Towards software-defined distributed systems

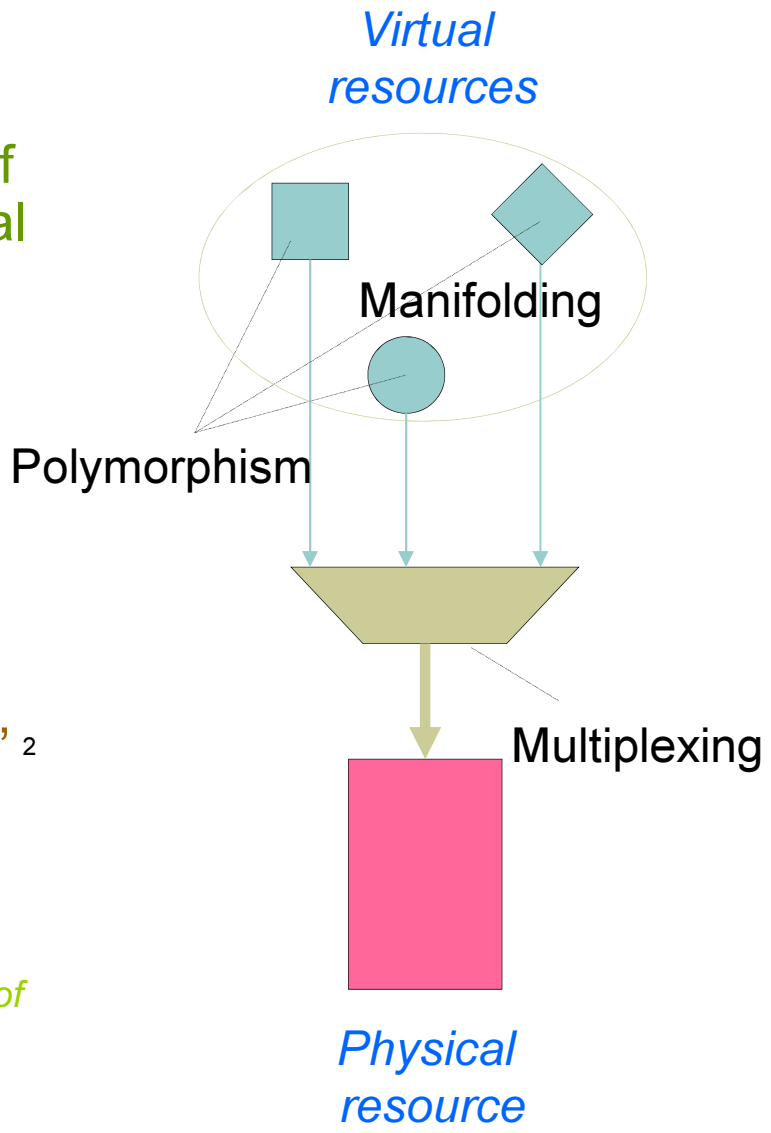
José Fortes
Center for Cloud and Autonomic Computing
Advanced Computing and Information Systems Lab

Outline

- Introduction to
  - Virtualization
  - Service computing
  - Cloud computing
  - Software-defined networking
- Software-defined systems
  - Software-defined virtual networking
  - Software-defined Hadoop
  - Software-defined distributed BLAST
  - Fault-tolerant MapReduce
- Conclusions

"Classic" Virtual Machine

- Copy of a real machine
  - "Any program run under the VM has an effect identical with that demonstrated if the program had been run in the original machine directly" [1]
- Isolated from other virtual machines
  - "…transforms the single machine interface into the illusion of many" [2]
- Efficient
  - "A statistically dominant subset of the virtual processor's instructions is executed directly by the real processor" [2]
- Also known as a "system VM"



[1] *"Formal Requirements for Virtualizable Third-Generation Architectures", G. Popek and R. Goldberg, Communications of the ACM, 17(7), July 1974*

[2] *"Survey of Virtual Machine Research", R. Goldberg, IEEE Computer, June 1974*

Process vs. System VMs

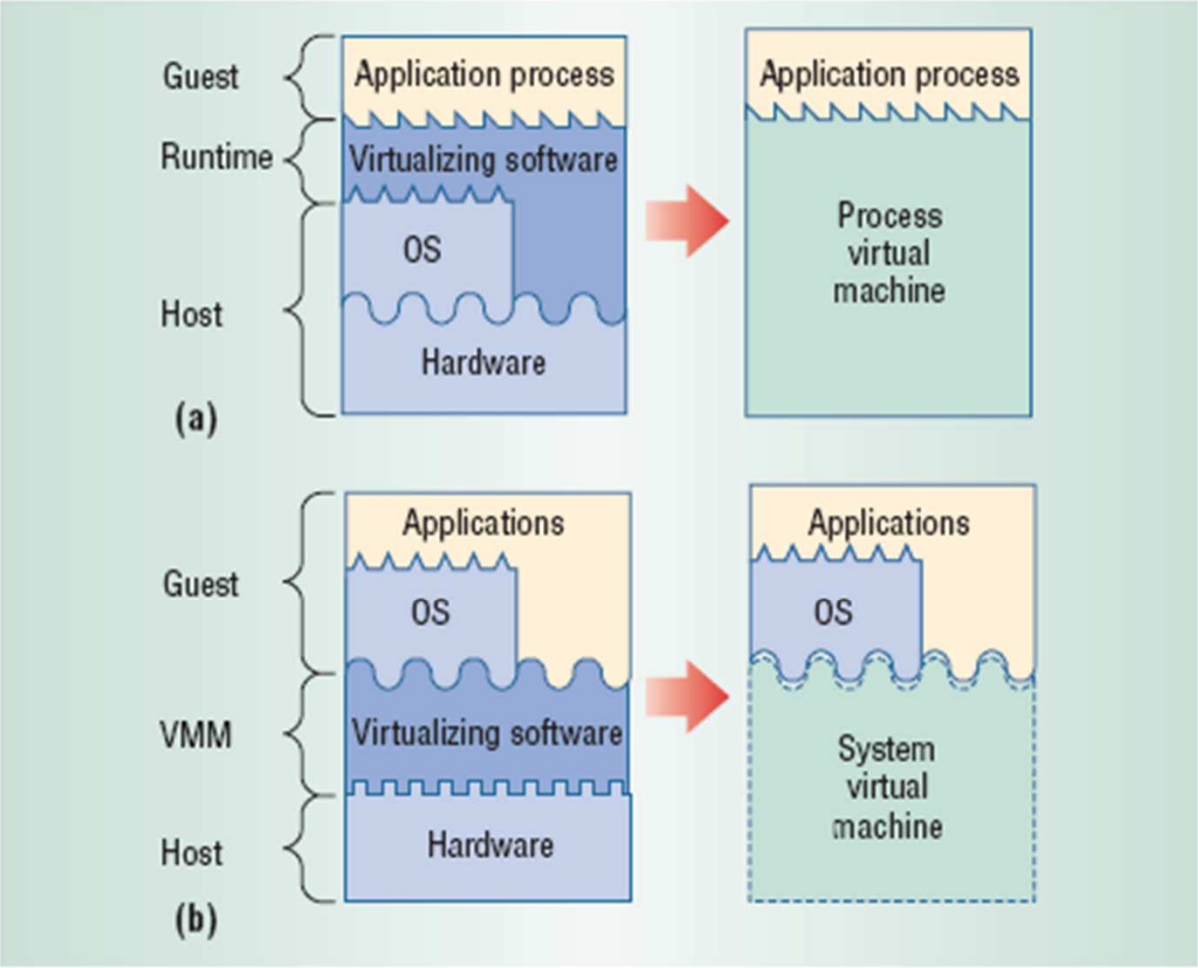
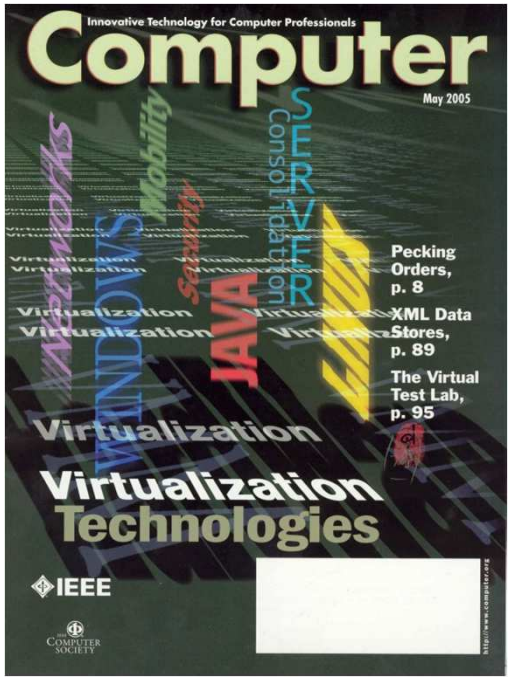


Figure 3. Process and system VMs. (a) In a process VM, virtualizing software translates a set of OS and user-level instructions composing one platform to those of another. (b) In a system VM, virtualizing software translates the ISA used by one hardware platform to that of another.

- In Smith and Nair's "The architecture of Virtual machines", Computer, May 2005



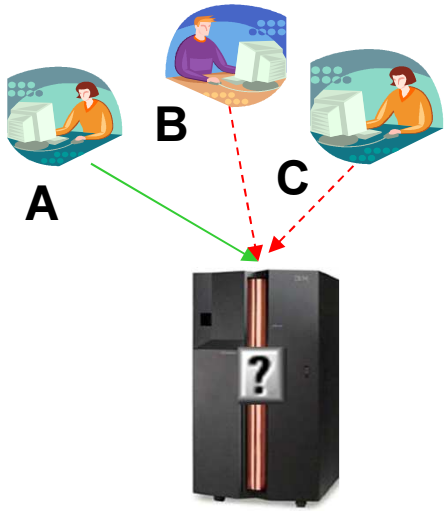
Classic/System Virtual Machines

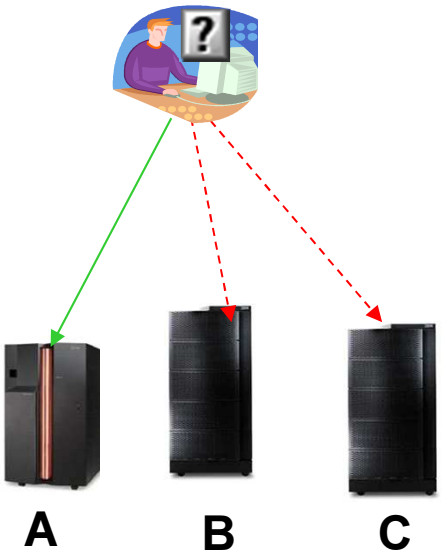
- Virtualization of instruction sets (ISAs)
  - Language-independent, binary-compatible (*not* JVM)
- 70's (IBM 360/370..) – 00's
  - (VMware, KVM, Microsoft Virtual Server/PC, z/VM, Xen, Power Hypervisor, Intel Vanderpool, AMD Pacifica, Virtual Box …)
- ISA+ OS + libraries + software = execution environment

- VM image = file(s) including the OS + applications + configuration that the VM will run once started.
- VM instance = a running VM.
- VM snapshot = state + data of a VM at a point in time

Virtualization technology

- Why is it good?
  - In one word: decoupling
  - Enables separation of concerns among IT layers
  - E.g. resource layer delivers virtual resources, application layer configures resources to run applications as services, provider layer configures services to serve users …





Case in point …

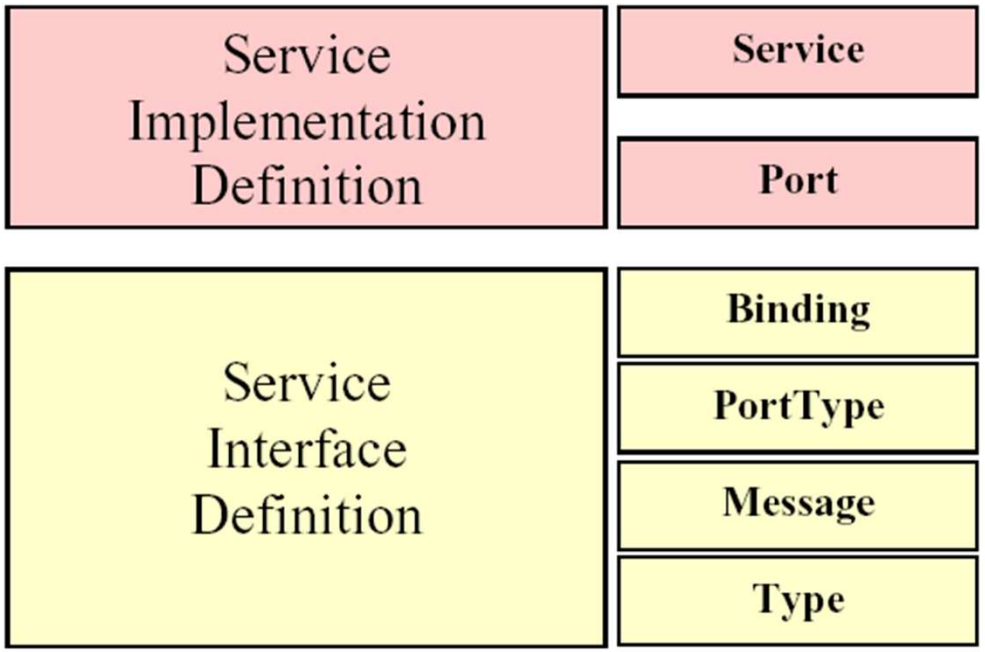
- **Virtualization to aid Intel in saving up to $1.8B through data center consolidation**
  - **Electronic News, 10/30/2007**
- To significantly reduce its total data center footprint and save between $1.4 and $1.8 billion over 7 years by replacing older technology with new multi-core Xeon processors, along with using techniques such as virtualization, Intel Corp. is reporting today that it is consolidating its 130 data centers worldwide to just 8 global hubs.
- "more images onto a server as opposed to what we have today: 2,500 applications that are normally on one server. Our ratio today is 1:1. One server actually has one operating system and we want to try to reduce that to a 4:1 …" (Brently Davis)

Services computing

- Aim “to enable IT services and computing technology to perform business services efficiently and effectively”
- Technology suite:
  - Web Services and Service-Oriented Architecture (SOA)
  - Cloud computing
  - Business process modeling, transformation and integration
- Scope: life-cycle of services innovation research
  - componentization, modeling, creation, realization, annotation, deployment, discovery, composition, delivery, collaboration, monitoring, optimization, management …

Basic service description: interface definition

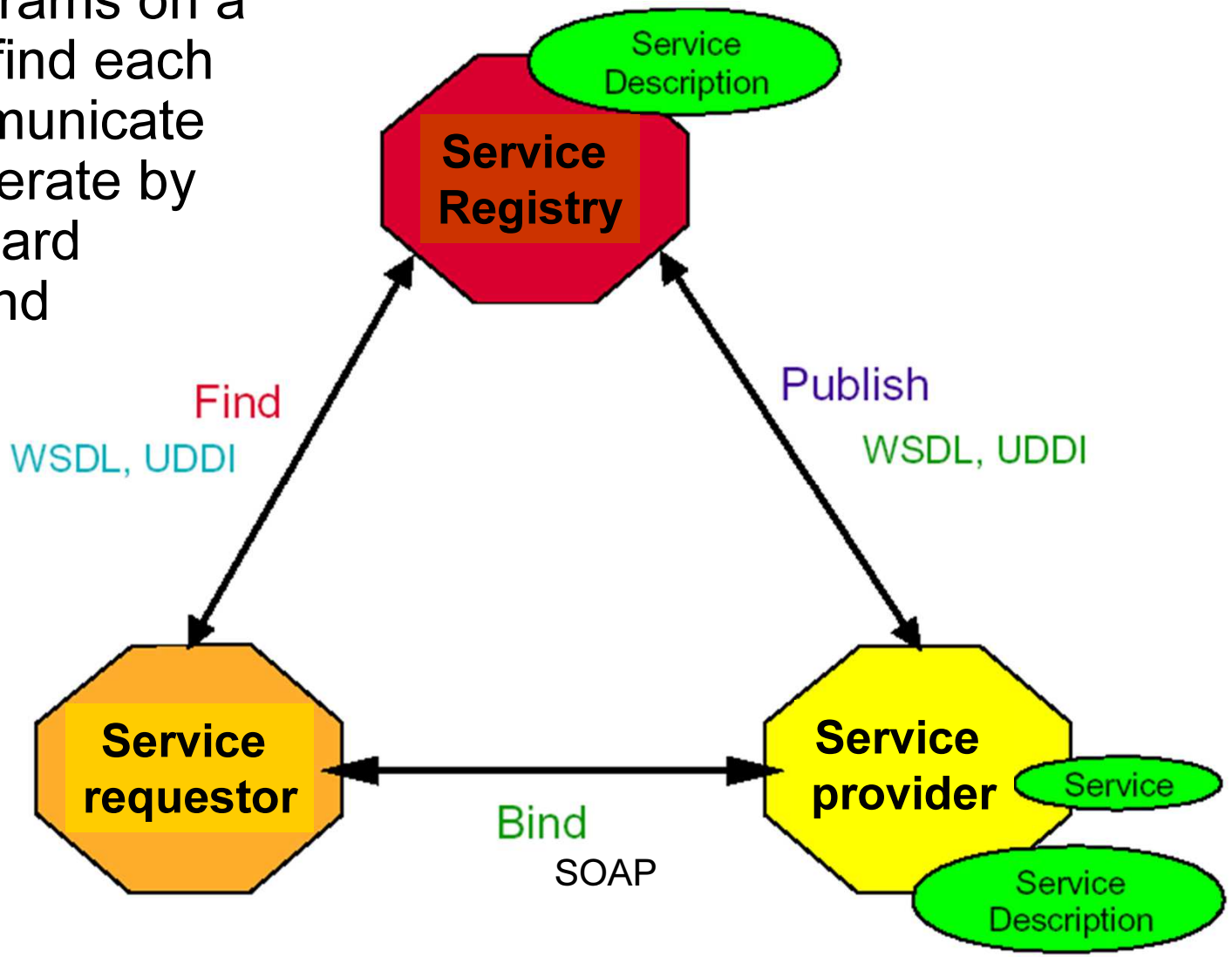
- abstract or reusable service definition that can be instantiated and referenced by multiple service implementation definitions



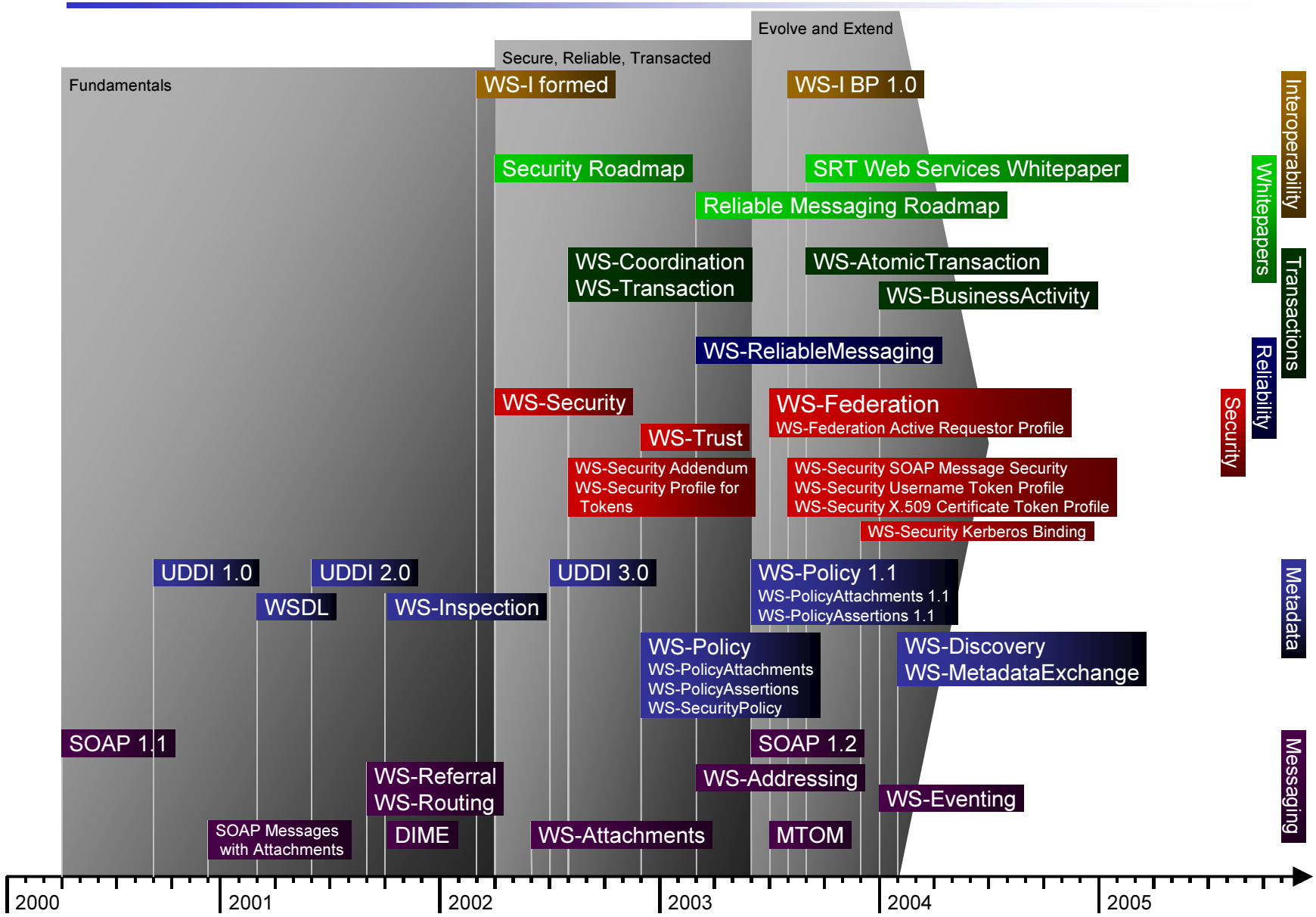
- different implementations using the same application can be defined to reference different service definitions – a form of virtualization

Web services framework

- allows programs on a network to find each other, communicate and interoperate by using standard protocols and languages



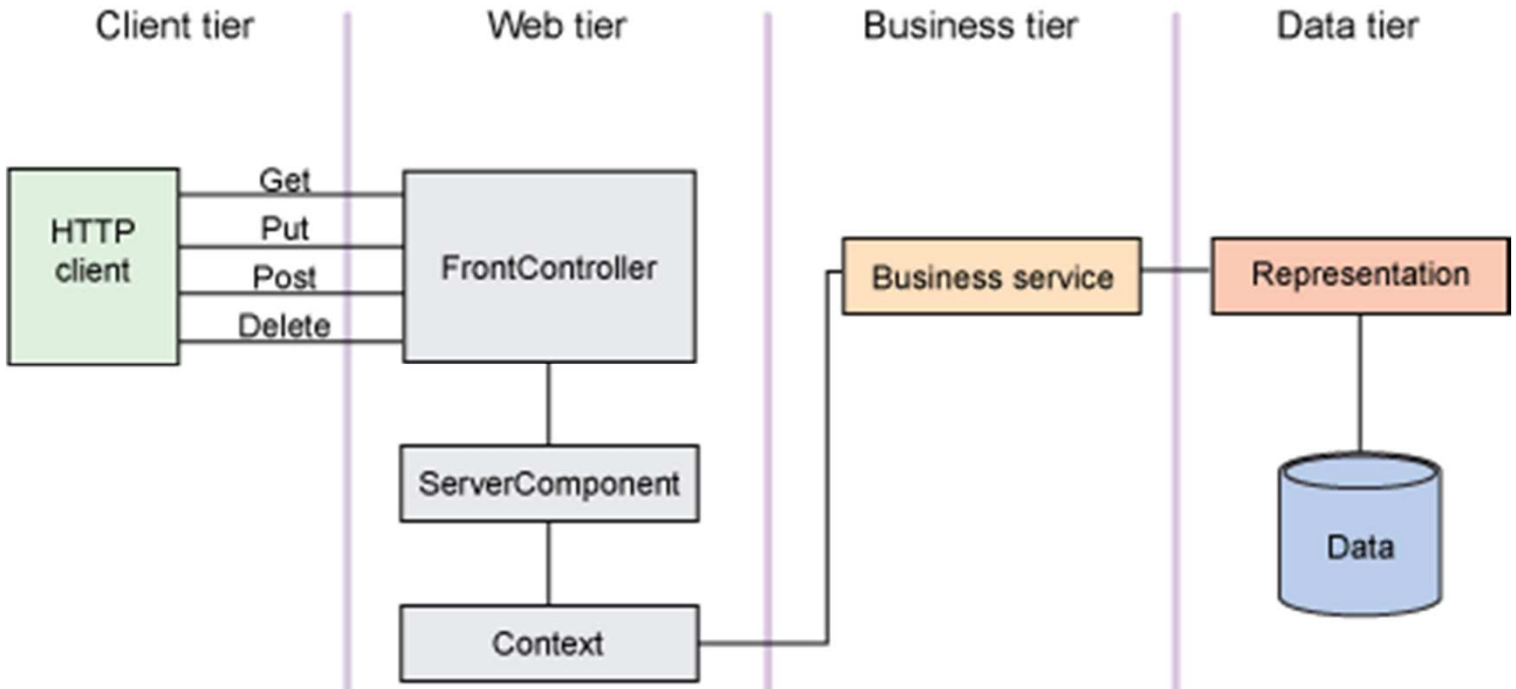
Web Services Architecture *Timeline (AS OF 2/2004)*



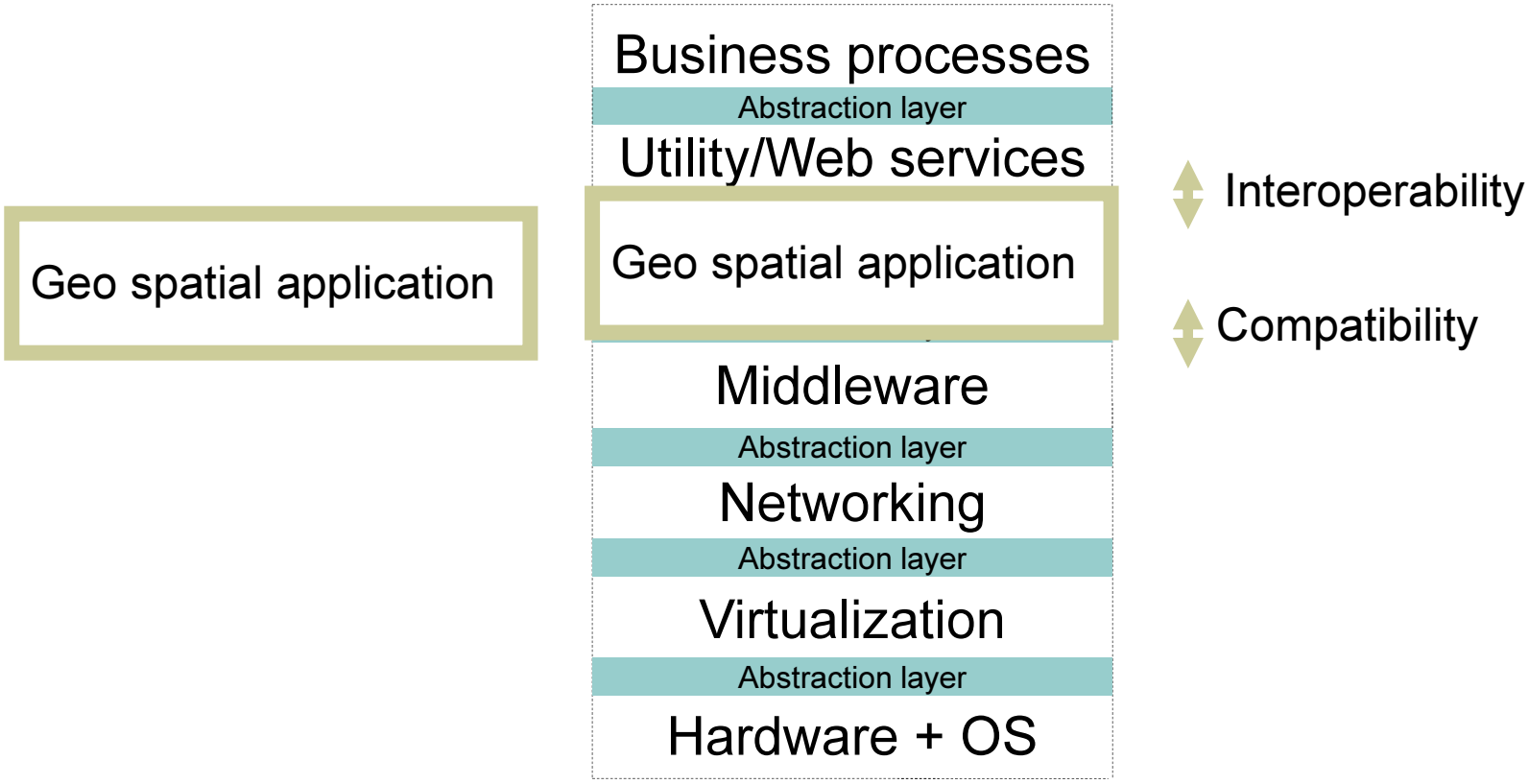
Provided by Mazin Yousif, Credit to someone unknown

REST Services

- **REST(Re**presentational **S**tate **T**ransfer) architecture
  - Resources as URLs
    - Many, small, linking to others and cacheable
  - No connection state (no cookies, requests carry all info)
  - Client-server, HTTP for Create/Read/Update/Delete (CRUD)



Interoperability/compatibility



Another case in point ...

- **Amazon Elastic Compute Cloud**, also known as "EC2", is a commercial web service which allows paying customers to rent computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a web services interface through which customers can request an arbitrary number of Virtual Machines, i.e. server instances, on which they can load any software of their choice. Current users are able to create, launch, and terminate server instances on demand, hence the term "elastic".

NIST Cloud Computing Definition

- a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. NIST SP 800-145, SP 500-292

- a model where resources … are abstracted and provided as services on the Internet… docs.openstack.org

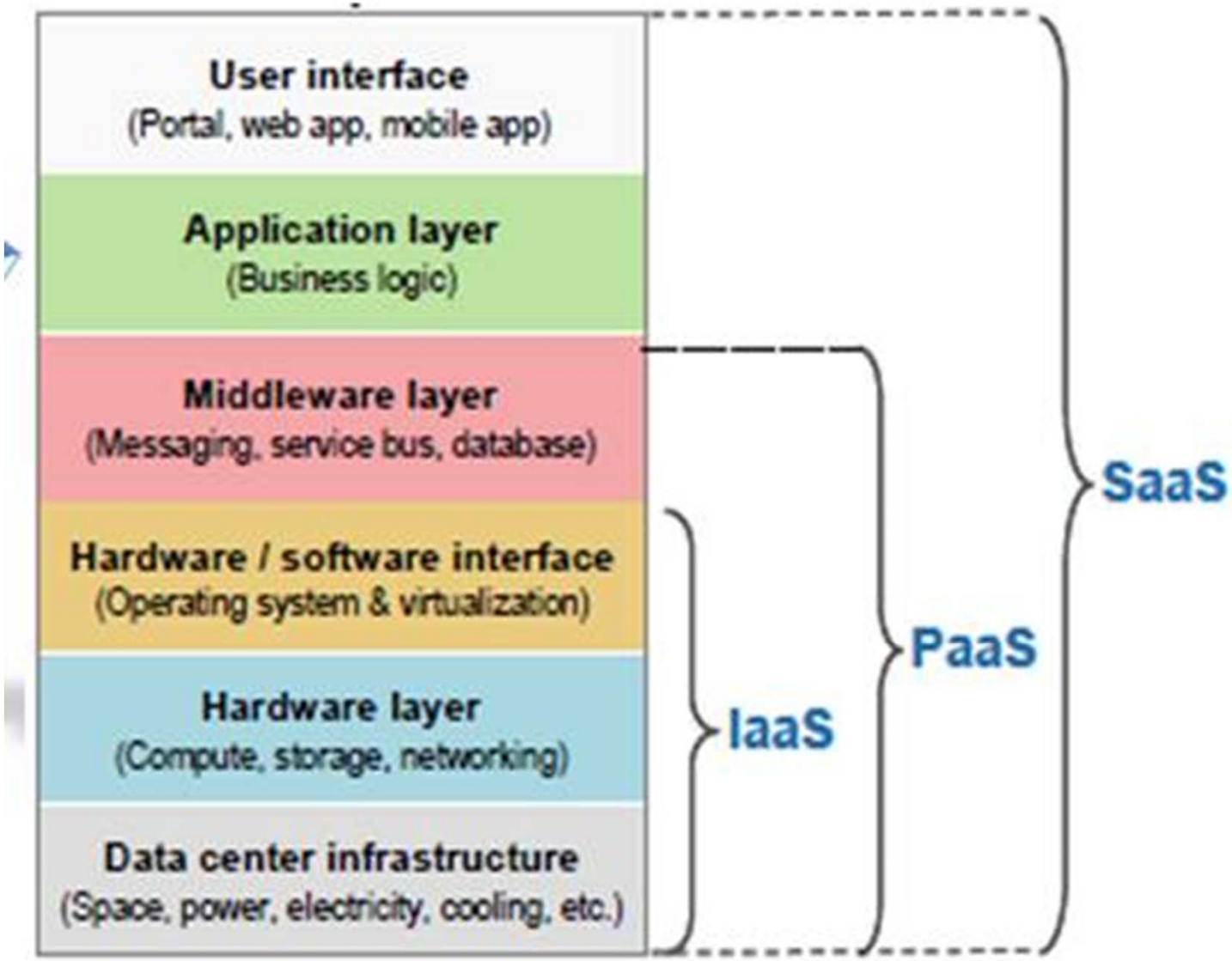
NIST's 5 cloud characteristics

- On-demand self-service
  - When needed at customer's initiative
- Broad network access
  - Access through standard networks
- Resource pooling
  - Transparent sharing and location of resources
- Rapid elasticity
  - So one can use as many resources as needed
- Measured Service
  - So one can pay/be paid per use

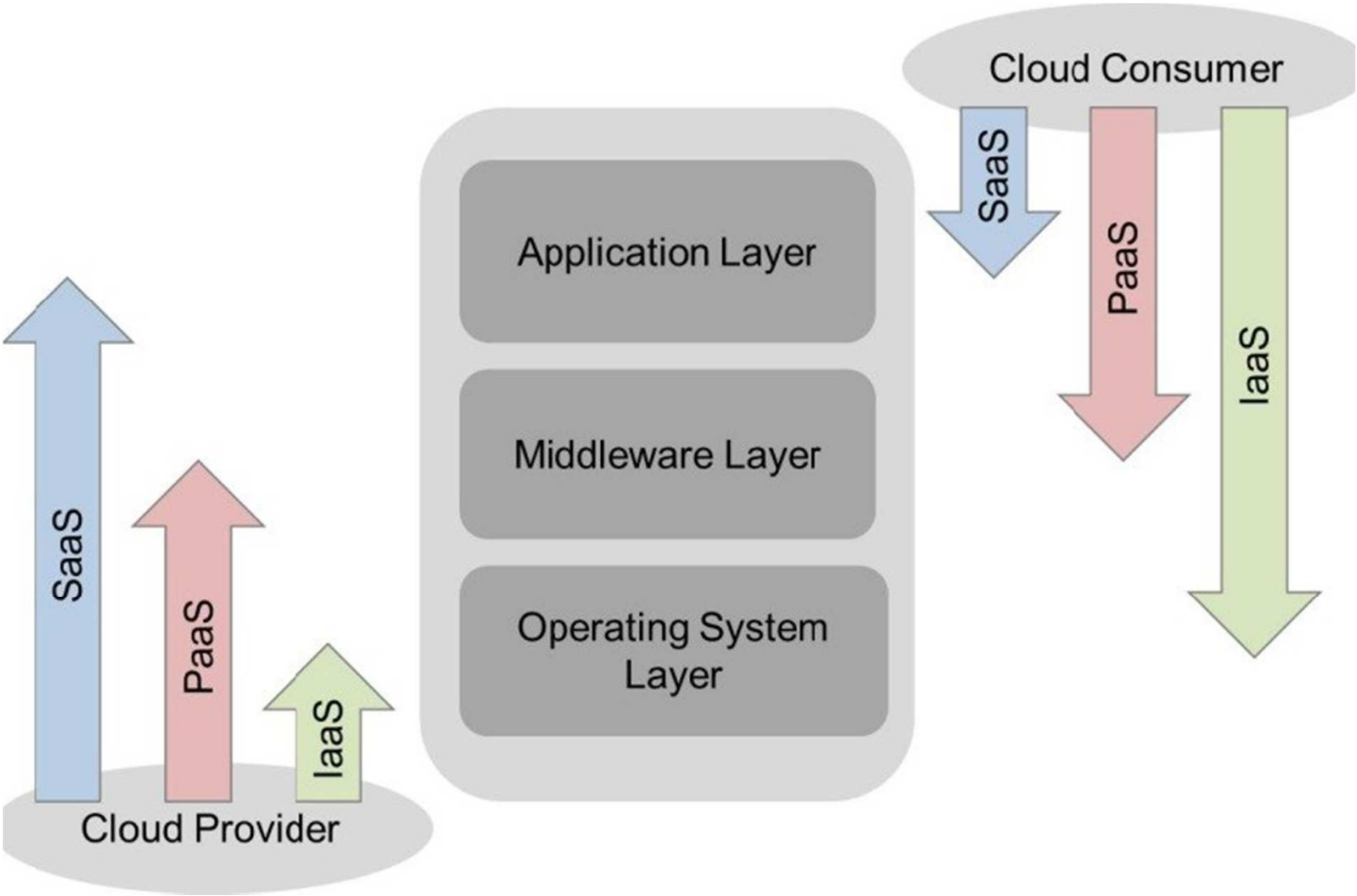
NIST Cloud Service Models

- Cloud Software as a Service (SaaS)
  consumer uses the provider's applications running on a cloud infrastructure.
- Cloud Platform as a Service (PaaS)
  consumer deploys or develops applications created using programming languages and tools supported by the provider.
- Cloud Infrastructure as a Service (IaaS)
  consumer deploys/runs arbitrary software on processing, storage, networks, and other managed computing resources

IaaS vs. PaaS vs. SaaS



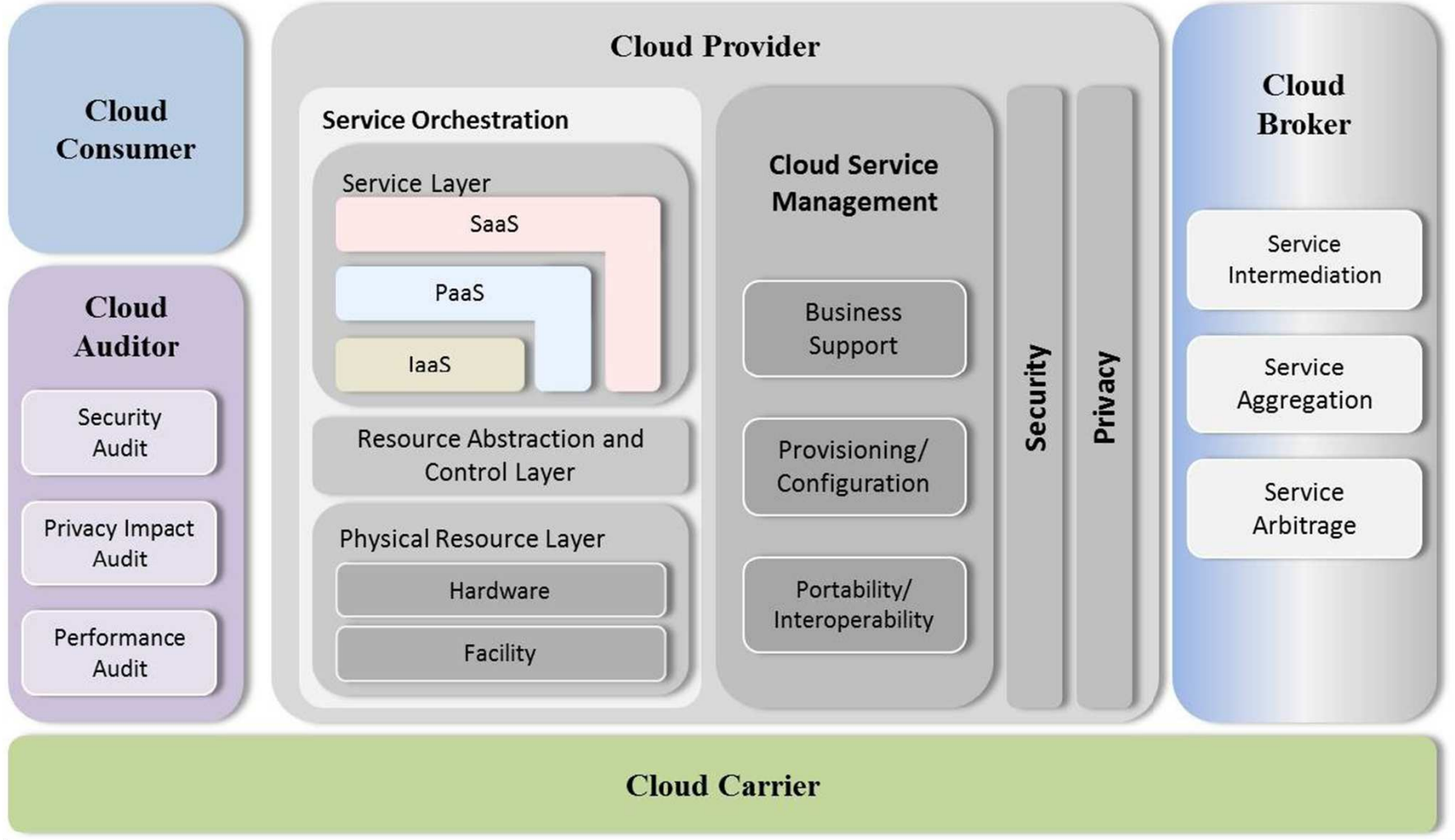
Consumer vs. provider concerns



NIST Deployment Models

- Private cloud: operated solely for an organization (managed by the organization or a third party, on-premise or off-premise)
- Community cloud: shared by several organizations, for a specific community with shared concerns
- Public cloud: generally available from seller
- Hybrid cloud: composition of clouds (private, community, or public) so to enable data and application portability (e.g., cloud bursting for load balancing between clouds).

NIST Reference Architecture



Main points

- Using clouds you can
  - Get the IT system/software/applications you need - as a service,
  - when you need them
  - for as long as you need them and
  - pay only (very little!) for what you use
- Cloud consumers can build IT systems using cloud components and provide end-users with services
  - Ability to customize components is desirable

What is Software-Defined Networking?

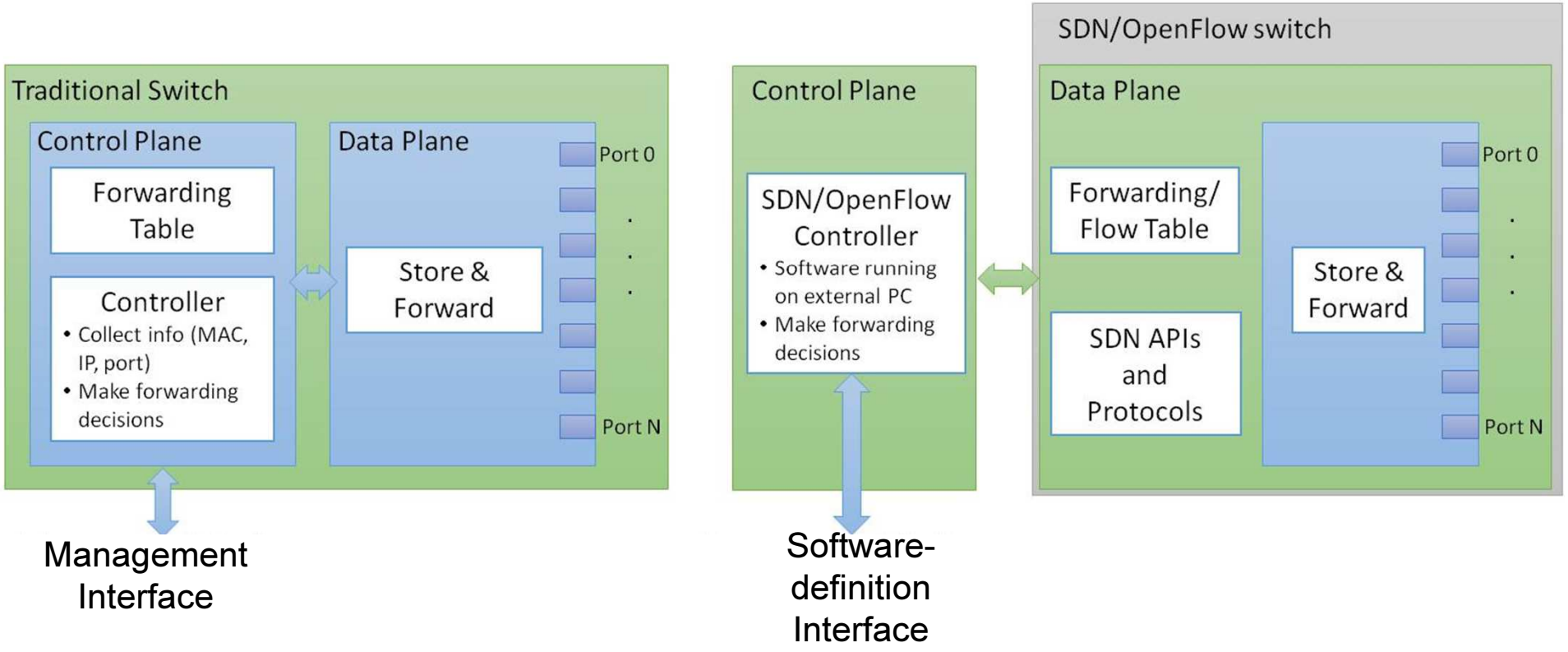
- Broad Definition
  - Open Network Foundation: “an architecture that enables direct programmability of networks”
  - Internet Engineering Task Force: “an approach that enables applications to converse with and manipulate the control software of network devices and resources” – *Internet Draft, Sep. 2011 by T. Nadeau*
- OpenFlow
  - An approach to SDN with physical separation between control and data planes
  - Provides open interfaces (APIs)
  - SDN is not OpenFlow but OpenFlow is a step towards SDN

Original need for Openflow

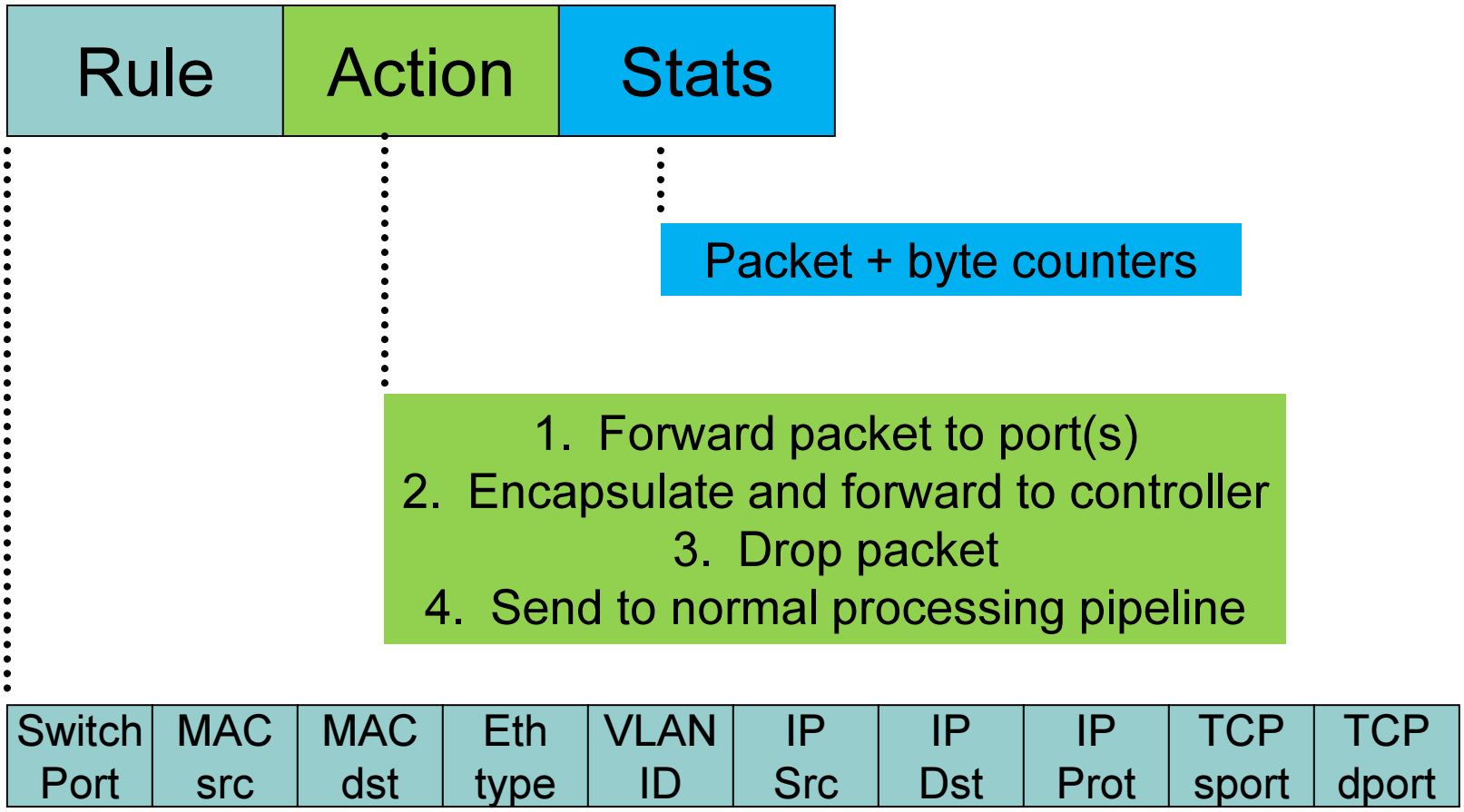
- Network infrastructure “ossification”
  - Large base of devices and protocols
  - Networking experiments cannot compete with production traffic
  - No practical way to test new network protocols in realistic settings
- Closed systems
  - Vendor lock-in
  - Proprietary management interfaces – lack of standard or open interfaces
  - Hard to establish collaborations
- “Today's networks are complex, functionally limited, and resistant to change; SDN changes all that”

OpenFlow Architecture

- Separate control plane and data plane
  - Run control plane software on general purpose hardware
  - Programmable data plane



OpenFlow Flow Table Entry



Source: Nick McKeown, "Why Can't I Innovate in My Wiring Closet?", MIT CSAIL Colloquium, April 2008

Examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

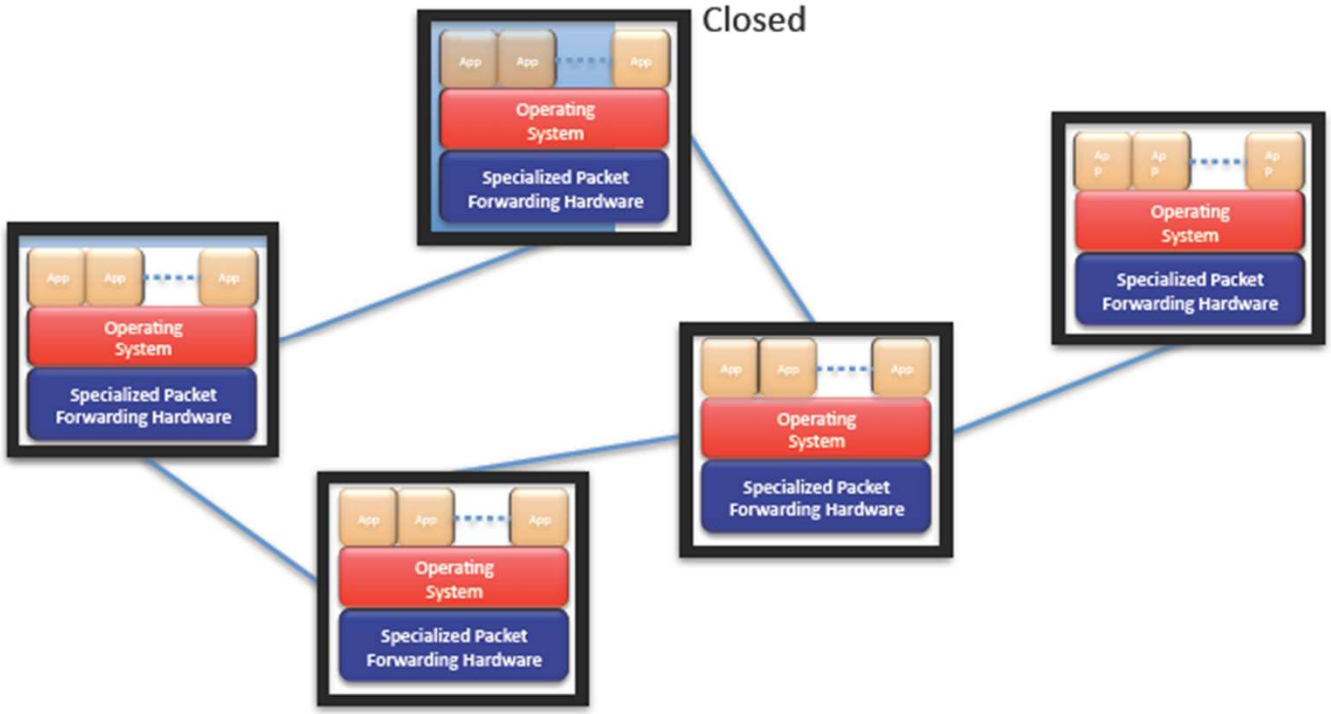
VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

From BrandonHeller's Tutorial at http://www.opennetsummit.org/archives-april2012.html

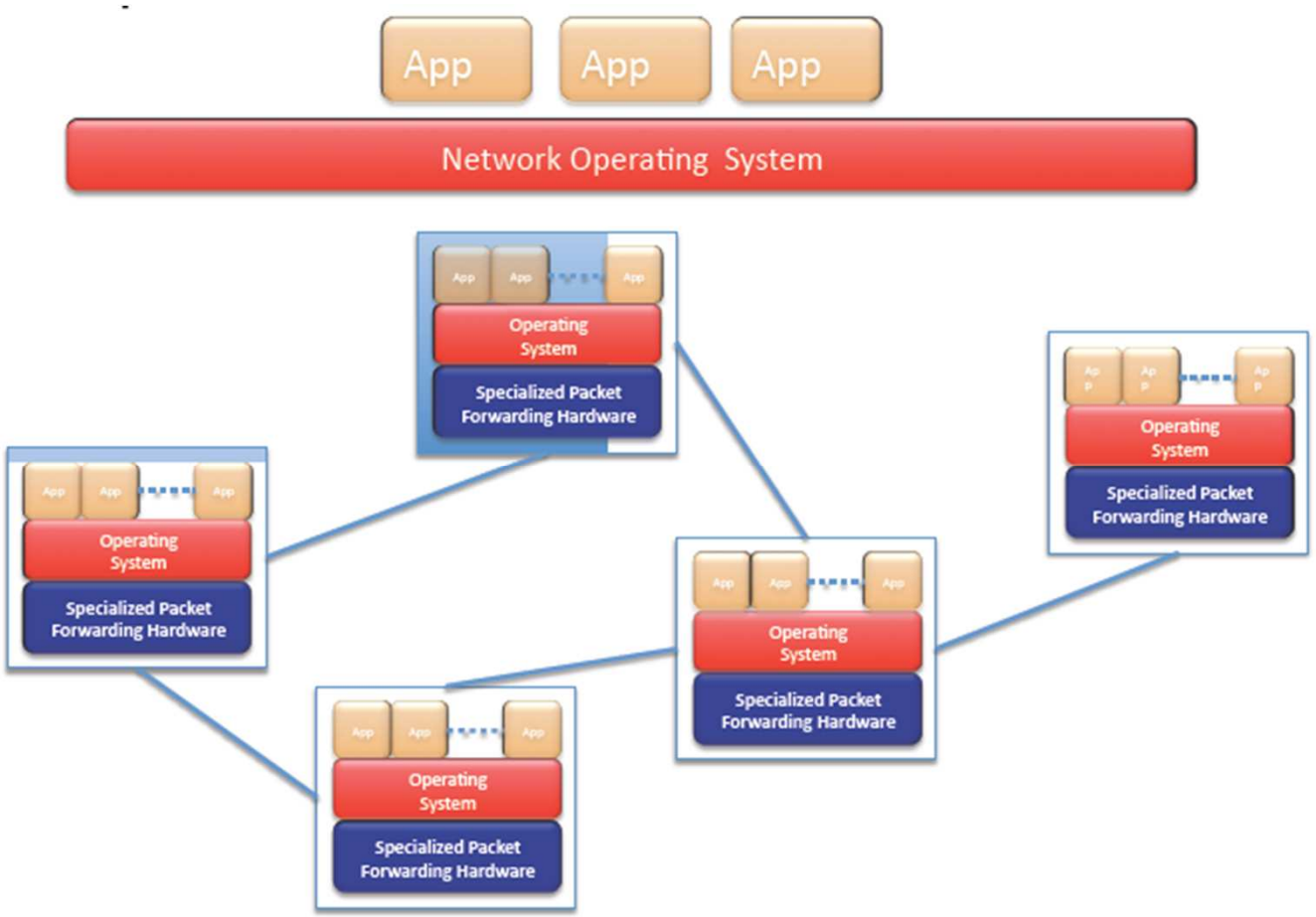
Nick McKeown's perspective

- Where is the functionality?
  - From closed boxes, distributed protocols

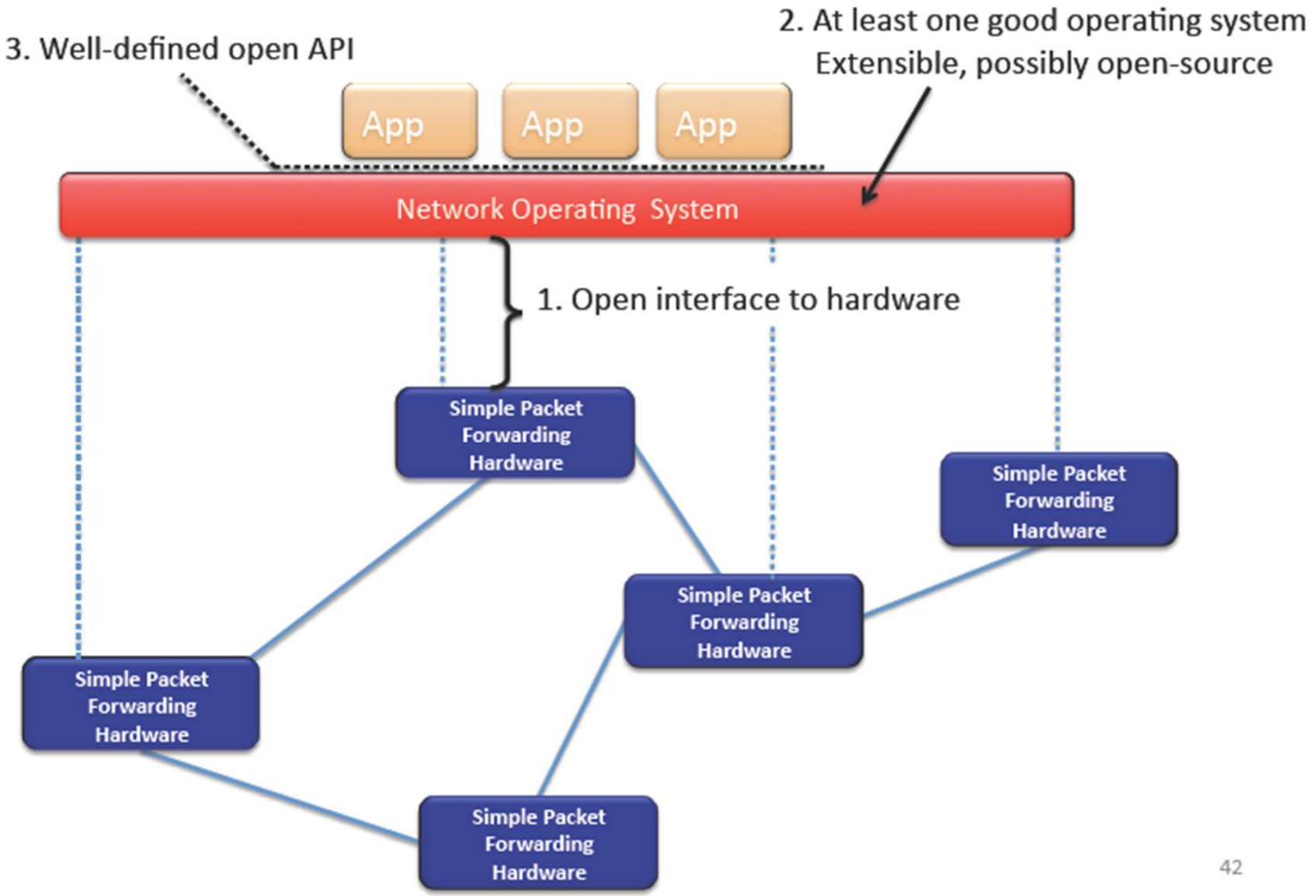


Software defined networking

- Where is the functionality?
  - From closed boxes, distributed protocols to open boxes



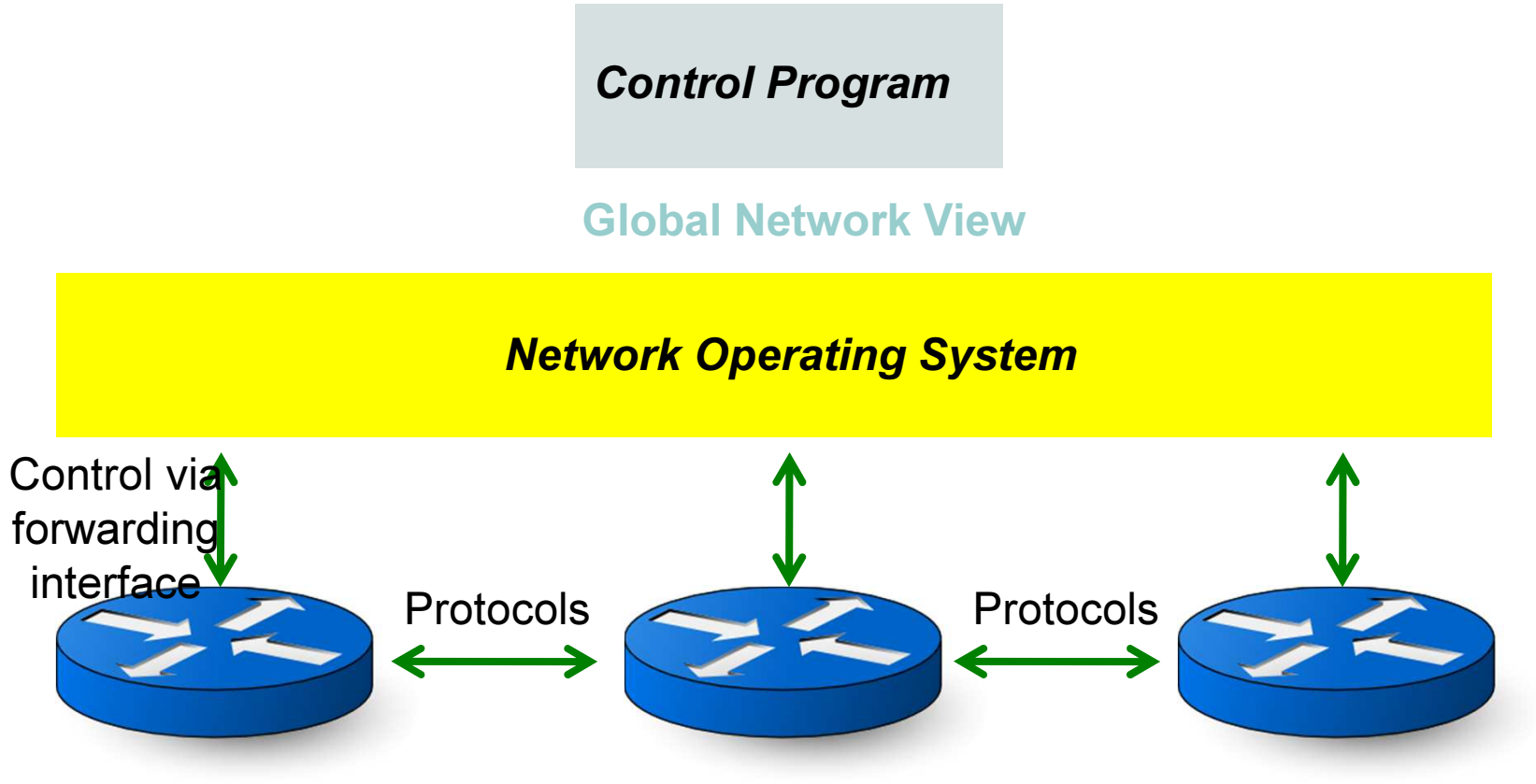
Software defined network



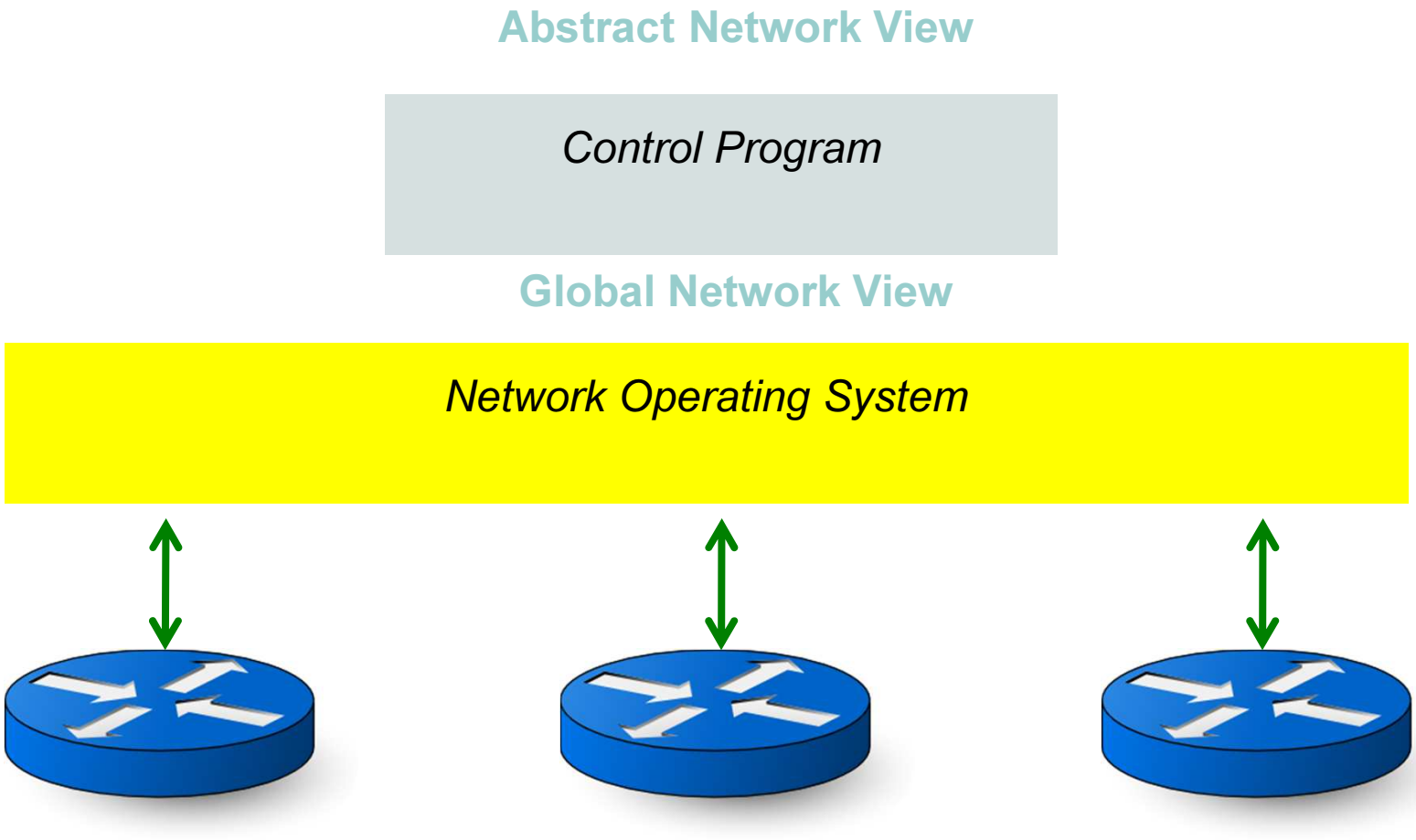
Scott Shenker's perspective

- SDN uses abstractions to program networks
  - Forwarding Abstraction
  - State Distribution Abstraction
  - Network Operating System (NOS)
- NOS plus Forwarding Abstraction = SDN v1
- Add a specification abstraction (SDN v2)
  - Implemented through "Nypervisor"
- Next two slides from Scott Shenker's talk "The Future of Networking, and the Past of Protocols"

Software-Defined Networking (v1)



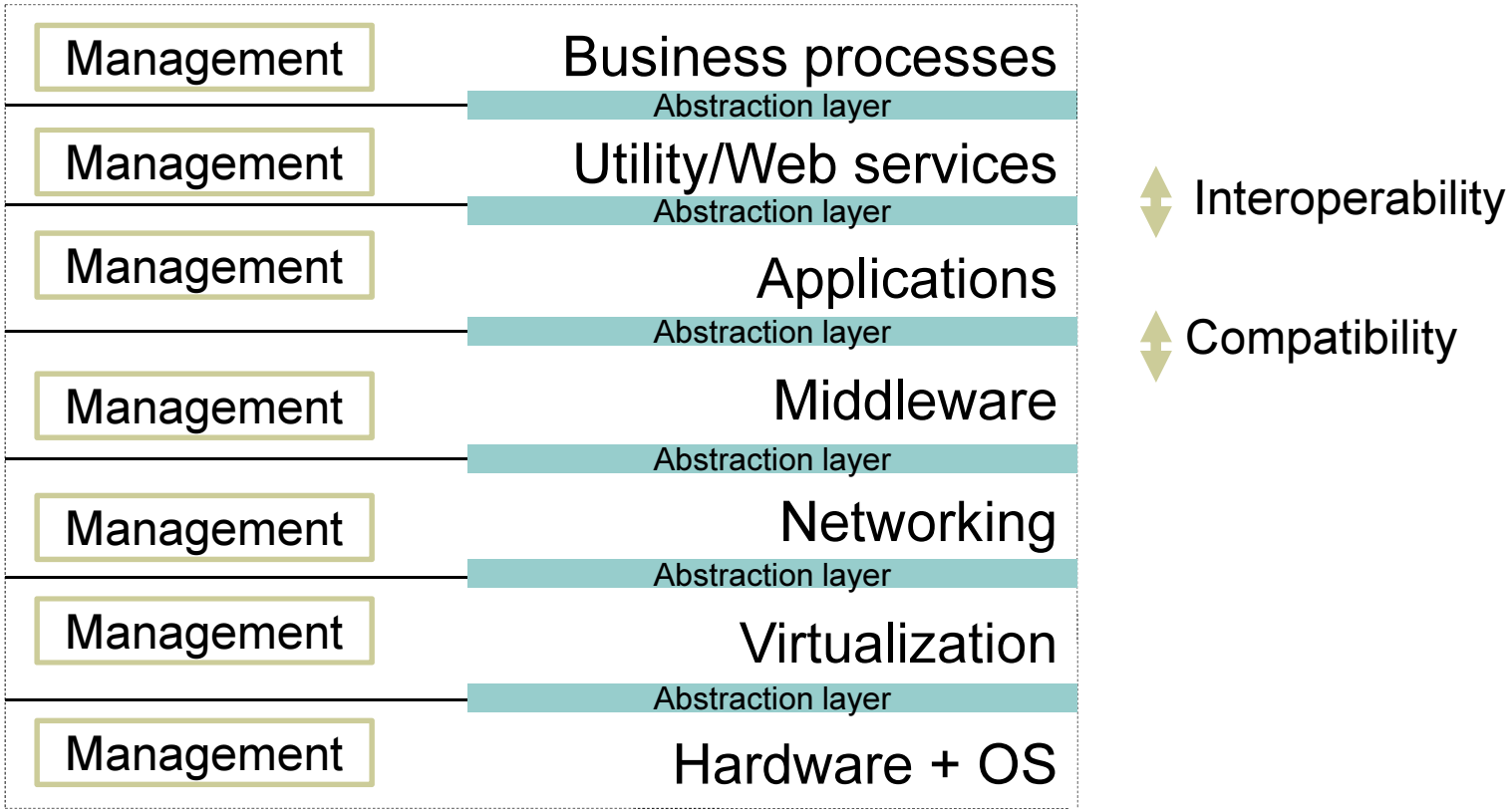
Moving from SDNv1 to SDNv2



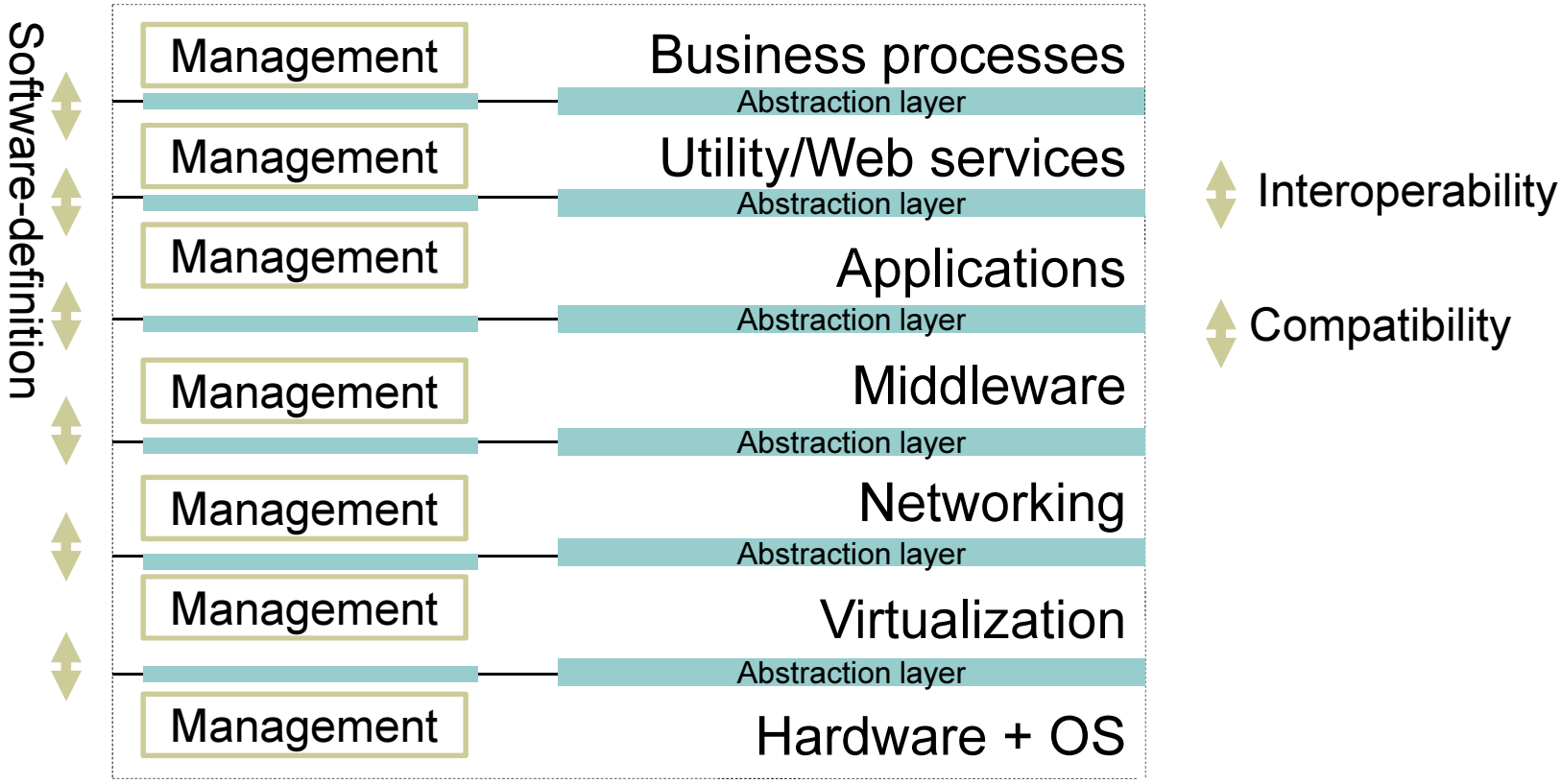
SDN and Cloud Computing

- Cloud Computing
  - Dynamic environment: resources (physical and virtual), users, and applications frequently come and go
  - Large scale infrastructure
  - Need efficient mechanisms to change how networks operate
- Without SDN
  - Rely on vendor-provided and in-house software to manage the network
  - Manually generated or semi-automatically generated configurations
  - Only cloud/network administrators can interact with network equipment
- With SDN
  - Network “programming” instead of network configuration
  - Potentially open to users/system-developers/applications

Current systems stacks

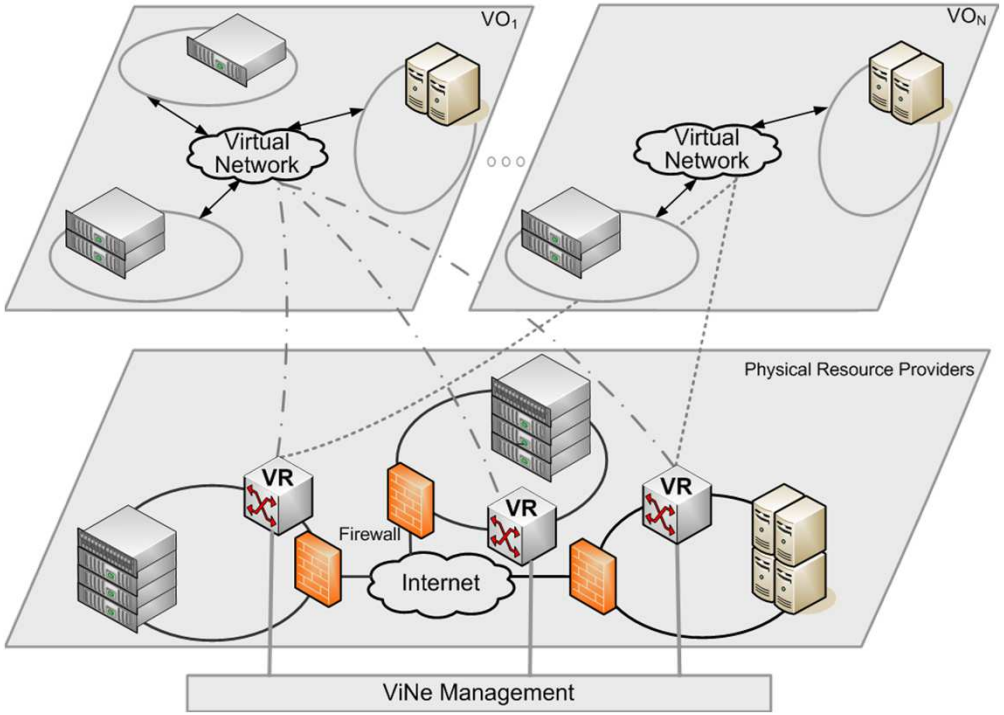


Software-defined systems stacks



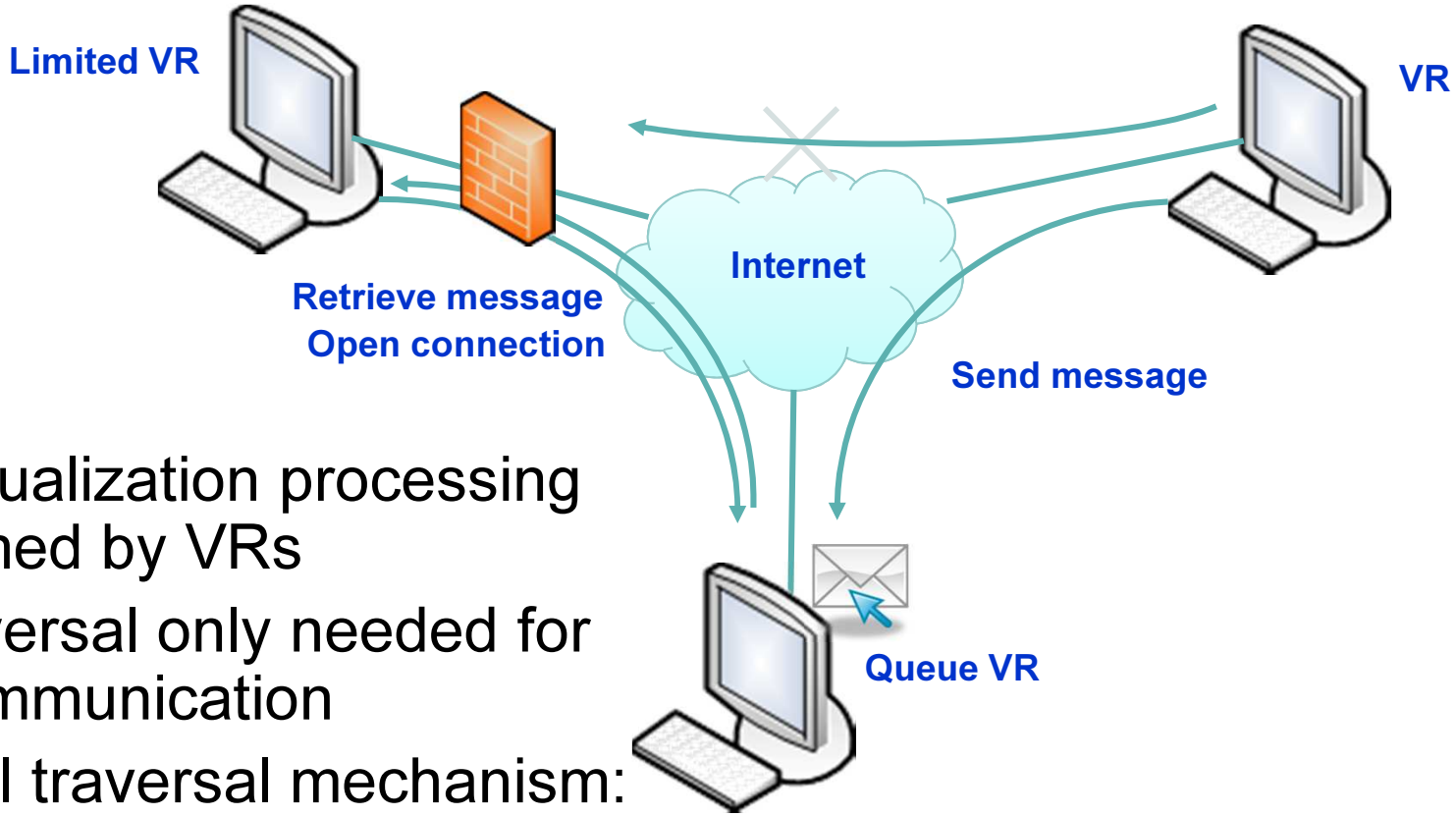
Virtual Networking (VN) with ViNe

- Dedicated resources in each broadcast domain (LAN) for VN processing –ViNe Routers (VRs)
  - No VN software needed on nodes (platform independence)
  - VNs can be managed by controlling/reconfiguring VRs
  - VRs transparently address connectivity problems for nodes



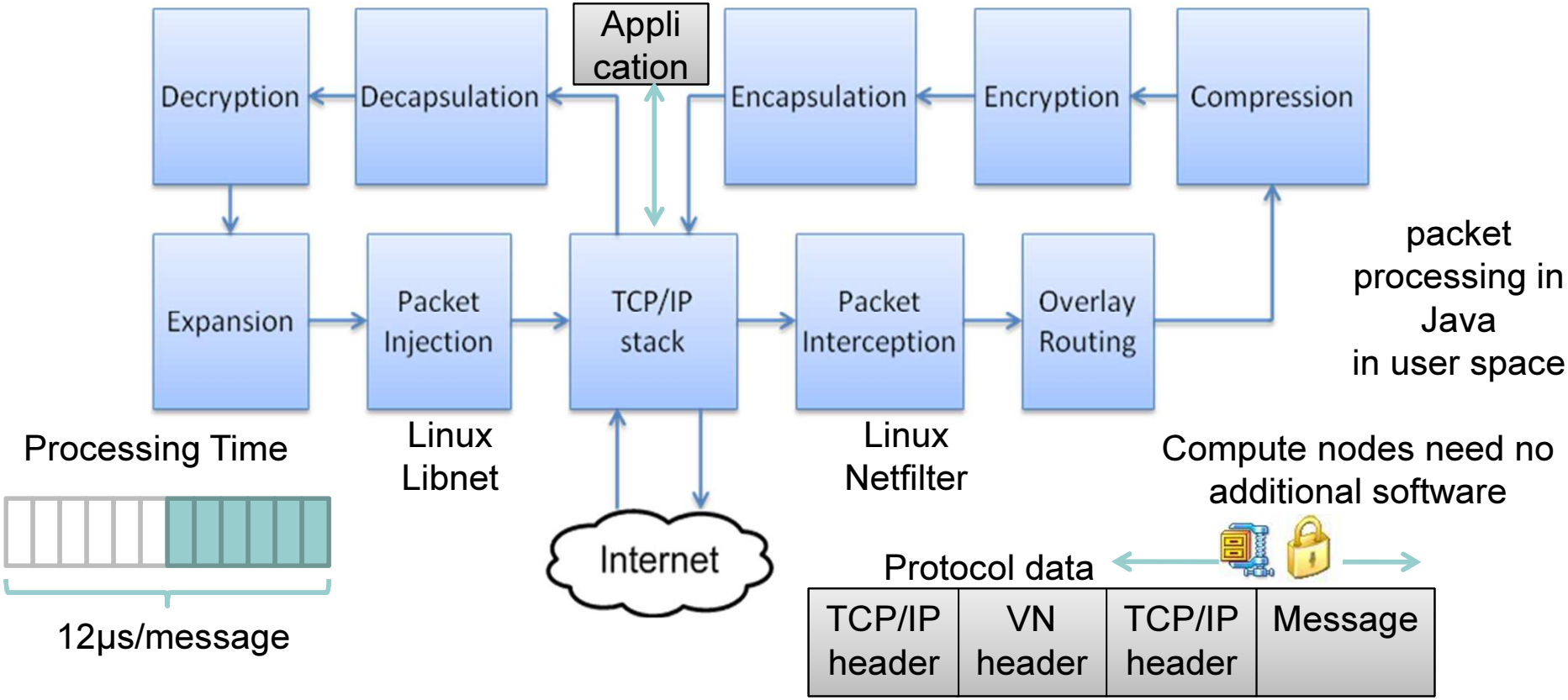
- VR = computer running ViNe software
  - Easy deployment
  - Proven mechanisms can be incorporated in physical routers and firewalls.
  - In OpenFlow-enabled networks, flows can be directed to VRs for L3 processing

Connectivity Recovery in ViNe



- Network virtualization processing only performed by VRs
- Firewall traversal only needed for inter-VR communication
- ViNe firewall traversal mechanism:
  - VRs with connectivity limitations (limited-VRs) initiate connection (TCP or UDP) with VRs without limitations (queue-VRs)
  - Messages destined to limited-VRs are sent to corresponding queue-VRs
  - Long-lived connection possible between limited-VR and queue-VR
  - Generally applicable

ViNe Routing



- Local Network Description Table (LNDT)
  - Describes the VN membership of a node
- Global Network Description Table (GNDT)
  - Describes sub-networks for which a VR is responsible

ViNe Routing

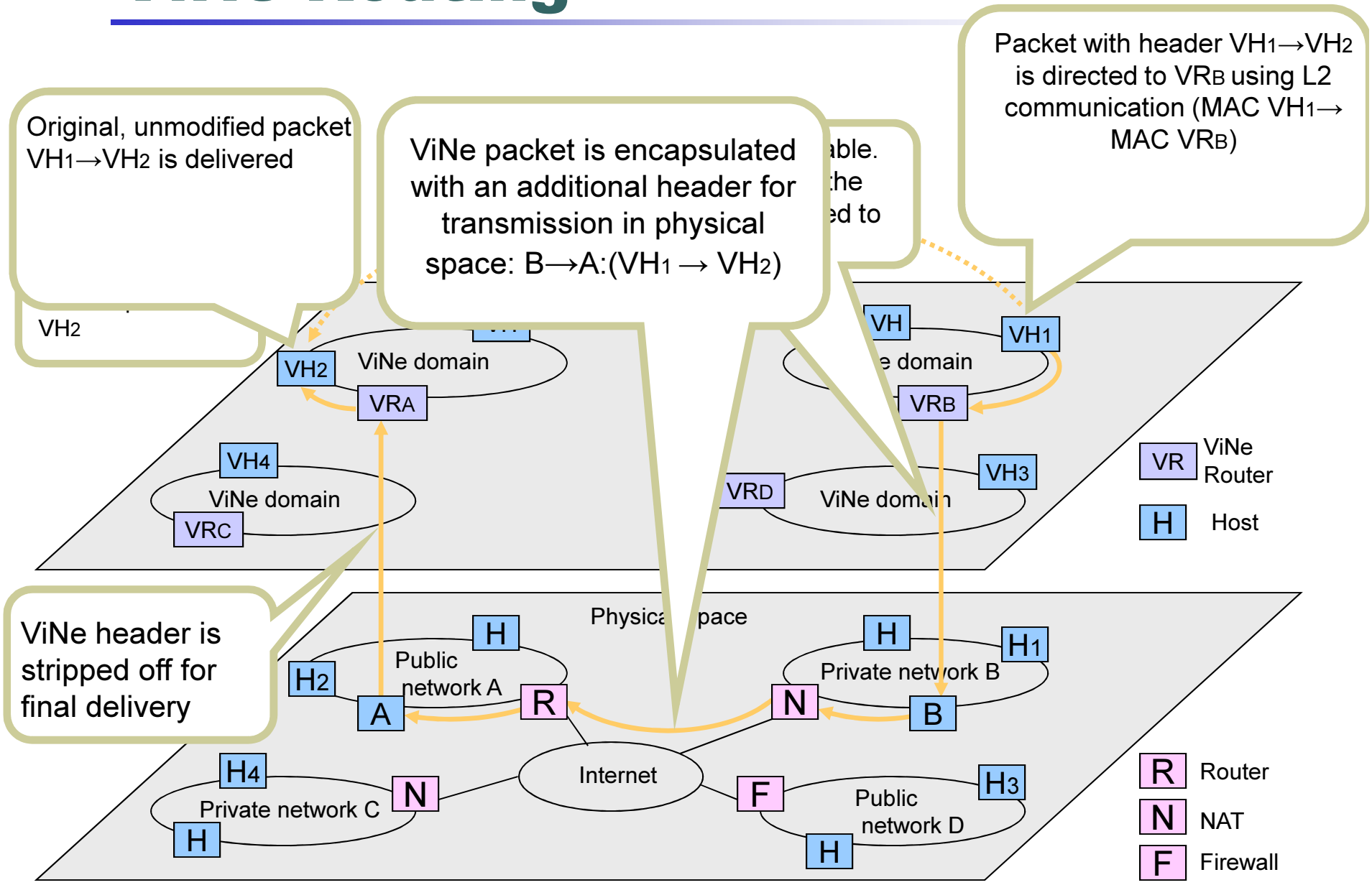
- Local Network Description Table (LNDT)
  - Describes the VN membership of a node
- Global Network Description Table (GNDT)
  - Describes the sub-networks that a VR is responsible for
- Suppose that a VR with the following routing tables, received a packet from 172.16.0.10 destined to 172.16.10.90

LNDT	
Host	ViNe ID
172.16.0.10	1
172.16.0.11	2

GNDT – ViNe ID 1	
Network/Mask	Destination
172.16.0.0/24	VR-a
172.16.10.0/24	VR-b

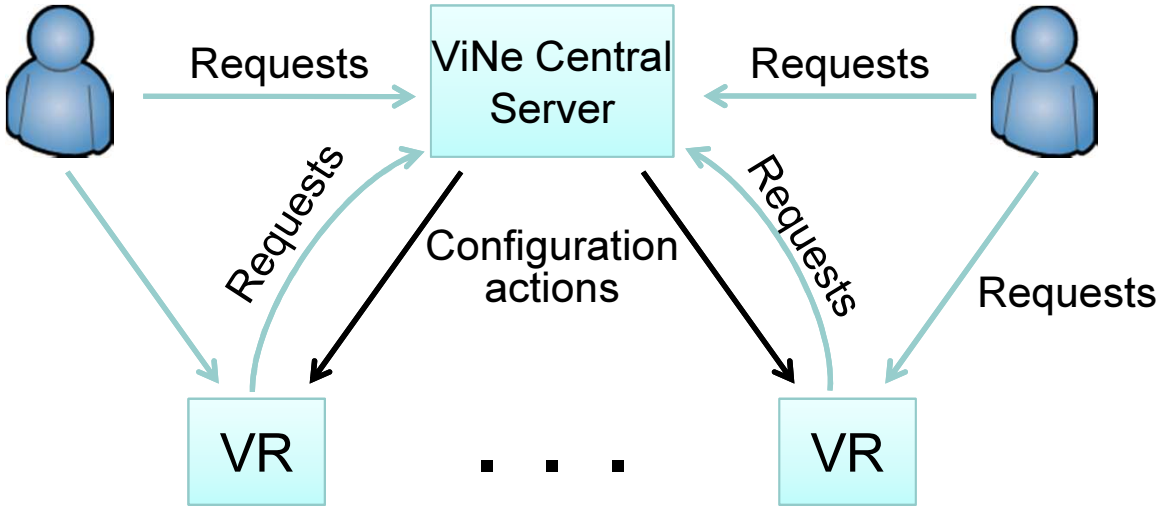
GNDT – ViNe ID 2	
Network/Mask	Destination
172.16.0.0/24	VR-a
172.16.20.0/24	VR-c

ViNe Routing



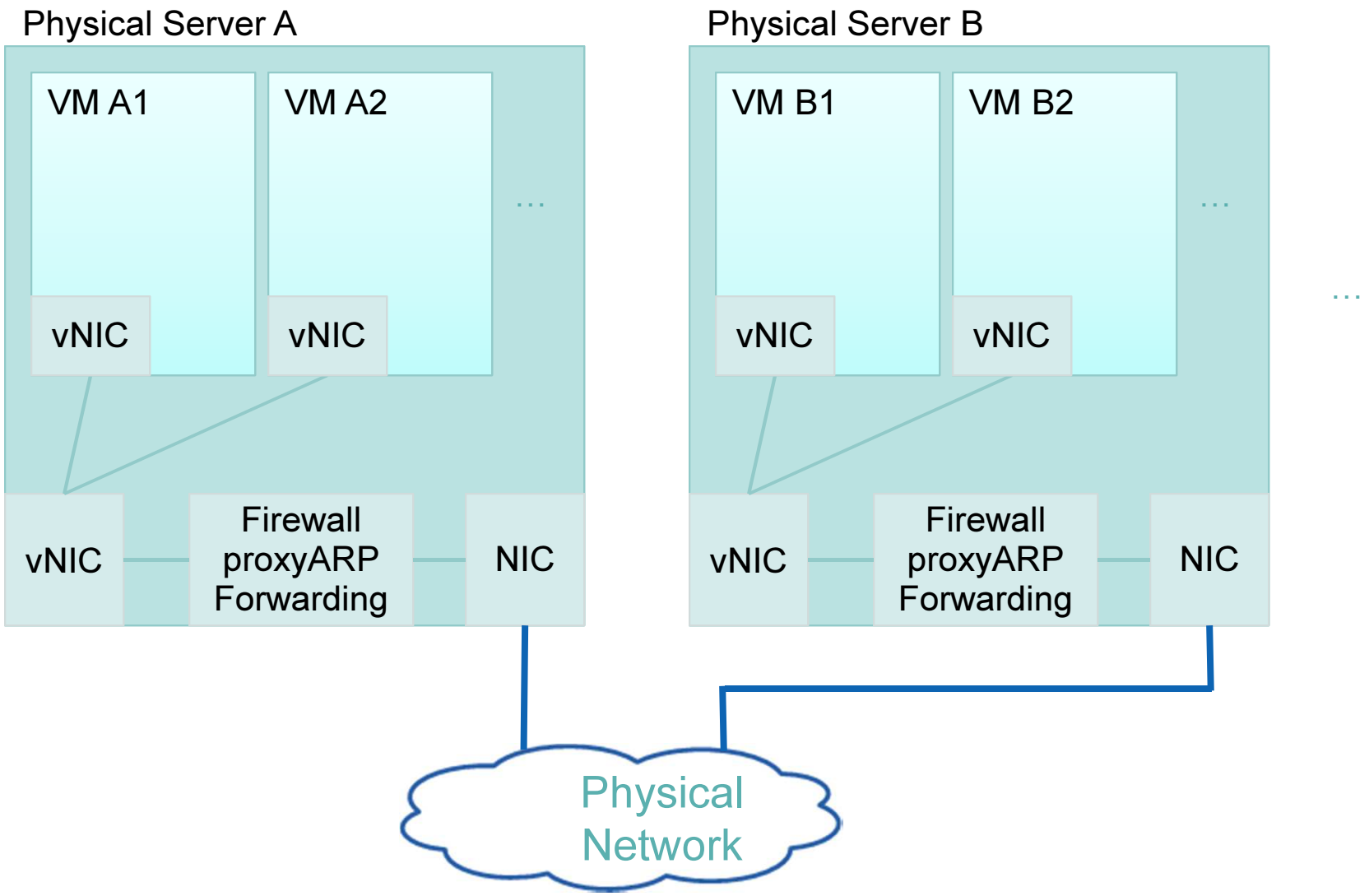
ViNe Management Architecture

- VR operating parameters changeable at run-time
  - Overlay routing tables, buffer size, encryption on/off
  - Autonomic approaches possible
  - Java reflection to map commands to method invocations



- ViNe Central Server
  - Oversees global VN management
  - Maintains ViNe-related information
  - Authentication/authorization based on Public Key Infrastructure
  - Remotely issue commands to reconfigure VR operation

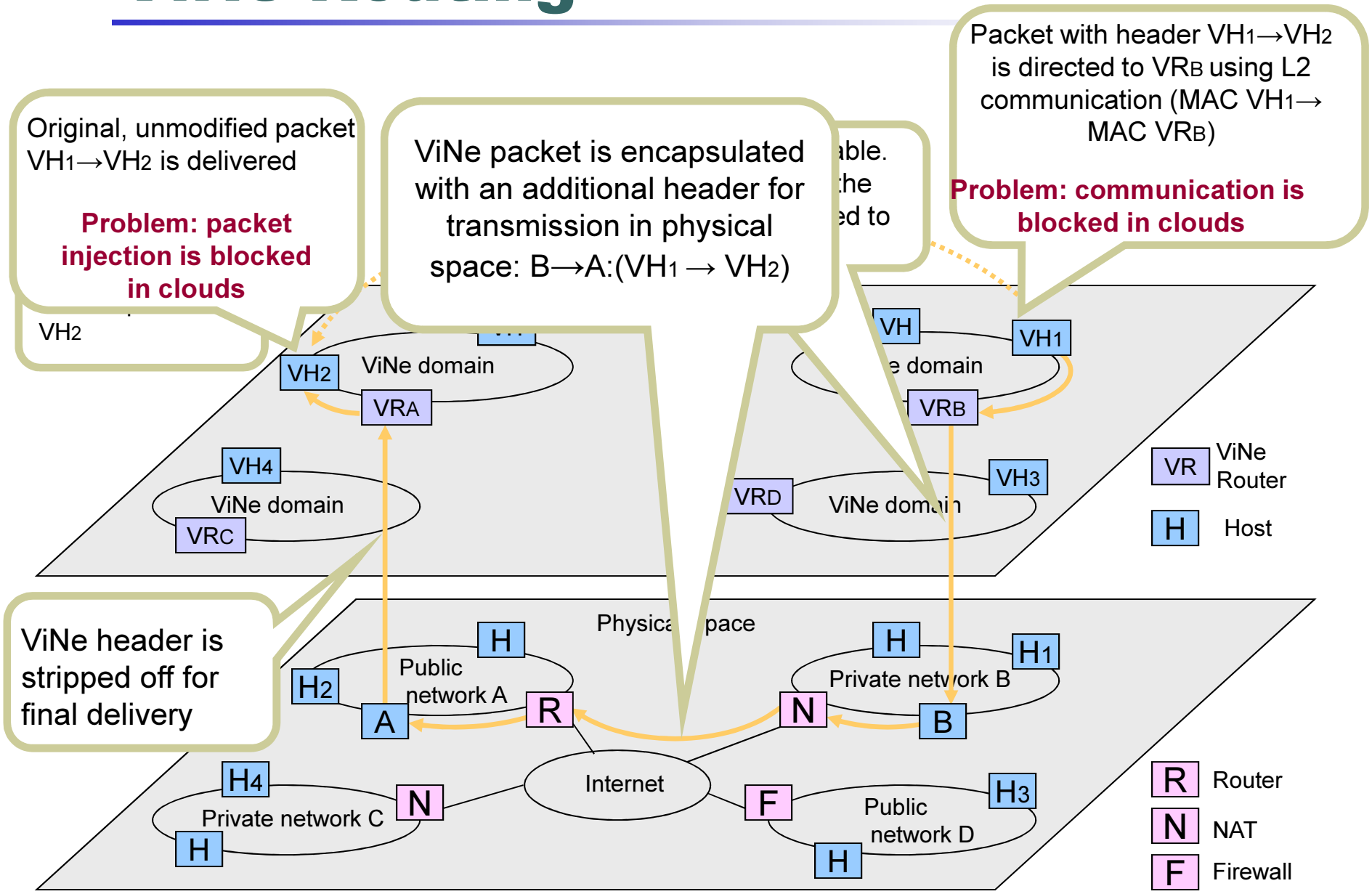
Typical IaaS Cloud Network



Network Restrictions in Clouds

- To address dangers of VM privileged users
  - change IP and/or MAC addresses, configure NIC in promiscuous mode, use raw sockets, attack network (spoofing, proxy ARP, flooding, …)
- Internal routing and NAT
  - granted IP addresses (especially public) are not directly configured inside VMs, and NAT techniques are used (many intermediate nodes/hops in LAN communication)
- Sandboxing (disables L2 communication)
  - VMs are connected to host-only networks
  - VM-to-VM communication is enabled by a combination of NAT, routing and firewalling mechanisms
- Packet filtering (beyond usual, VM can not be VR)
  - only those VM packets containing valid addresses (IP and MAC assigned by the provider) are allowed

ViNe Routing



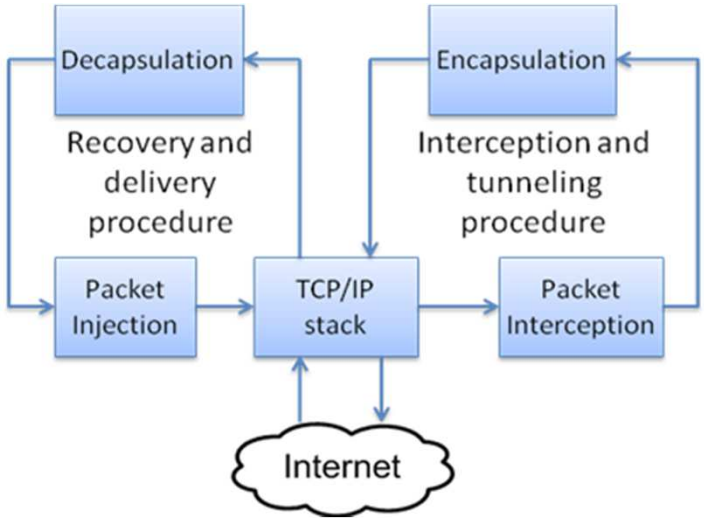
Solution

- Configure all nodes to work as VRs
  - No need for host-to-VR L2 communication
  - TCP or UDP based VR-to-VR communication circumvents the source address check restriction
- But…
  - Network virtualization software required in all nodes
  - Network virtualization overhead in inter- and intra-site communication
  - Complex configuration and operation
- TinyViNe
  - No need to implement complex network processing – leave it to specialized resources (i.e., full-VRs)
  - Keep it simple, lightweight, tiny
  - Use IP addresses as assigned by providers
  - Make it easy for end users to deploy

TinyViNe

- TinyViNe software
  - Enables host-to-VR communication on clouds using UDP tunnels
  - TinyVR – nodes running TinyViNe software
- TinyVR processing

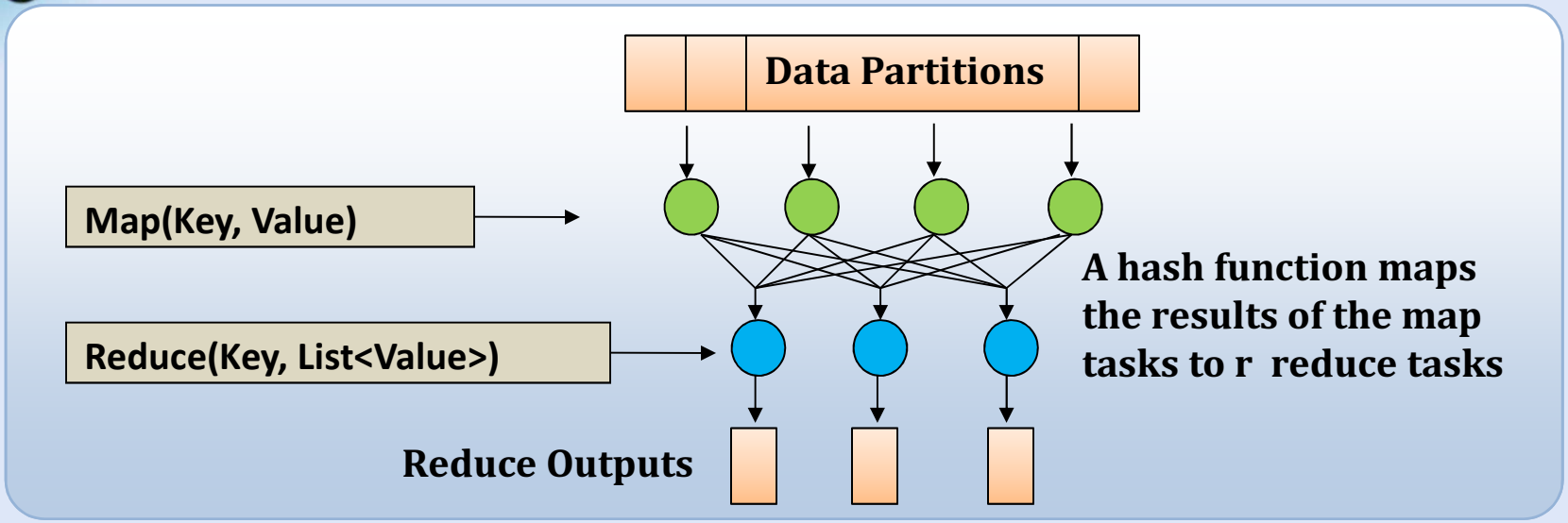
- Intercept packets destined to full-VRs
- Transmit to a VR the intercepted packets through UDP tunnels
- Decapsulate incoming messages through UDP tunnels
- Deliver the packets



What is MapReduce?

- programming model and associated implementation for processing/generating large data sets.
- Hadoop = open-source implementation of MapReduce
- users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key
- beginner's example: how many instances of a given word exist an a set of files?
  - <File1> <to be good> <File2> <to be or not to be>

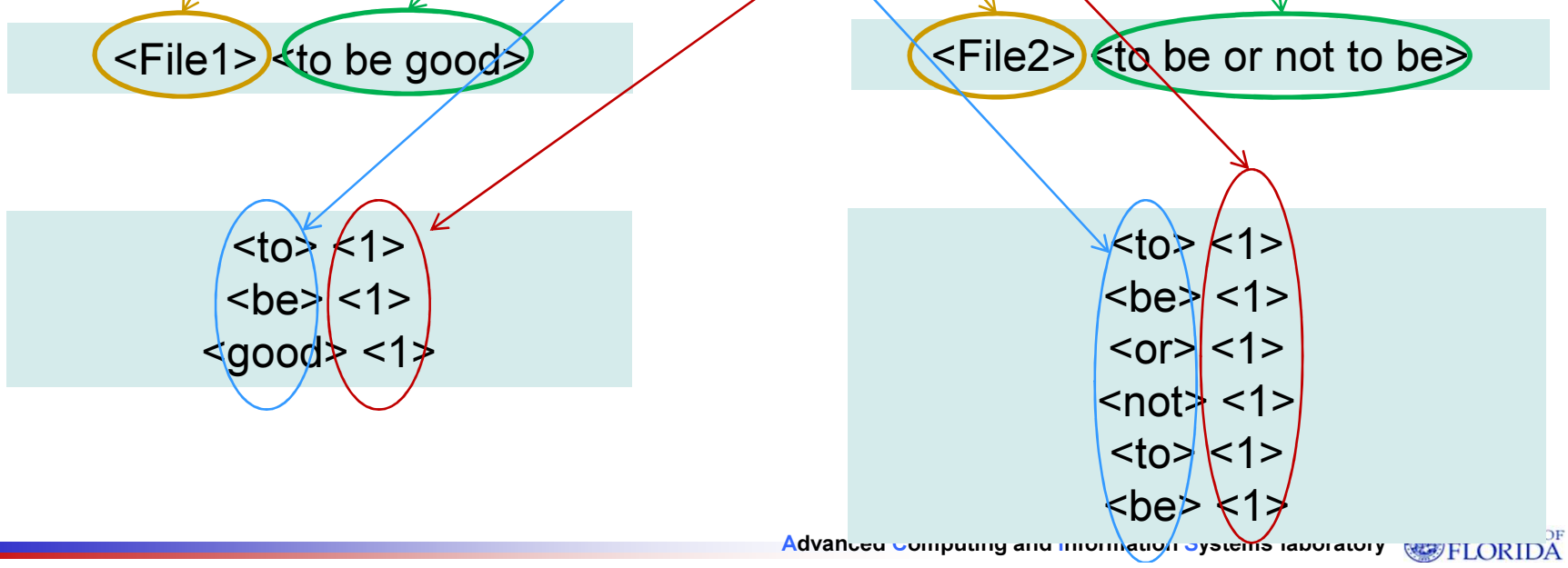
MapReduce



- Implementations support:
  - Splitting of data
  - Passing the output of map functions to reduce functions
  - Sorting the inputs to the reduce function based on the intermediate keys
  - Quality of service
  - Fault-tolerance
  - …

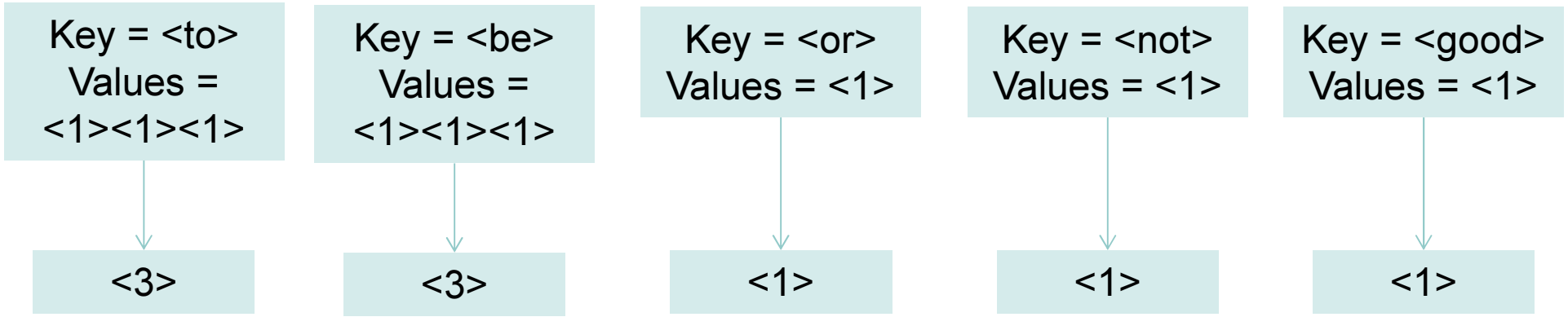
Map pseudo-code segment

```
map(String input_key, String input_value):
   // input_key: document name
   // input_value: document contents
   for each word w in input_value:
      EmitIntermediate(w, "1");
```



Reduce pseudo-code segment

```
reduce(String output_key, Iterator intermediate_values):
  // output_key: a word
  // output_values: a list of counts
  int result = 0;
  for each v in intermediate_values: result +=ParseInt(v);
 Emit(AsString(result));
```



Output ⟹

<to> <3>
<be> <3>
<or> <1>
<not> <1>
<good> <1>

Other Applications

- Distributed Grep
- Count of URL Access Frequency
- Reverse Web-Link Graph
- Term-Vector per Host
- Inverted Index
- Distributed Sort
- Web-indexing, ad analytics, data mining, bioinformatics, log analysis, recommendation systems, etc.

What happens under the hood

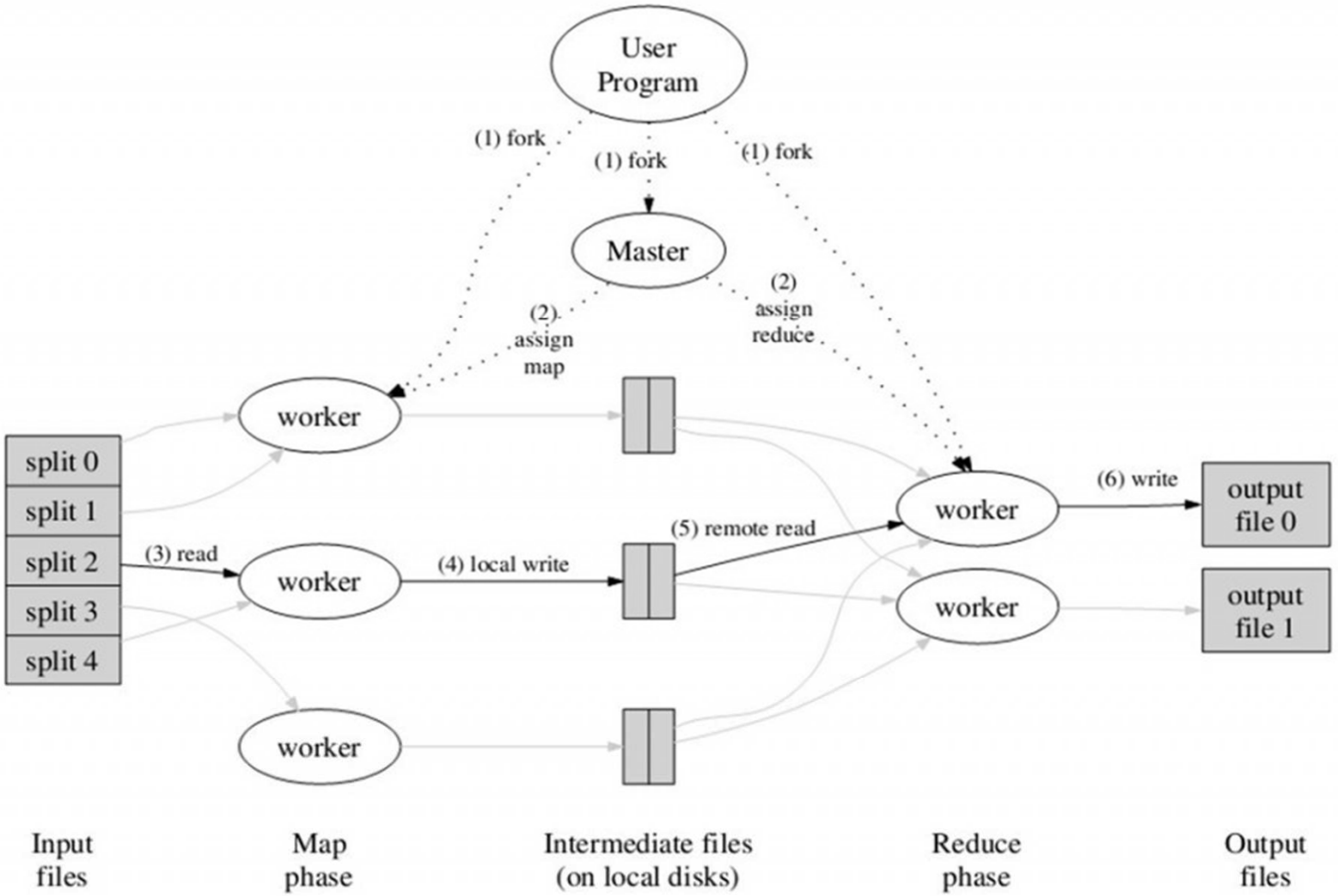


Figure 1: Execution overview

Locality

- Master program divides up tasks based on location of data: tries to have map() tasks on same machine as physical file data, or at least same rack
- map() task inputs are divided into 64 MB blocks: same size as Google File System chunks

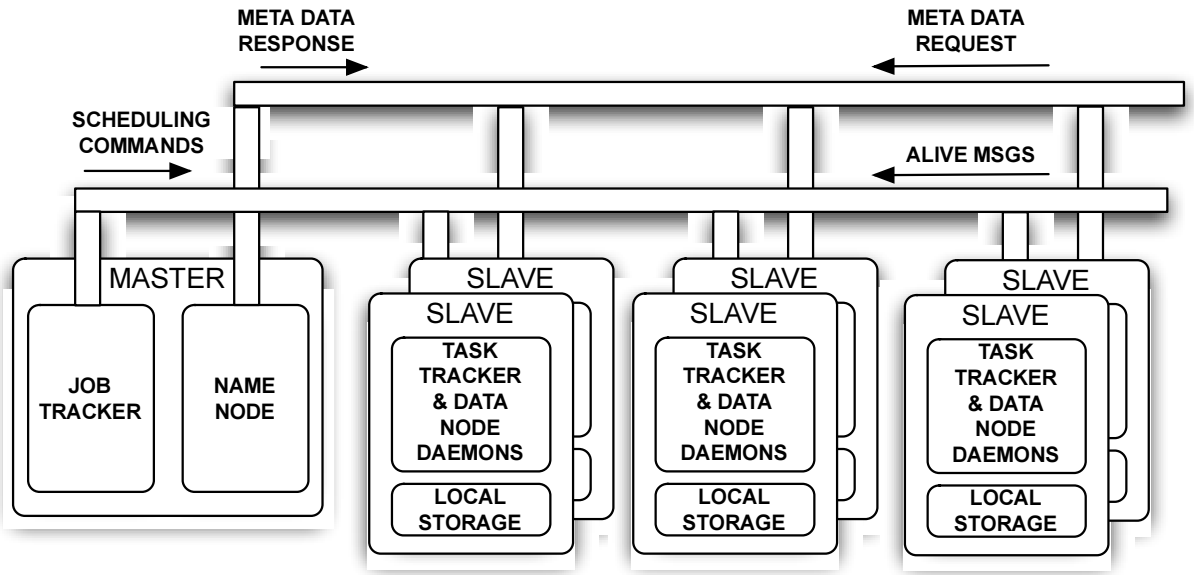
Fault Tolerance

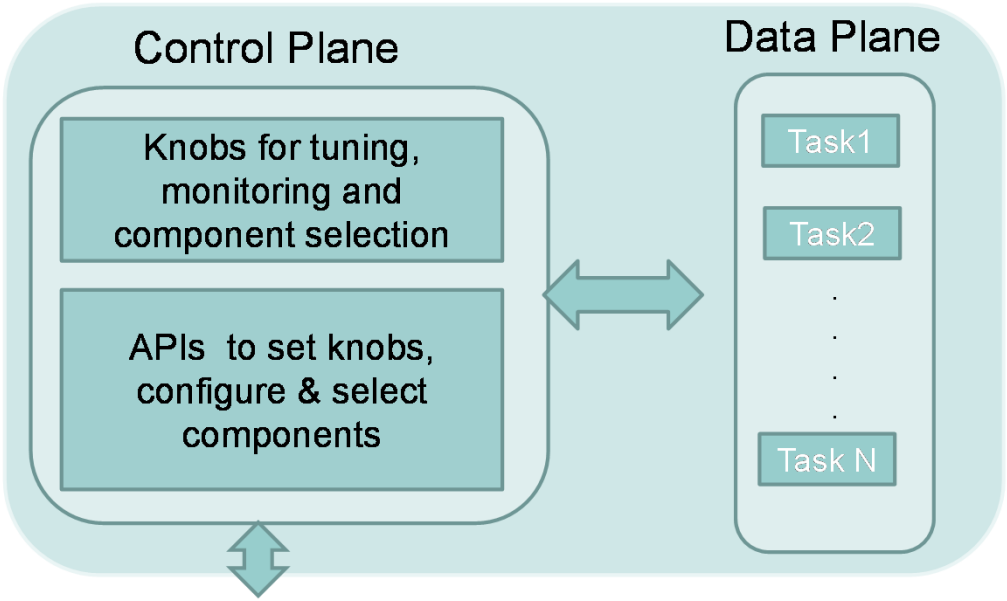
- Master detects worker failures
  - Re-executes completed & in-progress map() tasks
  - Re-executes in-progress reduce() tasks
- Master notices particular input key/values cause crashes in map(), and skips those values on re-execution.
  - Effect: Can work around bugs in third-party libraries!

Optimization

- No reduce can start until map is complete:
  - A single slow disk controller can rate-limit the whole process
  - Master redundantly executes “slow-moving” map tasks; uses results the copy that finishes first
- “Combiner” functions can run on same machine as a mapper
  - Causes a mini-reduce phase to occur before the real reduce phase, to save bandwidth

How it happens





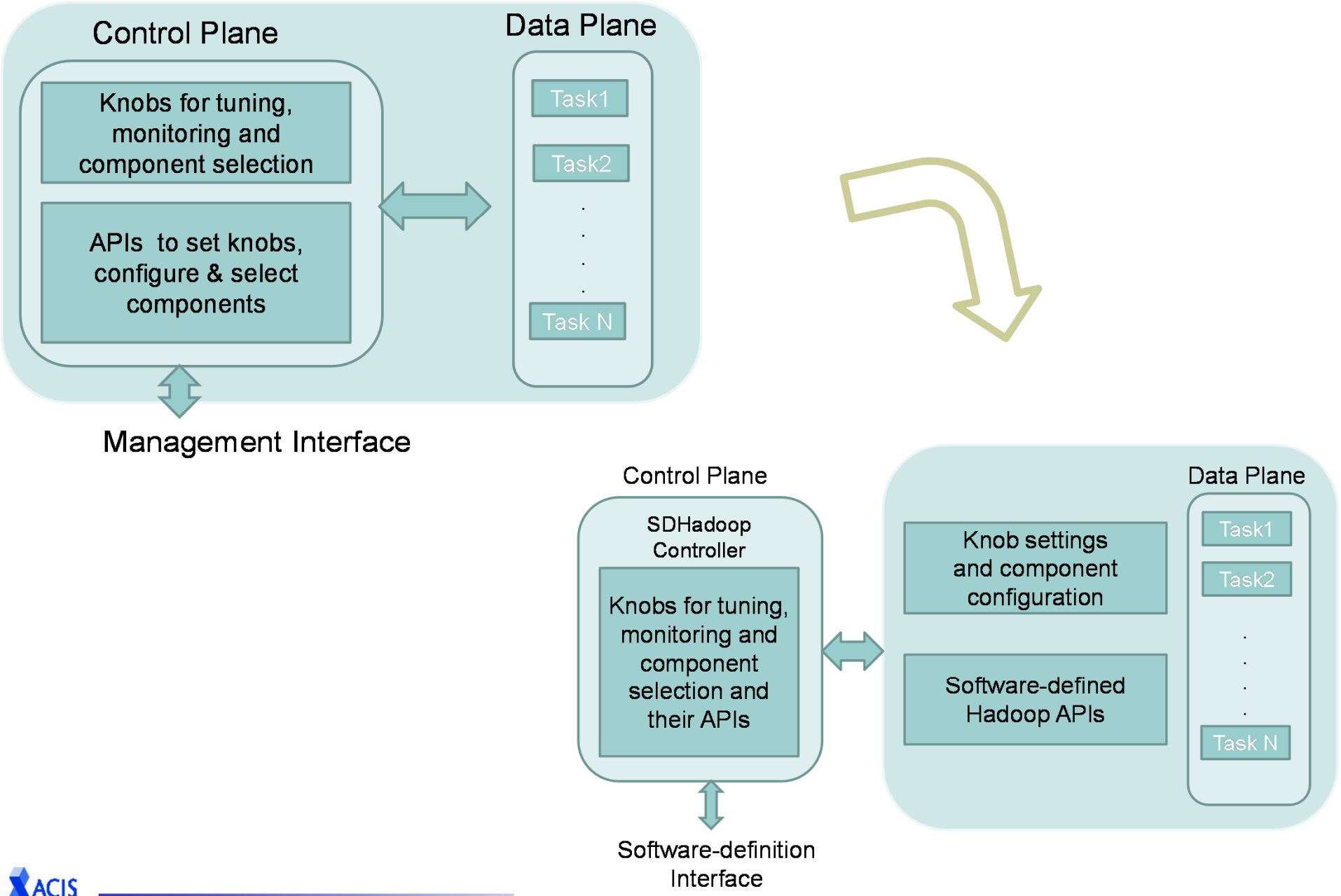
Many static parameters

- dfs.block.size
  - size of data blocks in which the input data set is split.
  - Larger: less map task creation overhead, may undertutilize cluster resources and lots of I/O.
  - Smaller: more overhead in map task creation
- io.sort.mb
  - size of in-memory buffer for map task to sort its output
  - Larger: sorting fast, fewer spills to disk
  - Too large: limits memory used by other MR threads
- io.sort.factor
  - max number of streams to merge at once when sorting files
  - Larger: fewer rounds of merging (faster) but more CPU
  - Smaller: require less CPU resources.

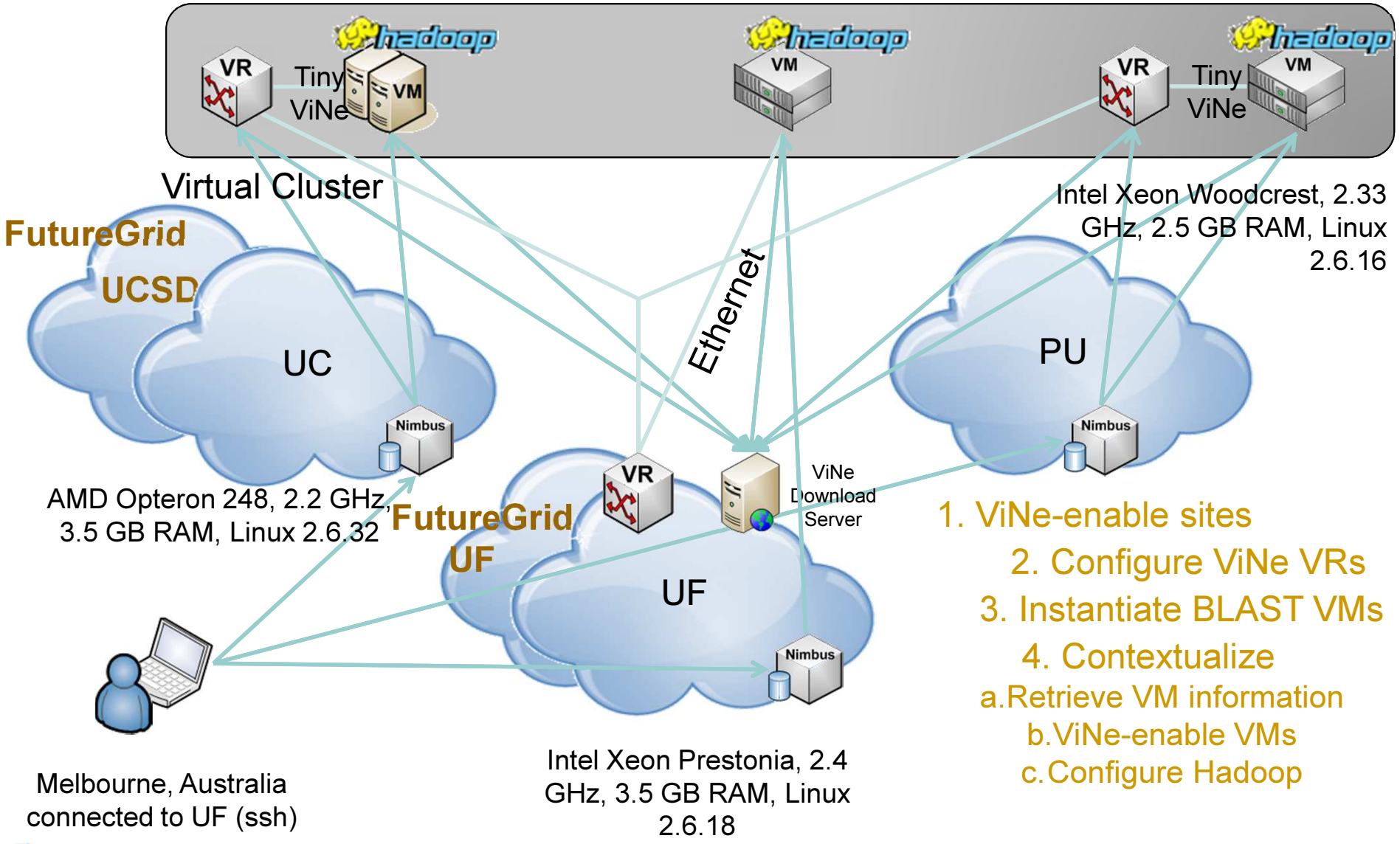
Many dynamic parameters

- in fair-scheduler.xml (reloaded every 3 seconds)
  - maxMaps, maxReduces: set max concurrent task slots
  - schedulingMode: fair sharing or fifo
  - maxRunningJobs: number of concurrent jobs
  - Weight: share relative to other pools
  - minSharePreemptionTimeout: seconds the pool will wait before killing other pools' tasks if it is below its minimum share
  - poolMaxJobsDefault: running job limit for pools
  - userMaxJobsDefault, running job limit for users
  - defaultMinSharePreemptionTimeout: fairSharePreemptionTimeout:
  - defaultPoolSchedulingMode:

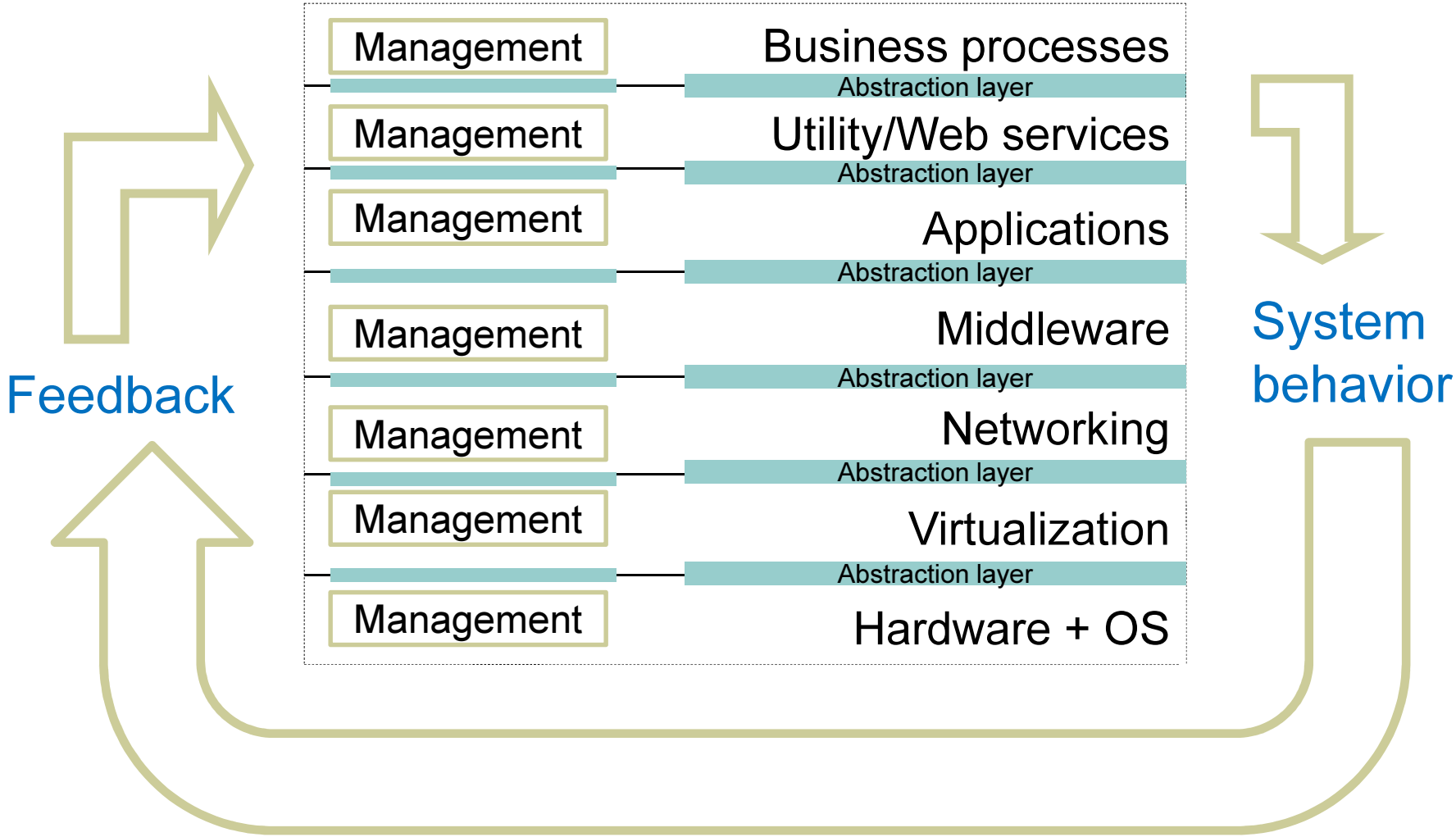
Software-defined MapReduce



Multicloud Hadoop-based BLAST



Autonomic SD systems



Faults & MapReduce

Built-in fault-tolerance in Hadoop

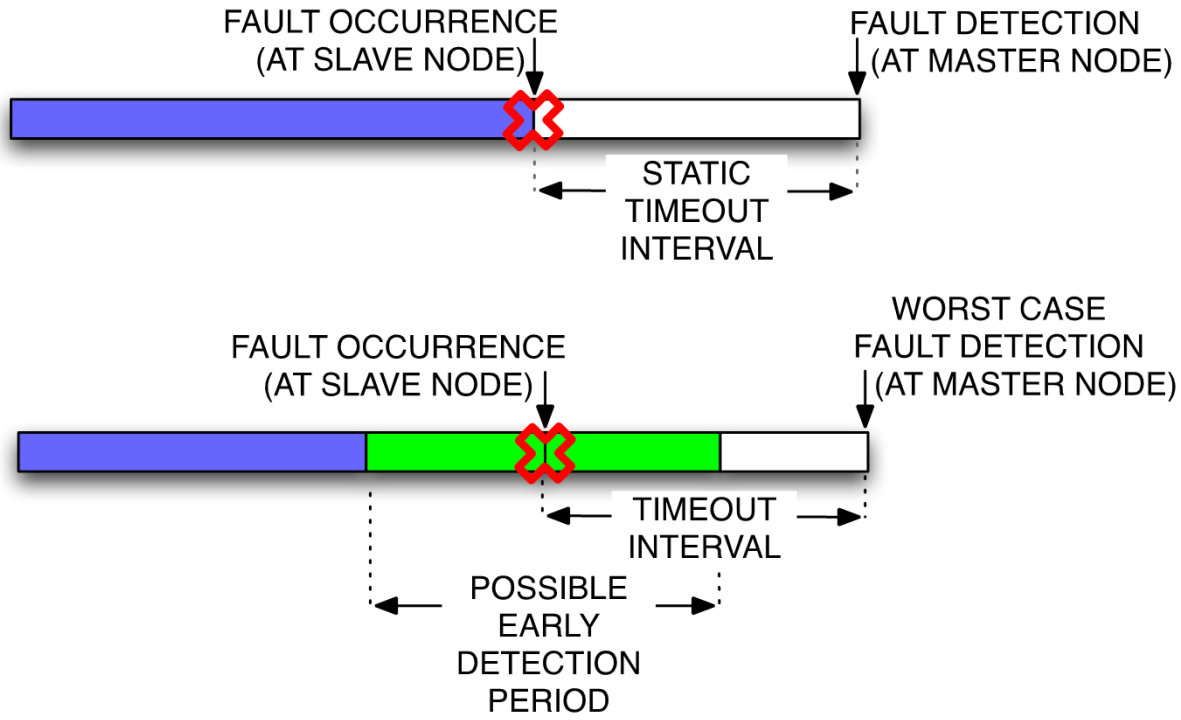
- Simple and based on task re-execution
- Job can run to completion
- Can result in performance penalties
  - Penalties of up to 139%, Node failure: [MASCOTS2009]
  - Penalties of up to 34%, Outliers, [OSDI2010]
  - Penalties of up to 350%, TaskTracker failure [HPDC2012]
  - “Usually allot 2 hours to a 1 hour job”: Yahoo, personal communication
  - depend on workload characteristics, faultoad (time, number & type of faults), resources (number of slave nodes in the job), configuration parameters (block size of data, timeout interval)

Goal: Improve fault management in MapReduce through an online, on-demand, closed-loop solution.

- Not re-implement MapReduce or change Hadoop

Early fault detection

- Time taken for fault detection



- After detection, recover through horizontal scaling

Anomaly detection

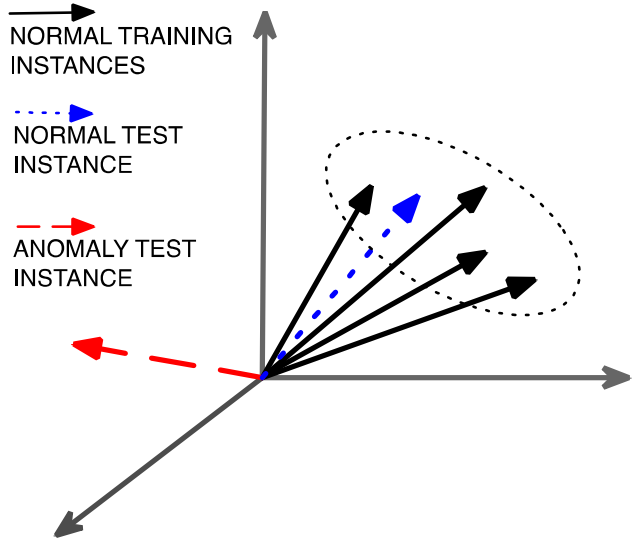
- Goal: Observe slave node operation and detect a performance anomaly as soon as possible

- Desired characteristics of anomaly detection technique
  - Good accuracy for both
    - Normal condition (True negative rate)
    - Anomalous condition (True positive rate)
  - Short testing/prediction computation time
  - Short training time
  - Scalable to a large number of nodes
  - Able to work with limited or no anomalous data

- Vast number of anomaly detection techniques, however with very specific applicability

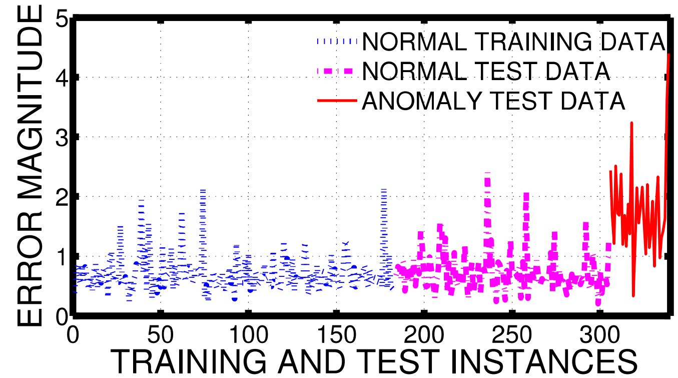
Single class classification

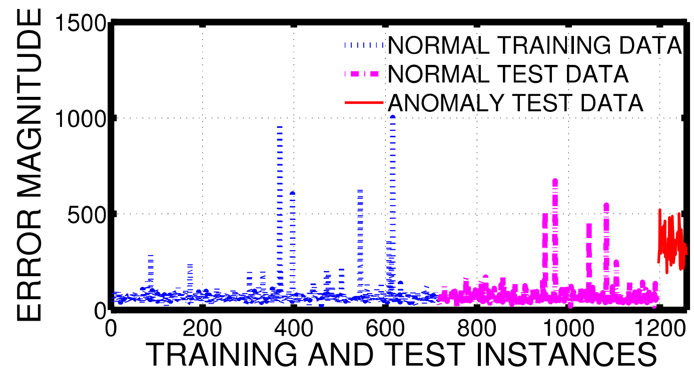
- Using sparse code representation of input feature vector
  - Represent high-dimensional data using few new dimensions
  - Captures inherent nature of processes generating data
  - Only an approximation. So there is always an “error”.
- Error is key!
  - When normal data is represented in sparse code, errors are similar (i.e. belong to a pre-characterized distribution)
  - When anomalous data is represented in sparse-code, errors do not belong to this distribution
- State-of-art applications in:
  - Signal processing
  - Image processing

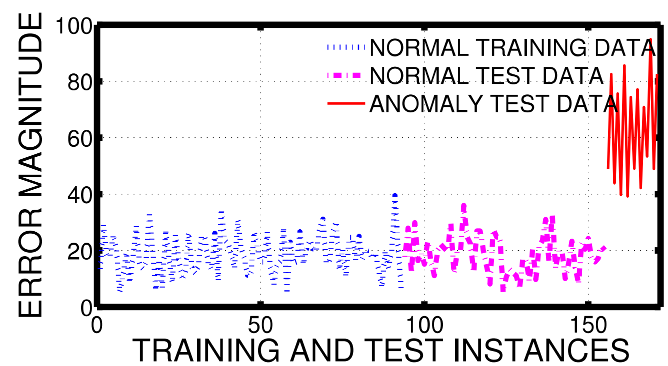


Sparse code representation

- Difference in sparse coding errors for normal and anomalous data.
- Applications shown in figure:
  - Wordcount
  - Pi estimation
  - Grep
- Training: Offline; Find sparse code for all training vectors.
- Testing: Find error for just one test vector

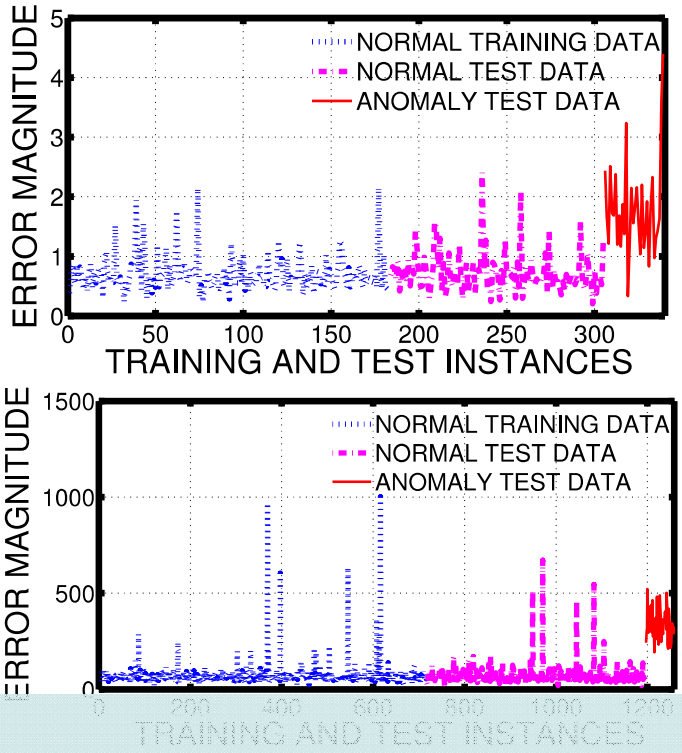






Sparse code representation

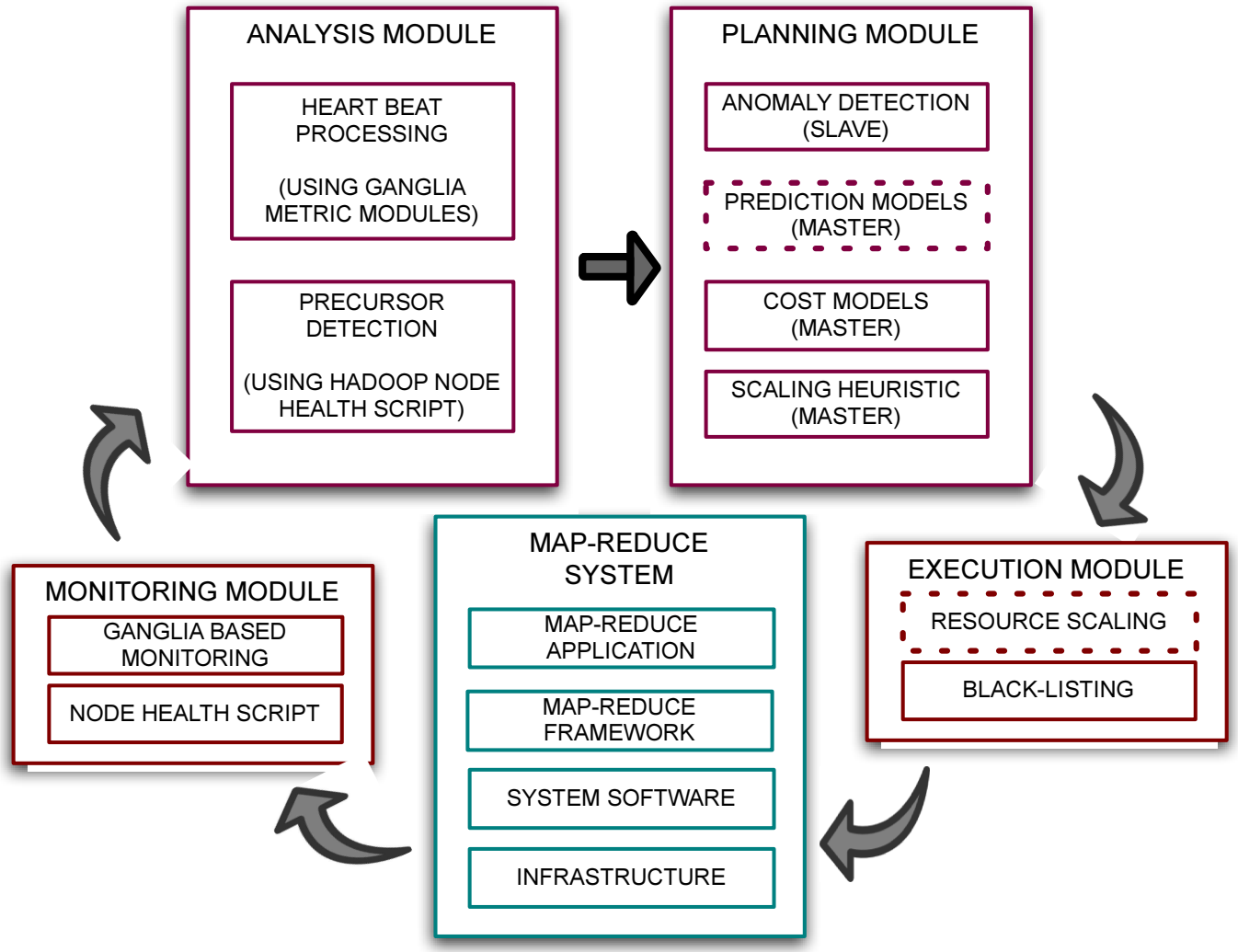
- Difference in sparse coding errors for normal and anomalous data.
- Applications shown in figure:
  - Wordcount
  - Pi estimation
  - Grep



- Training: Online, Find sparse code for training vectors
- Testing: Find error for just one test vector

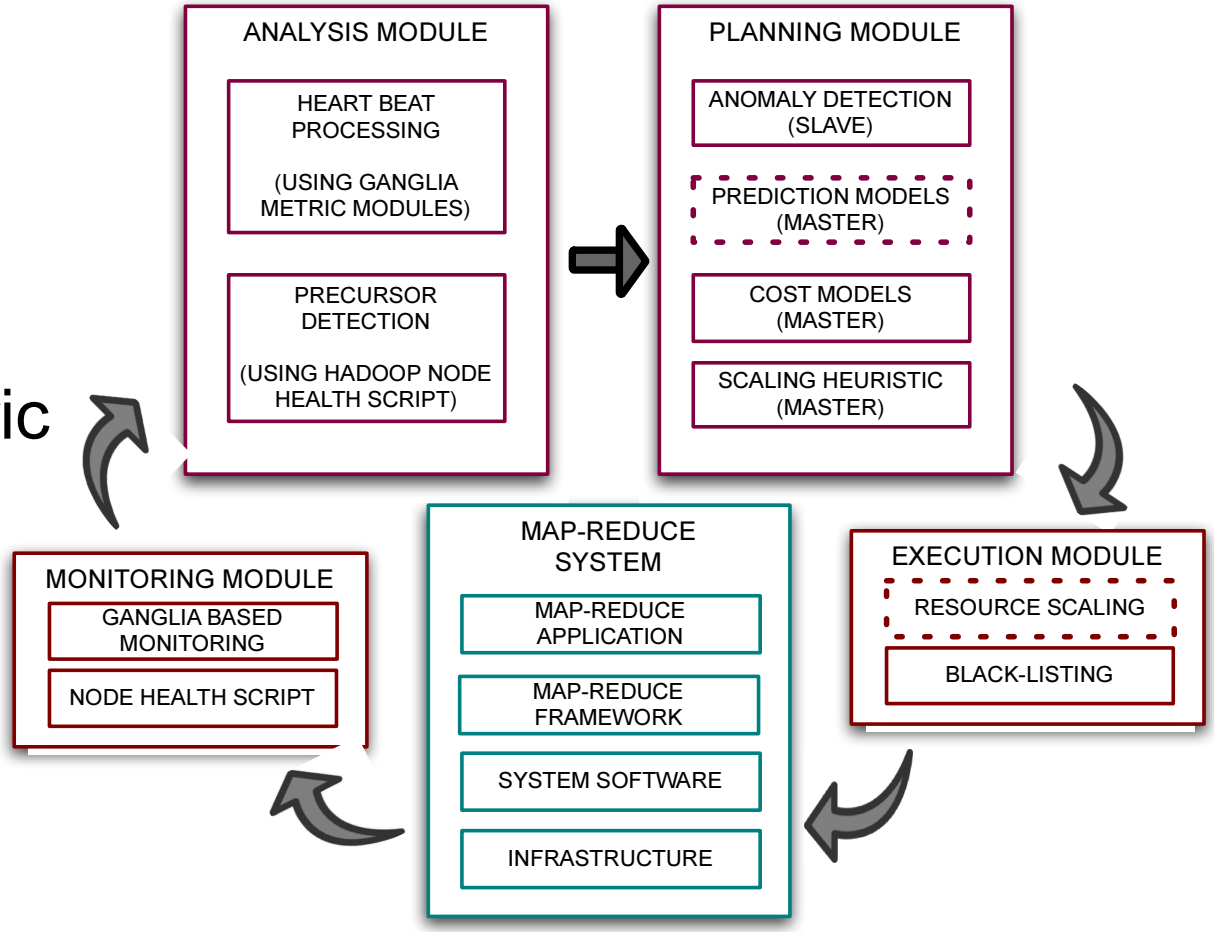
- Consequences:
  - Fast testing time (few vector operations)
  - No need for anomaly data
  - Decentralized local models -> scale well

MAPE components of FMR



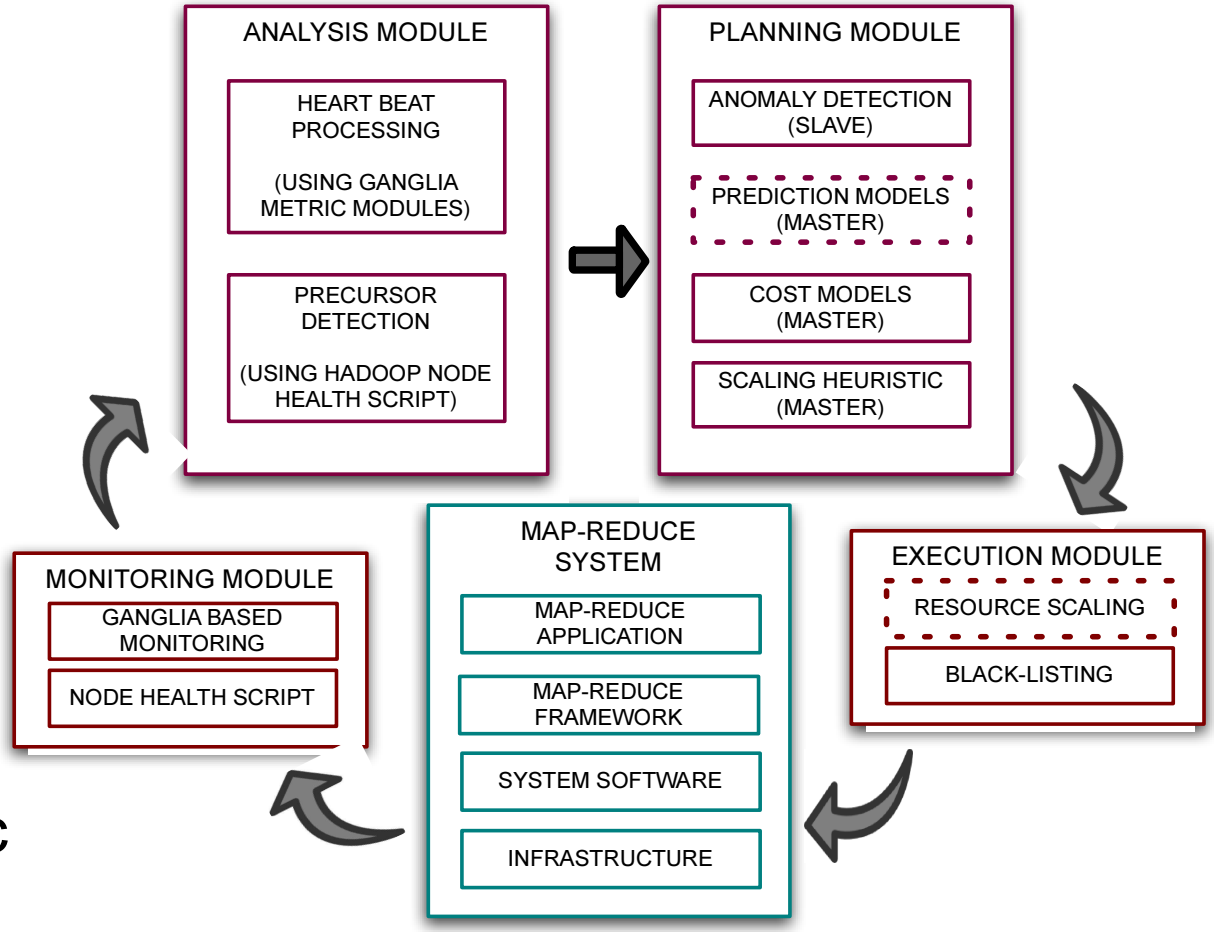
MAPE components of FMR

- Monitoring & Analysis:
  - Ganglia + metric modules
  - Node health script



MAPE components of FMR

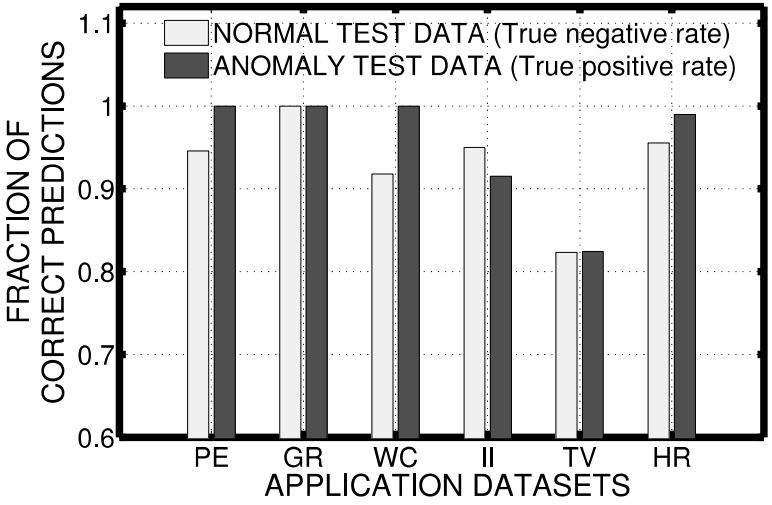
- Planning & Execution:
  - Anomaly detection
  - Execution time prediction:
    - Regression ML models [ICCCN2012]
  - Scaling heuristic
    - Analytical model

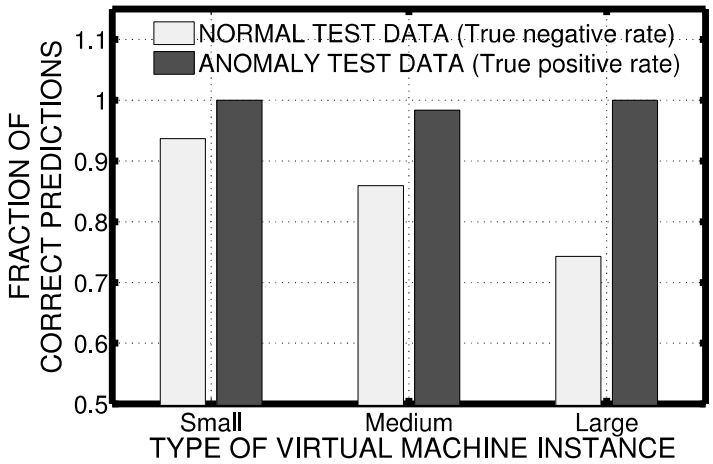


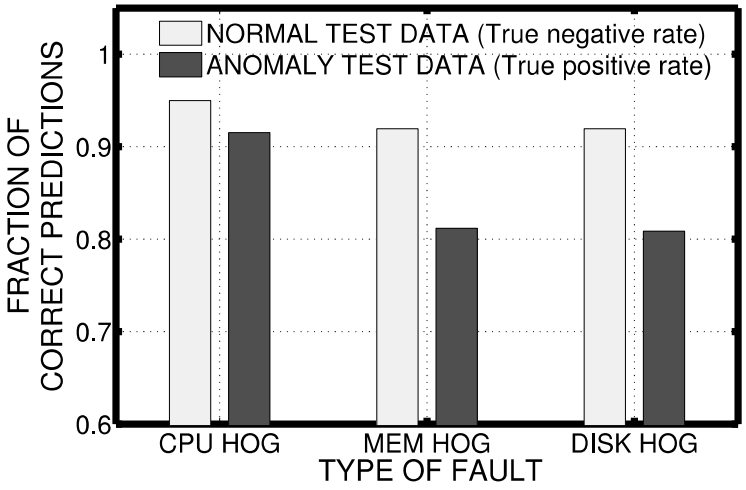
Evaluation – Anomaly detection

Anomaly detection accuracy for

- Different applications
- Different types of faults
- Different VM instance sizes



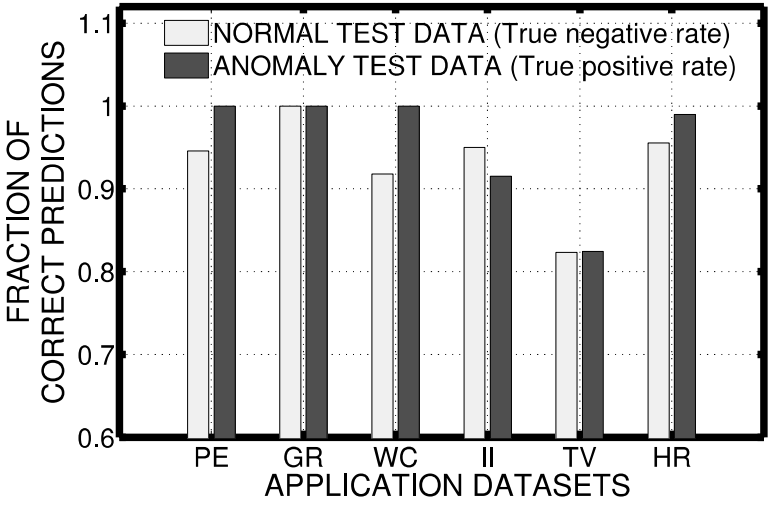


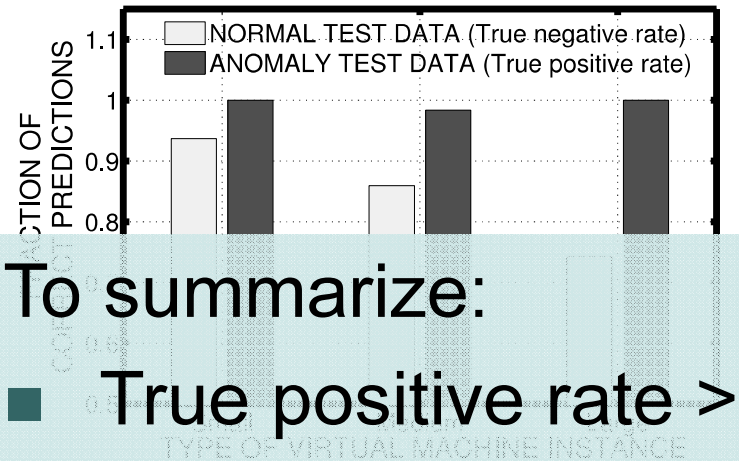


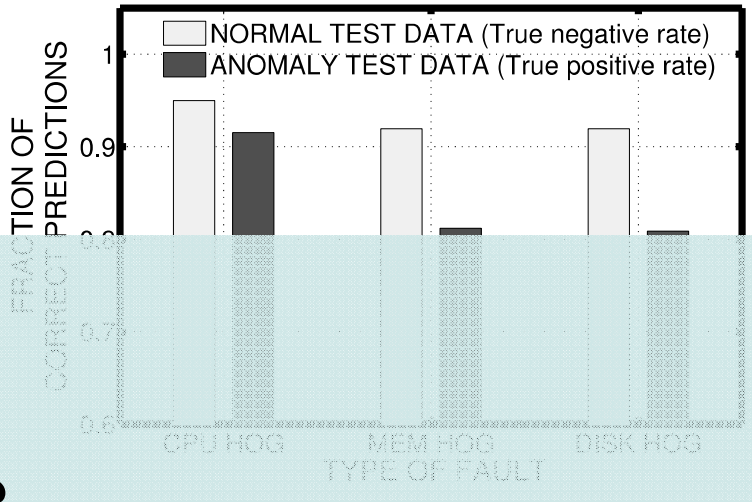
Evaluation – Anomaly detection

Anomaly detection accuracy for

- Different applications
- Different types of faults
- Different VM instance sizes







To summarize:

- True positive rate >0.7
- True negative rate > 0.8

Evaluation – Anomaly detection

COMPARISON OF ANOMALY DETECTION TECHNIQUES

Application	Multilayer Perceptron	K-means clustering	Support Vector Machines	Sparse-coding
	True positive rate / True negative rate			
PI	1.0/0.96	0.99/0.88	0.76/0.3	1.0/0.93
GR	1.0/0.94	0.7/0.31	1.0/0.65	1.0/1.0
[illegible]	[illegible]/0.88	0.96/1	1.0/0.69	1.0/0.92
II	0.99/0.95	0.8/0.66	1.0/0.49	0.92/0.95
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
HR	0.99/0.98	0.975/0.99	1.0/0.51	0.99/0.96

- Multilayer perceptron, K-means clustering: Requ[illegible] training data
- Single-class SVM: Low true negative rates

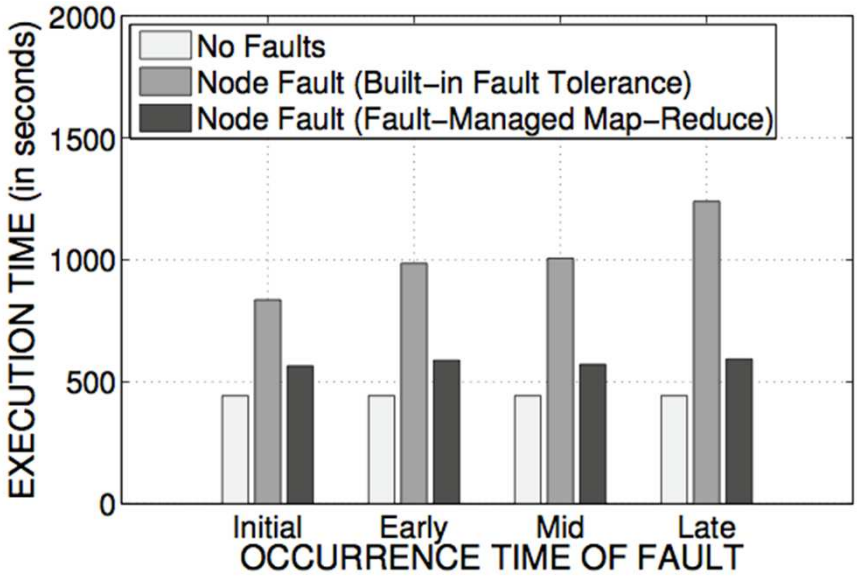
To summarize:

We now have an anomaly detection technique with

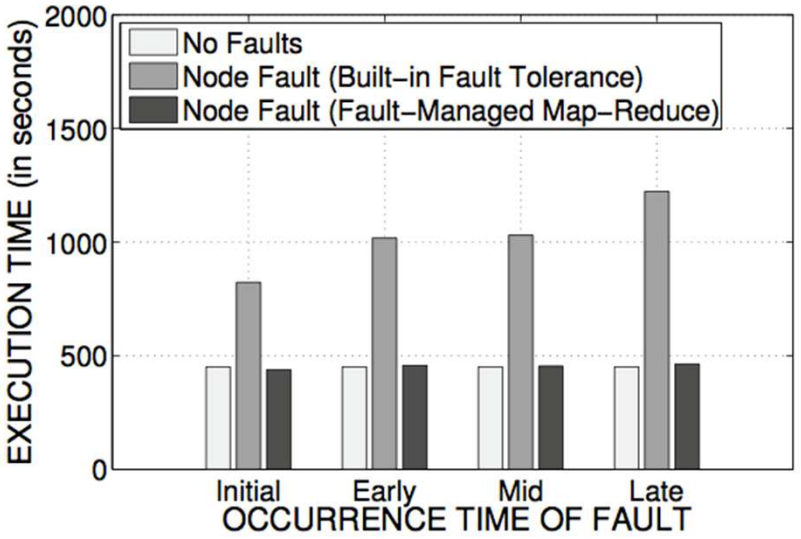
- Good accuracy
- Fast training and testing time
- Capable of scaling to a large number of slave nodes
- No need for anomalous training data

Evaluation: Closed-loop FMR

- Anomaly detection through sparse coding provides a key component to complete the control loop
- Example evaluation of FMR on a 32-node and 62-node cluster:



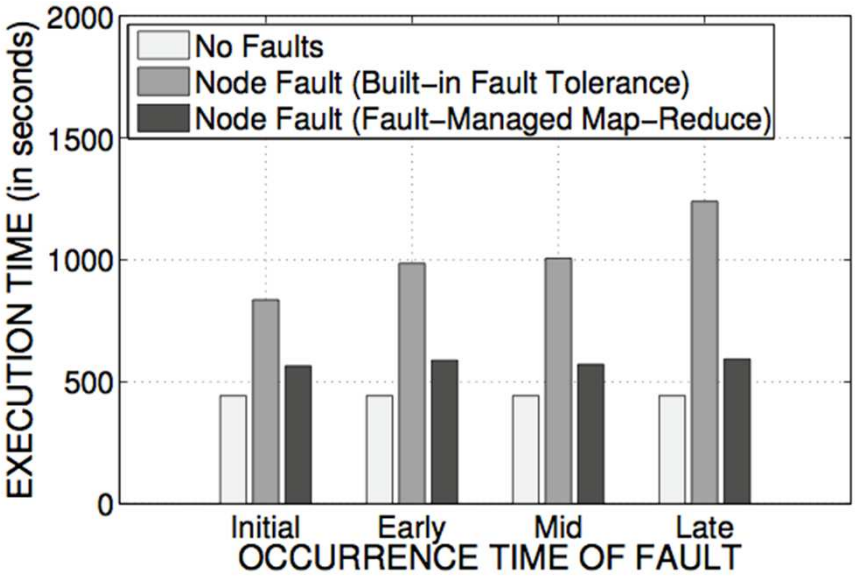
(c) 32 node cluster + 10 nodes for scaling

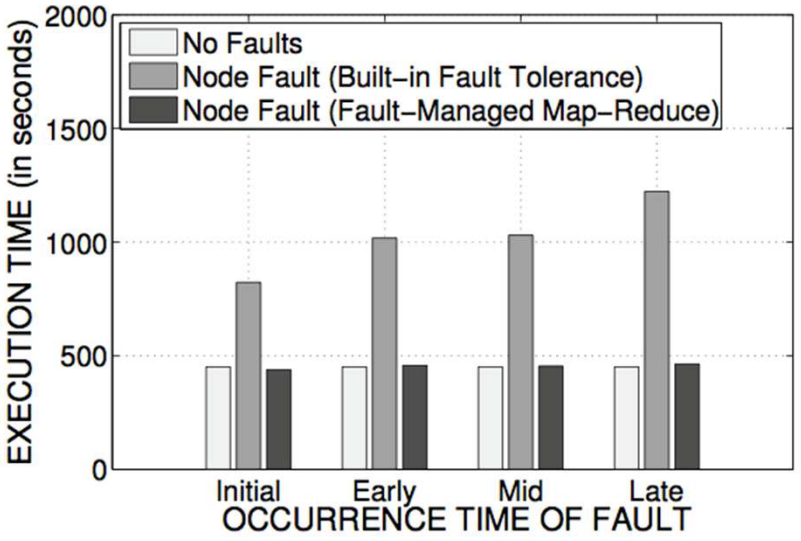


(f) 62 node cluster + 10 nodes for scaling

Evaluation: Closed-loop FMR

- Anomaly detection through sparse coding provides a key component to complete the control loop
- Example evaluation of FMR on a 32-node and 62-node cluster:





To summarize:

- Averaged over all experiments: Penalty reduced from 119% to 14%

Conclusions

- Software definition of different layers of IT systems enables flexibility, optimizations and functionality not possible otherwise
- Much work needs to be done
  - What control and management capabilities can be factored out or added and made available to upper layers?
  - What are the security and management implications for lower layers?
  - What should be static and what should be dynamic?
  - Can we build self-software-defined systems?

Acknowledgements

- Work done jointly with Mauricio Tsugawa, Andrea Matsunaga and Selvi Kadirvel
- Funding from NSF and CAC industry members
- Colleagues from ACIS and NSF Center for Cloud and Autonomic Computing