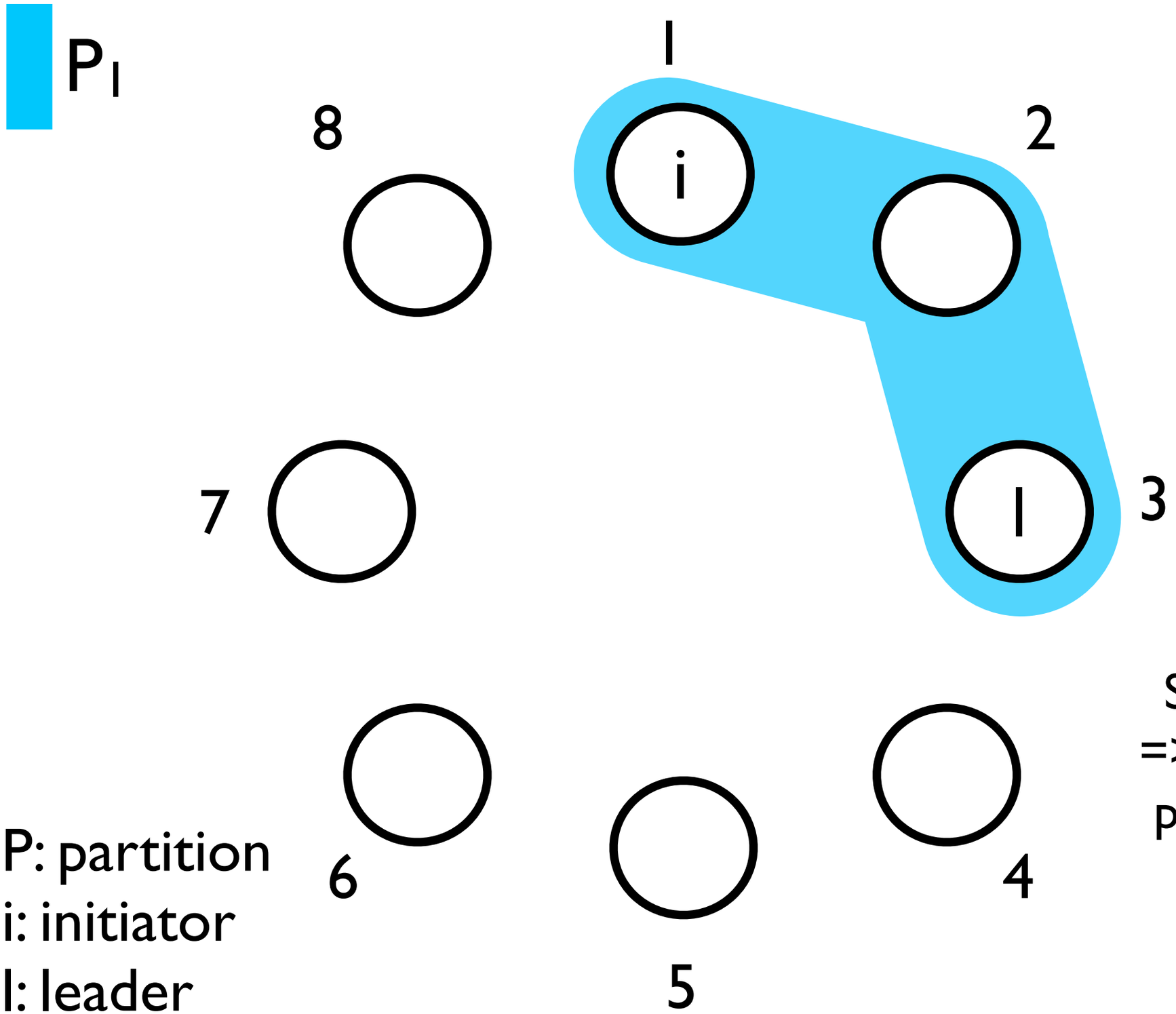
P: partition
i: initiator
l: leader

DVMS Proposal - Main Algorithm



P: partition
i: initiator
l: leader

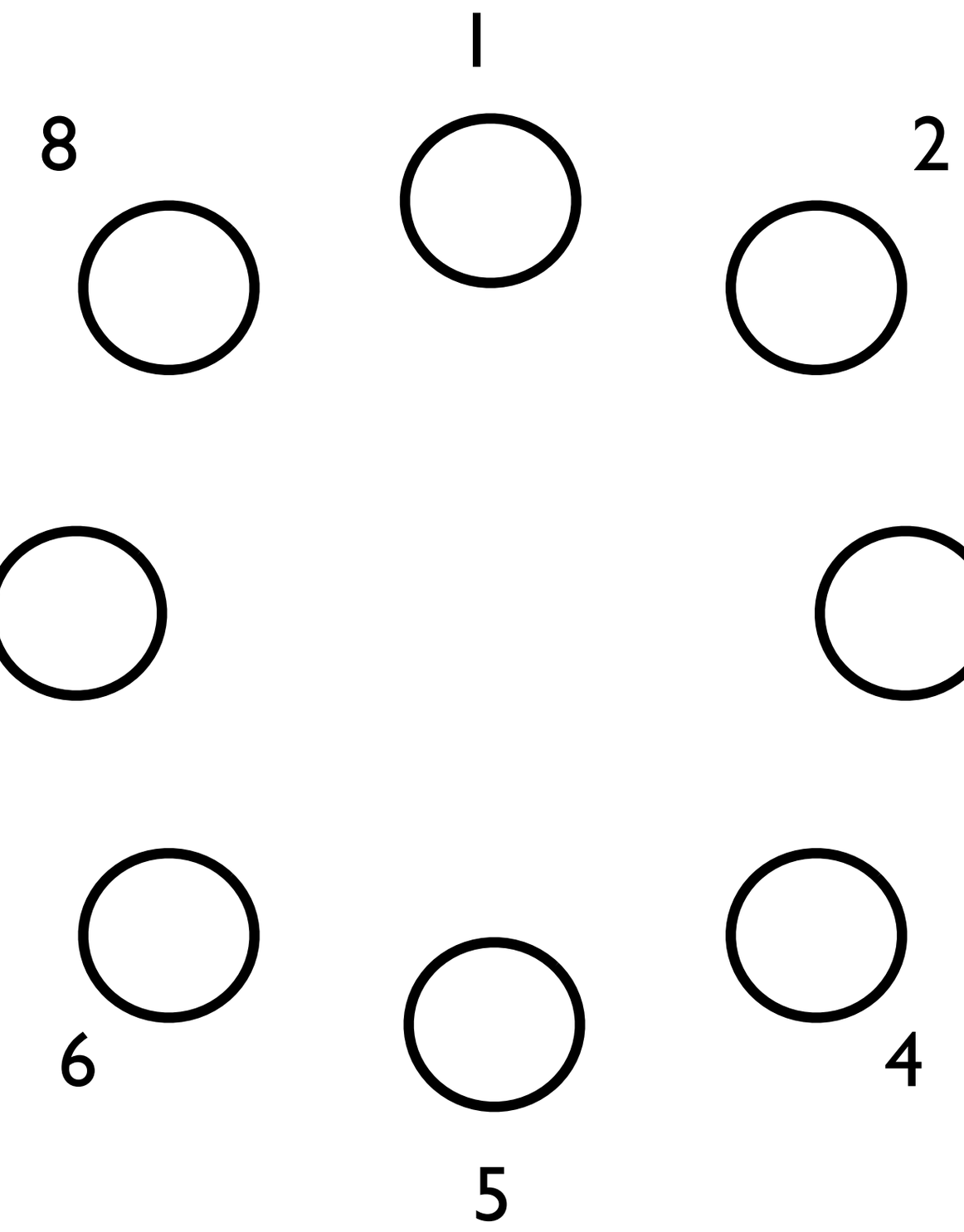
DVMS Proposal - Main Algorithm



P: partition
i: initiator
l: leader

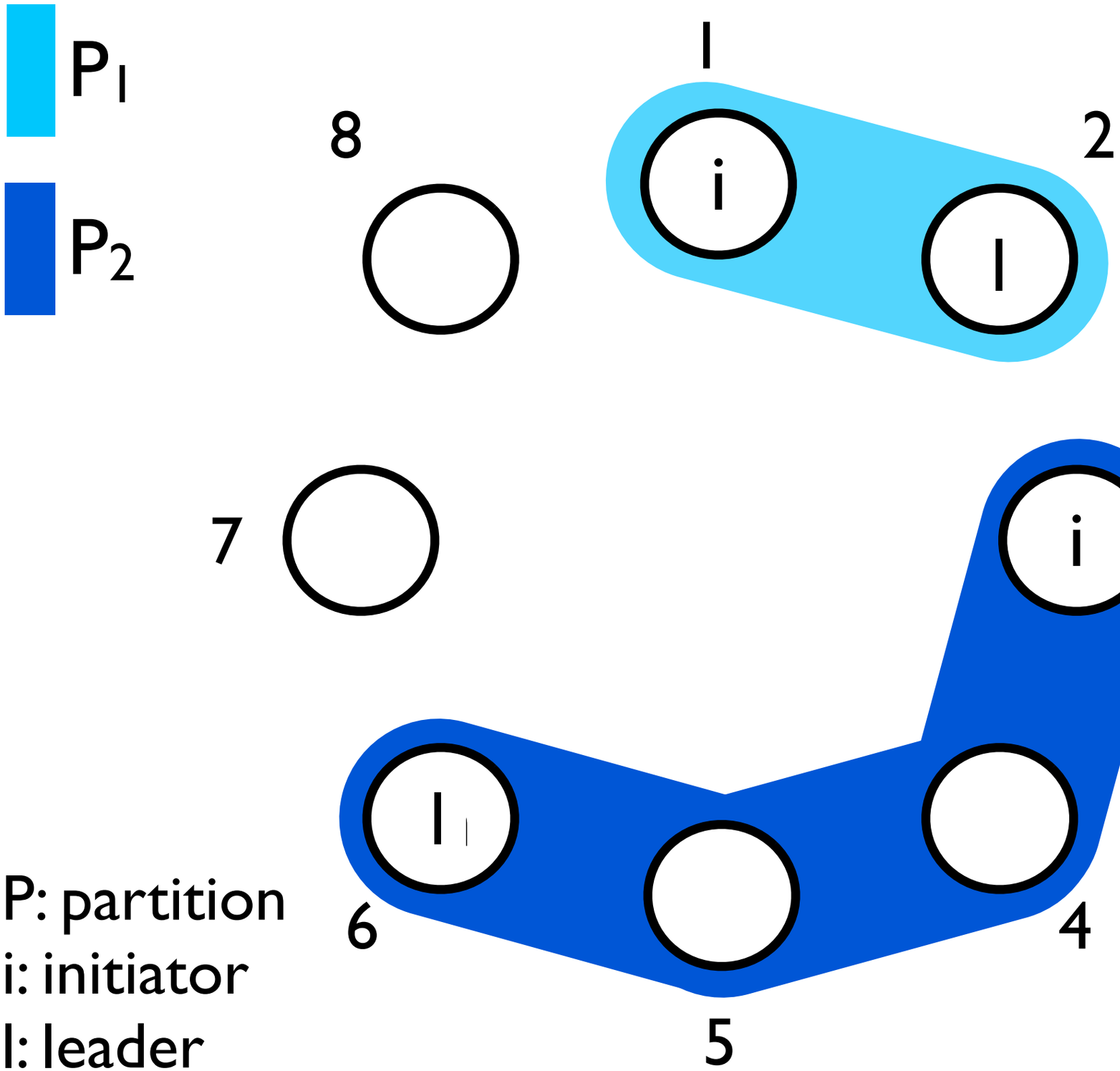
Small partition
=> Computation
phase negligible

DVMS Proposal - Main Algorithm



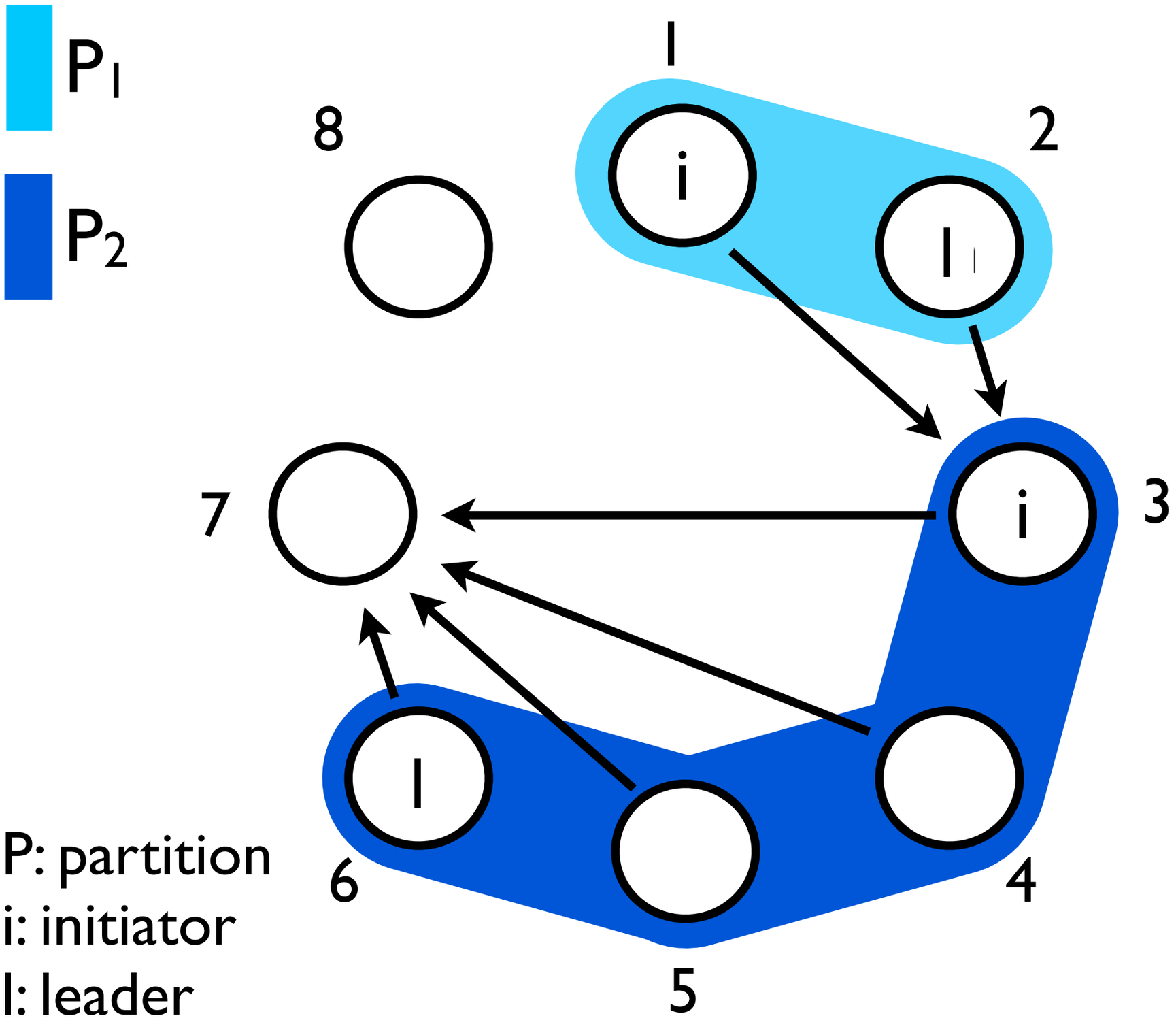
P: partition
i: initiator
l: leader

DVMS Proposal - Shortcuts



P: partition
i: initiator
l: leader

DVMS Proposal - Shortcuts



P: partition
i: initiator
l: leader

DVMS Proposal - Shortcuts