cloudscaling

Open Cloud Networking Vision

The state of OpenStack networking and a vision of things to come...

Dan Sneddon

Member Technical Staff

Twitter: @dxs

OCS 2.0

Public Cloud Benefits | Private Cloud Control | Open Cloud Economics

Our Journey Today

1. Cloudscaling Introduction
2. Elastic Clouds and Private Hybrid Clouds
3. Advanced Networking
4. Future Vision
5. Network-Based Service Resiliency

Elastic Clouds

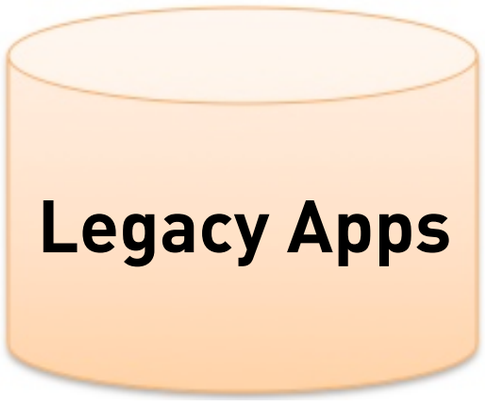
cloudscaling

Two Cloud Infrastructure Models

Two Cloud Infrastructure Models

1

Enterprise Virtualization



vmware SAVVIS EMC² terremark

Two Cloud Infrastructure Models

1 Enterprise Virtualization

vmware SAVVIS EMC² terremark

2 Elastic Infrastructure

amazon web services™

Google Compute Engine

Elastic Cloud vs Enterprise Virtualization

	Enterprise Virtualization	Elastic Cloud
Applications	Traditional & Legacy	Dynamic
Scaling Architecture	Managed Silos	Horizontal
Technology Stack	Heavy & Proprietary	Distributed & Open
Price/Performance	Low	High (4-7x better)
Failure Domains	Large	Small
Provisioning	Slower & Manual	Faster & 100% API
Best For:	Server consolidation and lower datacenter mgmt costs	On-demand, scale-out infrastructure for new apps

Public Elastic Clouds Are Not Enough

Why?

- **Loss of control/governance**
- **Security concerns**
- **Integration w/ existing systems**
- **Data privacy/patriation issues**
- **Expensive at scale**
- **Limited performance options**
- **Geographic reach**

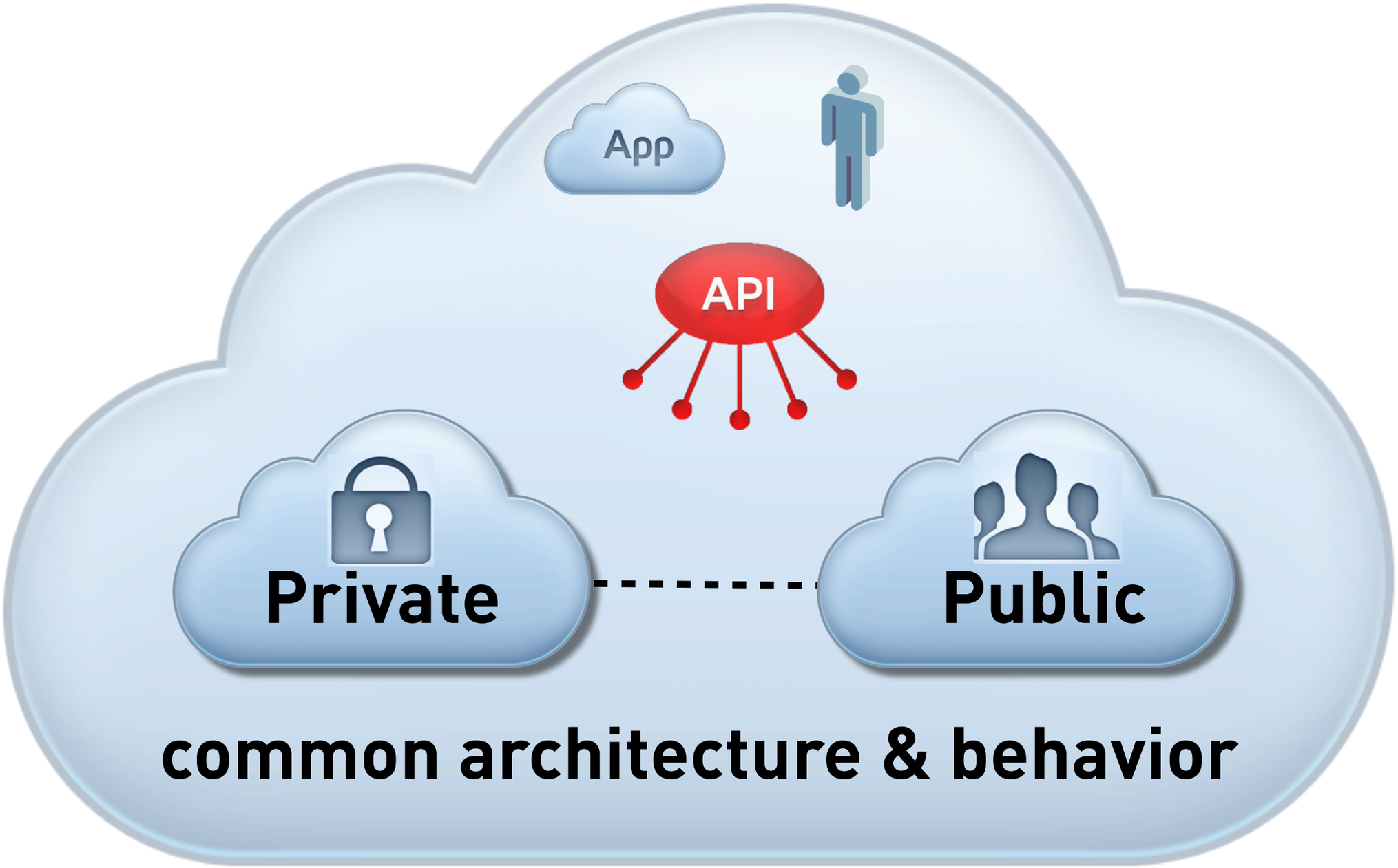
Wanted: Private Elastic Clouds

Dynamic Applications

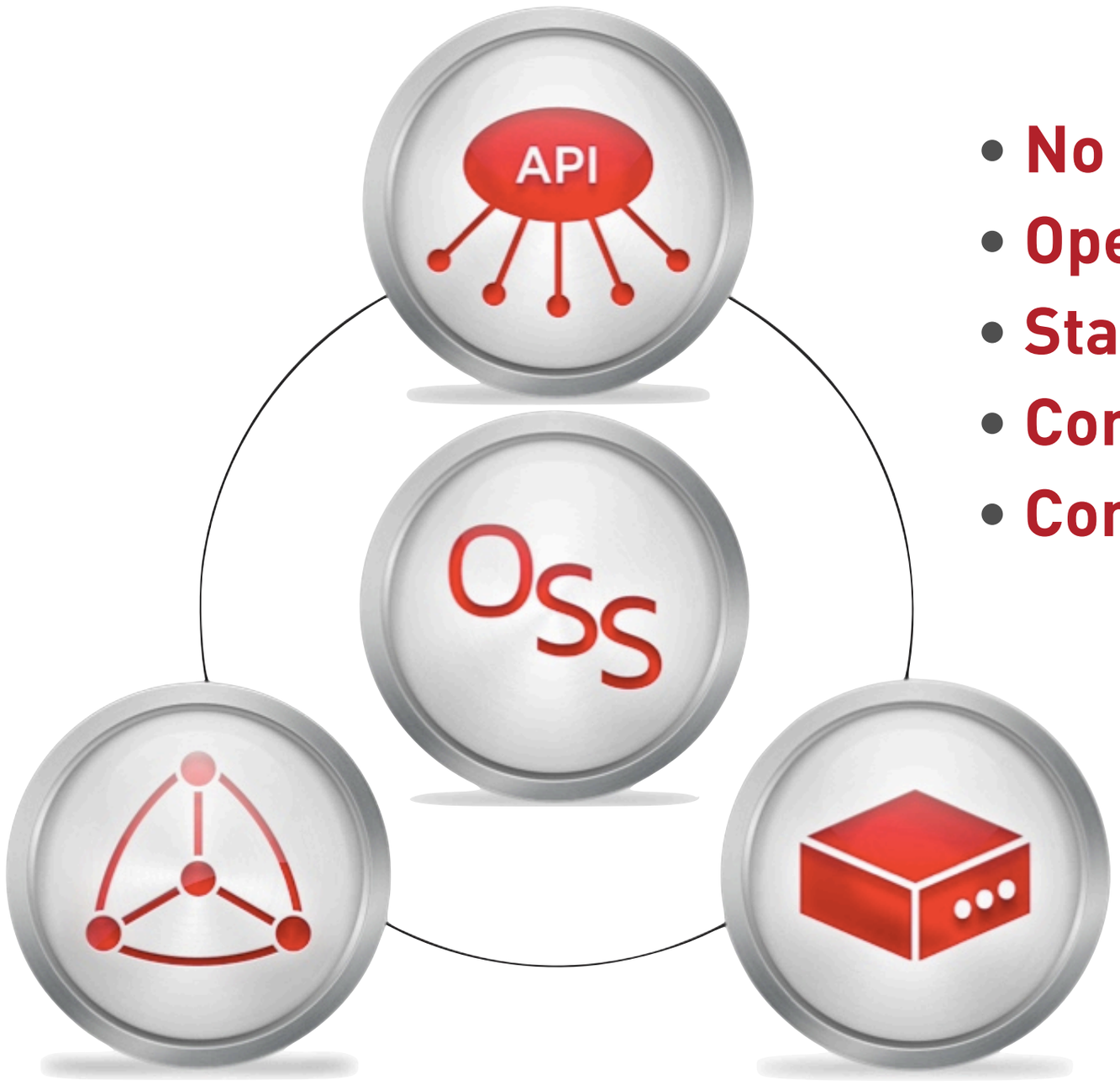
Public Benefits

Private Control

Wanted: Private/Public Federation

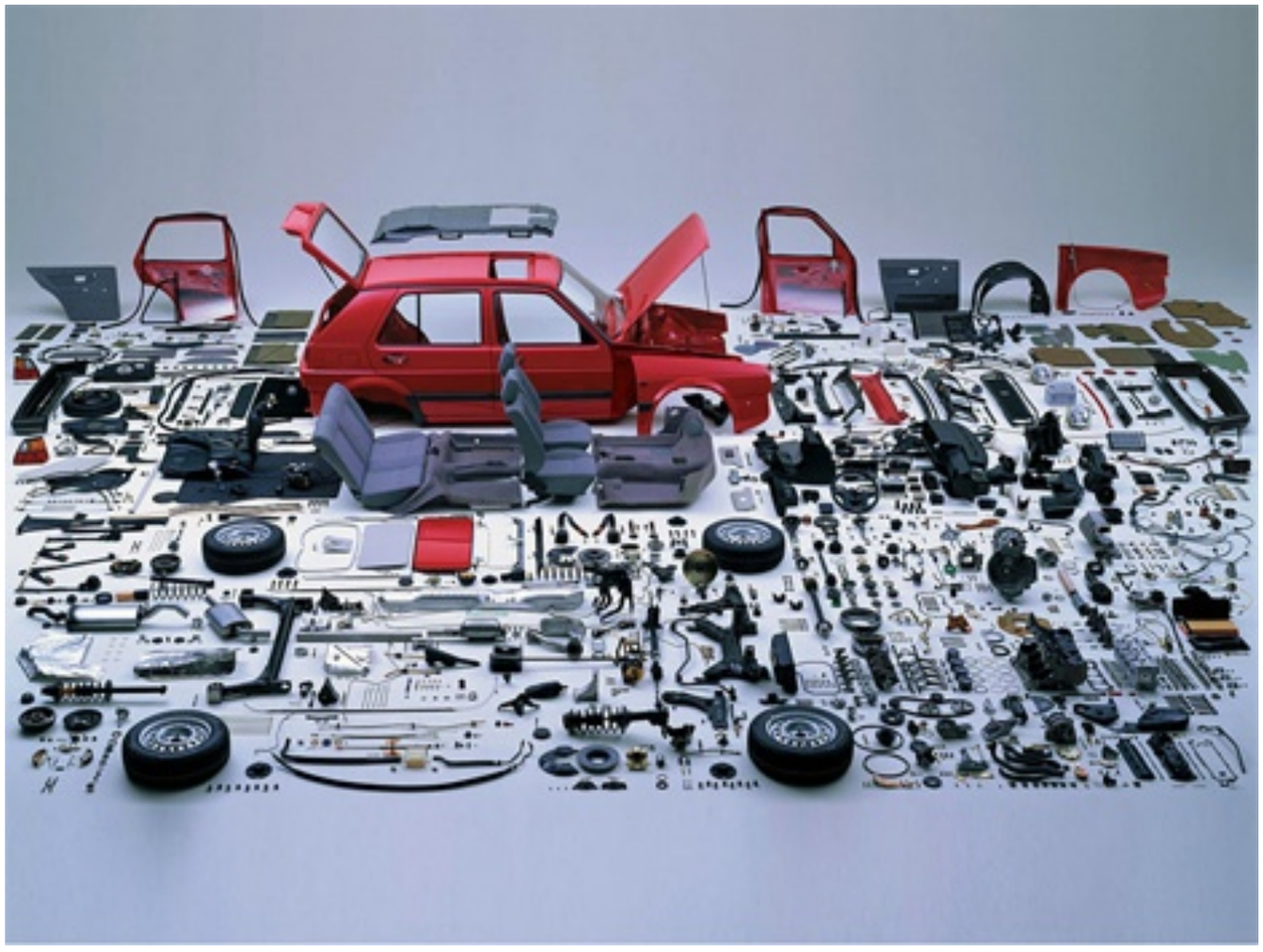


Wanted: Open Economics/Choice



- **No vendor lock-in**
- **Open source software**
- **Standard interfaces**
- **Commodity hardware**
- **Community/rapid innovation**

Can I Build A Private Elastic Cloud Myself?



Yes, but painfully

- **Highly complex**
- **Costly to build and maintain**
- **Not where the value is**
- **No innovation path**
- **Not production-grade**
- **Requires special skills**

Our Solution:

Open Cloud System

Our Product

Open Cloud System 2.0

The most reliable, scalable and production-grade solution for private elastic clouds powered by OpenStack technology.

Powered by

OpenStack Elastic Cloud

Compute	Block Storage	Object Storage	Advanced Networking
Nova	Cinder	Swift / Hadoop	Quantum

Production-Grade Focus

Performance

Can the system guarantee quality of service, responsiveness, scalability, and economic performance?

Availability

Does the system offer redundancy, resiliency, fault isolation, graceful degradation, and scale-out engineering?

Security

Are best practices of default deny, least privilege, a minimal attack surface, and encryption/data privacy followed?

Maintainability

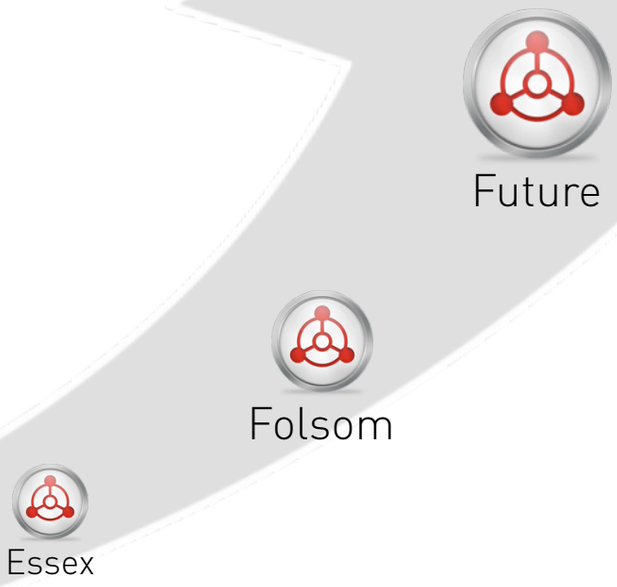
Does the system provide a reliable upgrade path for new enhancements & system updates, transparency & measurability of performance, comprehensive lifecycle management, and predictable behavior?

OCS is Production OpenStack

OCS is more than just software ... it's OpenStack release synchronicity, Cloudscaling innovations, a compelling roadmap, community involvement & production support from a team with deep operational experience.

	openstack CLOUD SOFTWARE	
Virtual Machines (Nova)	✓	✓
Object Storage (Swift)	✓	✓
VM Image Management (Glance)	✓	✓
Identity Service (Keystone)	✓	✓
Hardware Lifecycle Mgmt		✓
Topology & Data Model		✓
Block-based Architecture		✓
Security Hardening		✓
Scale-Out Networking		✓
Service Redundancy		✓
Public Cloud Compatibility		✓

OCS is **open core** software that tracks the OpenStack release cycle closely.



OpenStack ecosystem contributions

aws-compat, zeromq-rpc-driver, nova-gerrit-monitor, tarkin, cs-nova-simplescheduler, sheep

Advanced Networking

OCS Advanced Networking

Layer 3 Networking Plugin

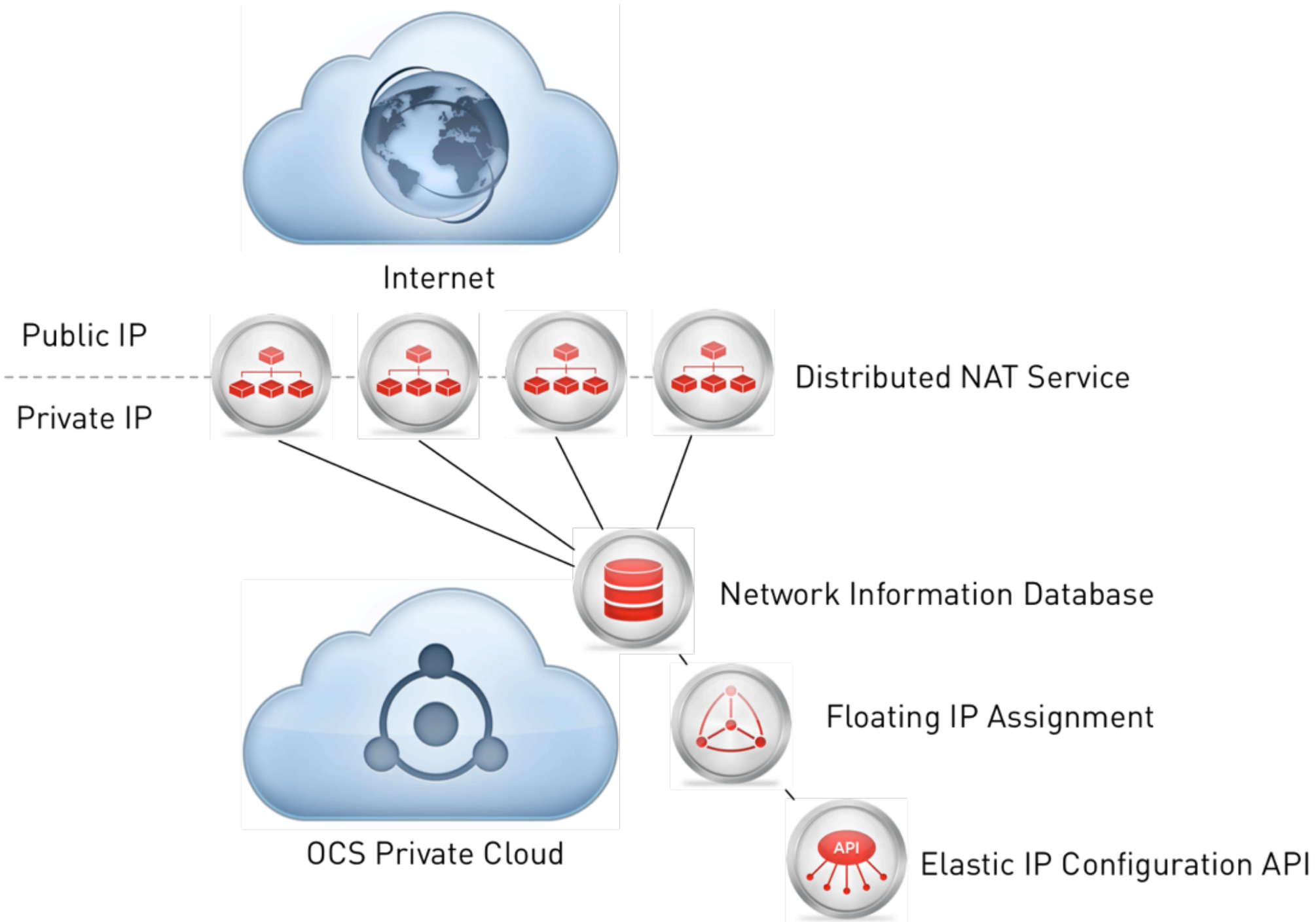
Distributed NAT Service

Network Definition API

Tenant Isolation

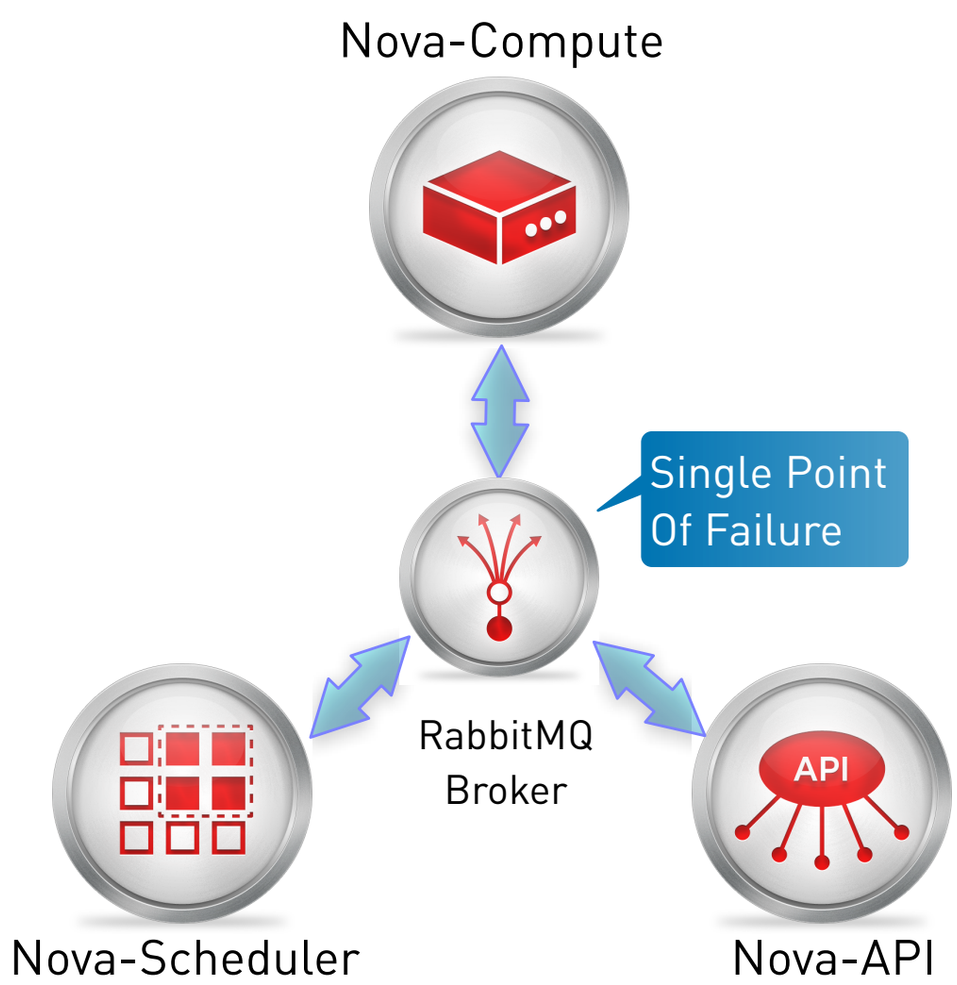
Distributed Load Balancing

Distributed NAT Service



Brokerless Messaging With ZeroMQ

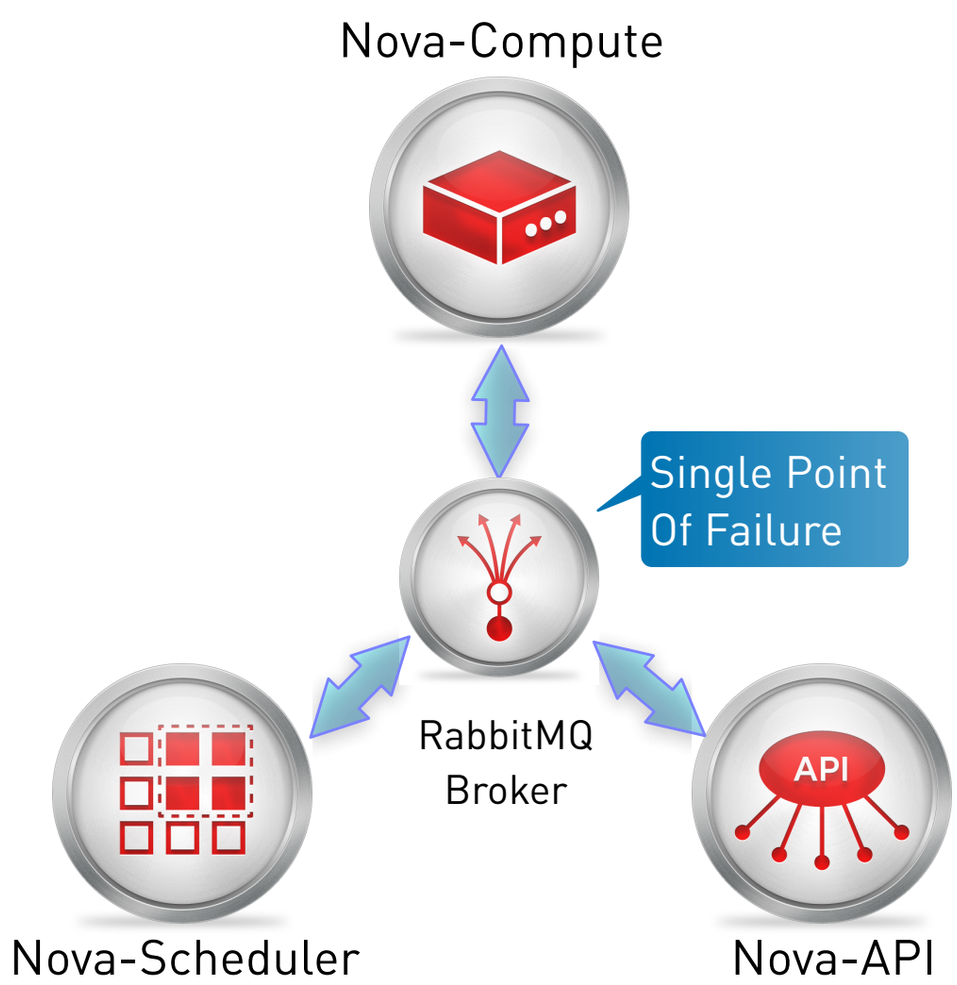
Avoiding RabbitMQ's Single Point Of Failure



RabbitMQ
(Brokered)

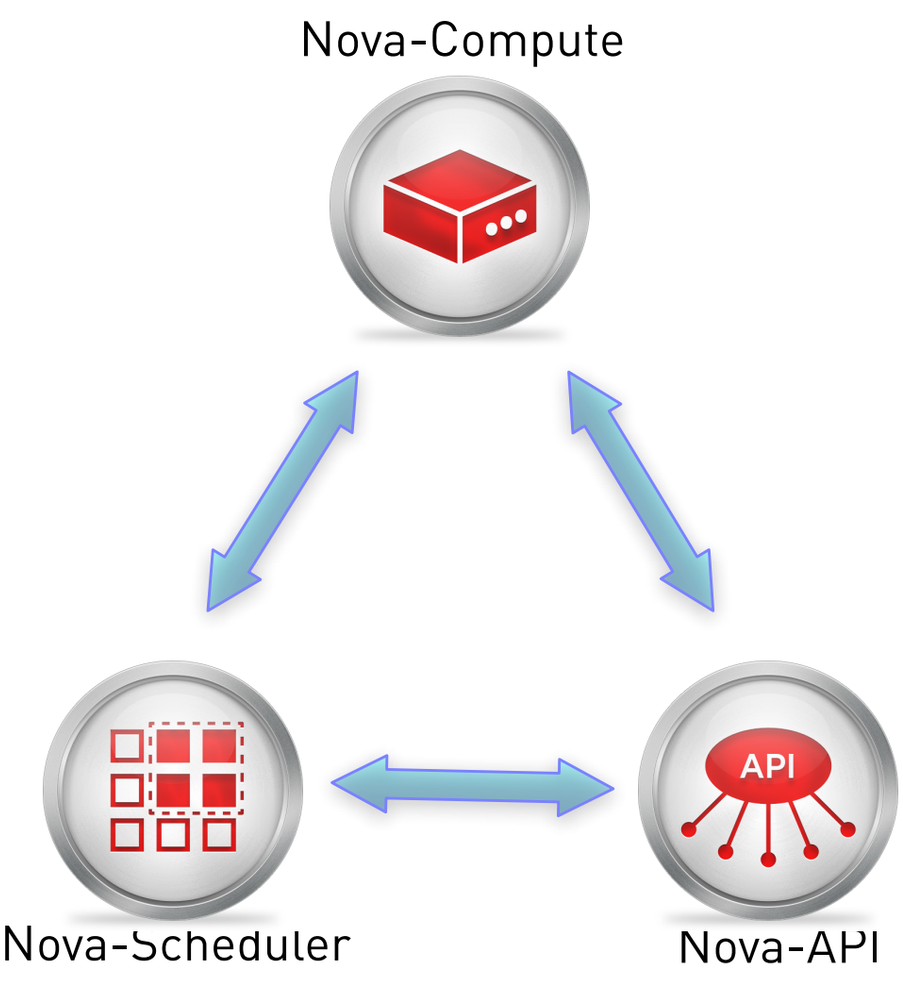
Brokerless Messaging With ZeroMQ

Avoiding RabbitMQ's Single Point Of Failure



RabbitMQ
(Brokered)

vs.

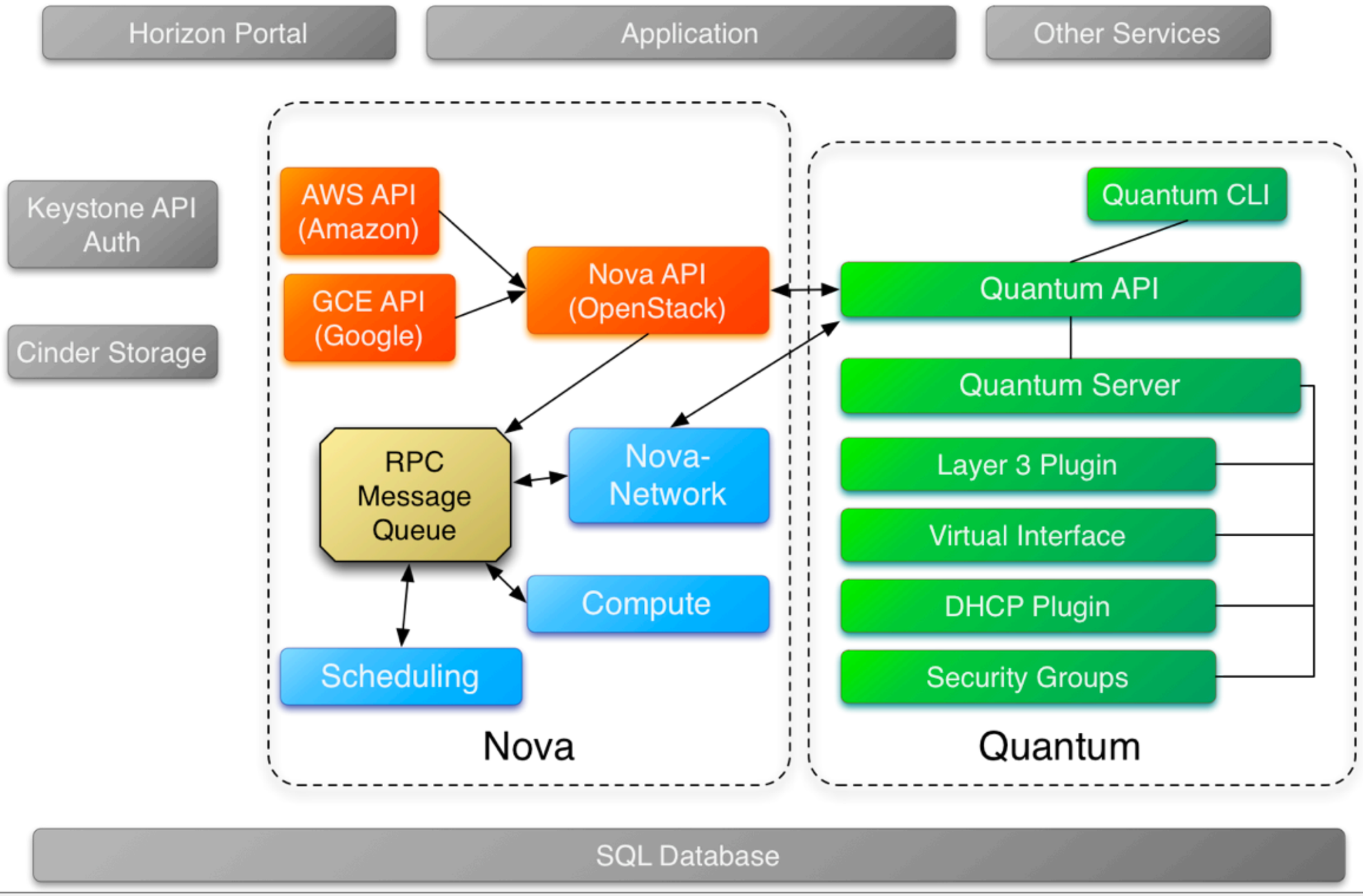


ZeroMQ
(Peer To Peer)

Quantum Development

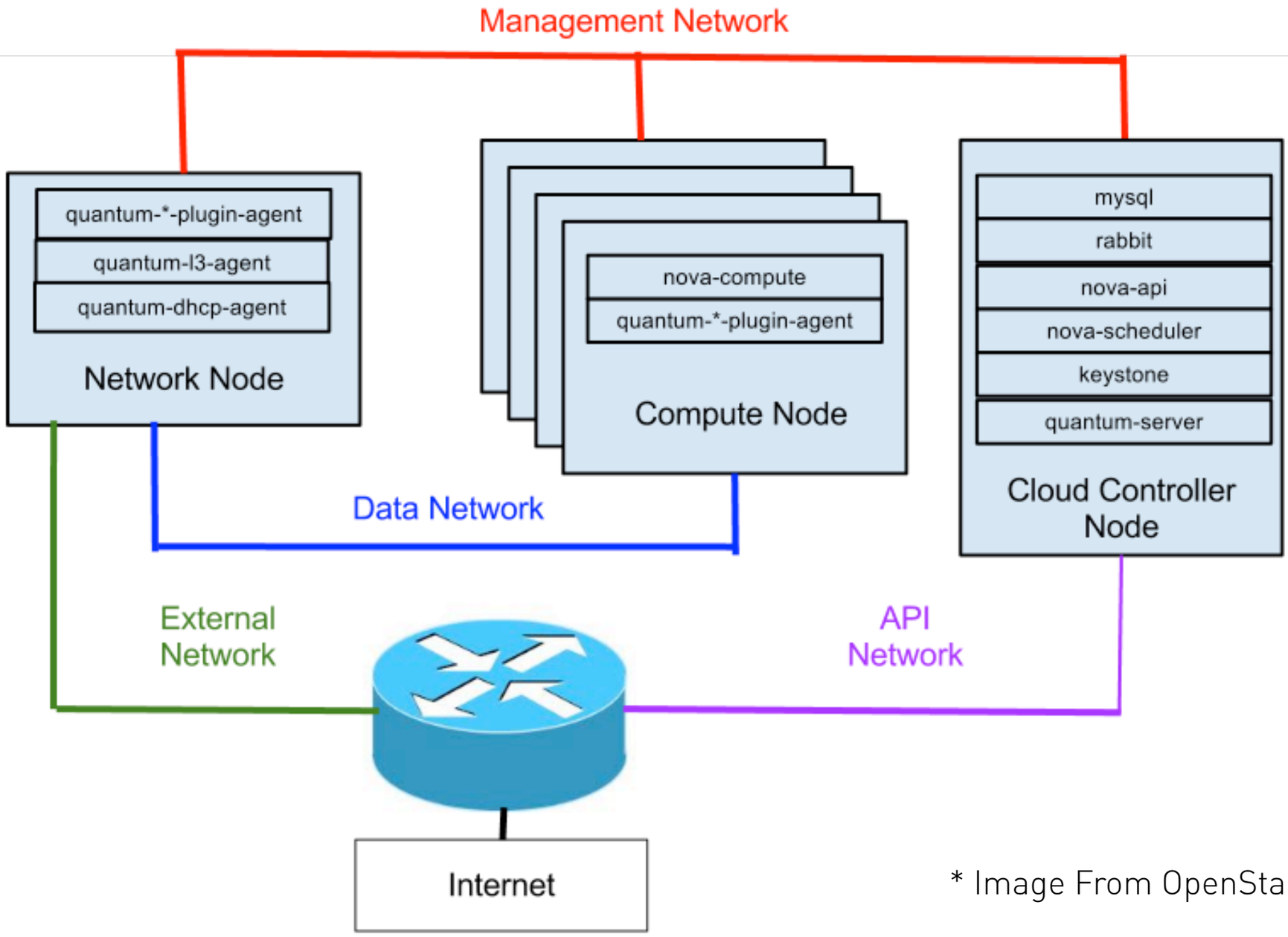
OpenStack Networking

Nova Manages VMs, Quantum Manages Network



Quantum Architecture

Unified Open APIs For Advanced Networking



* Image From OpenStack Documentation

Quantum in Folsom (2012)

- Layer 3 plugin to Nova Networking
- DHCP service plugin
- Network Address Translation (NAT)
- No overlapping IP assignments
- Very limited multi-tenant security

Quantum in Grizzly (2013)

- Complete Rewrite of Quantum API (v3)
- Security Group Enhancements
- Distributed Firewall Configuration API
- Distributed Load Balancer Services API
- VPN Services API
- Better integration with OpenFlow Controllers

Quantum Compatibility

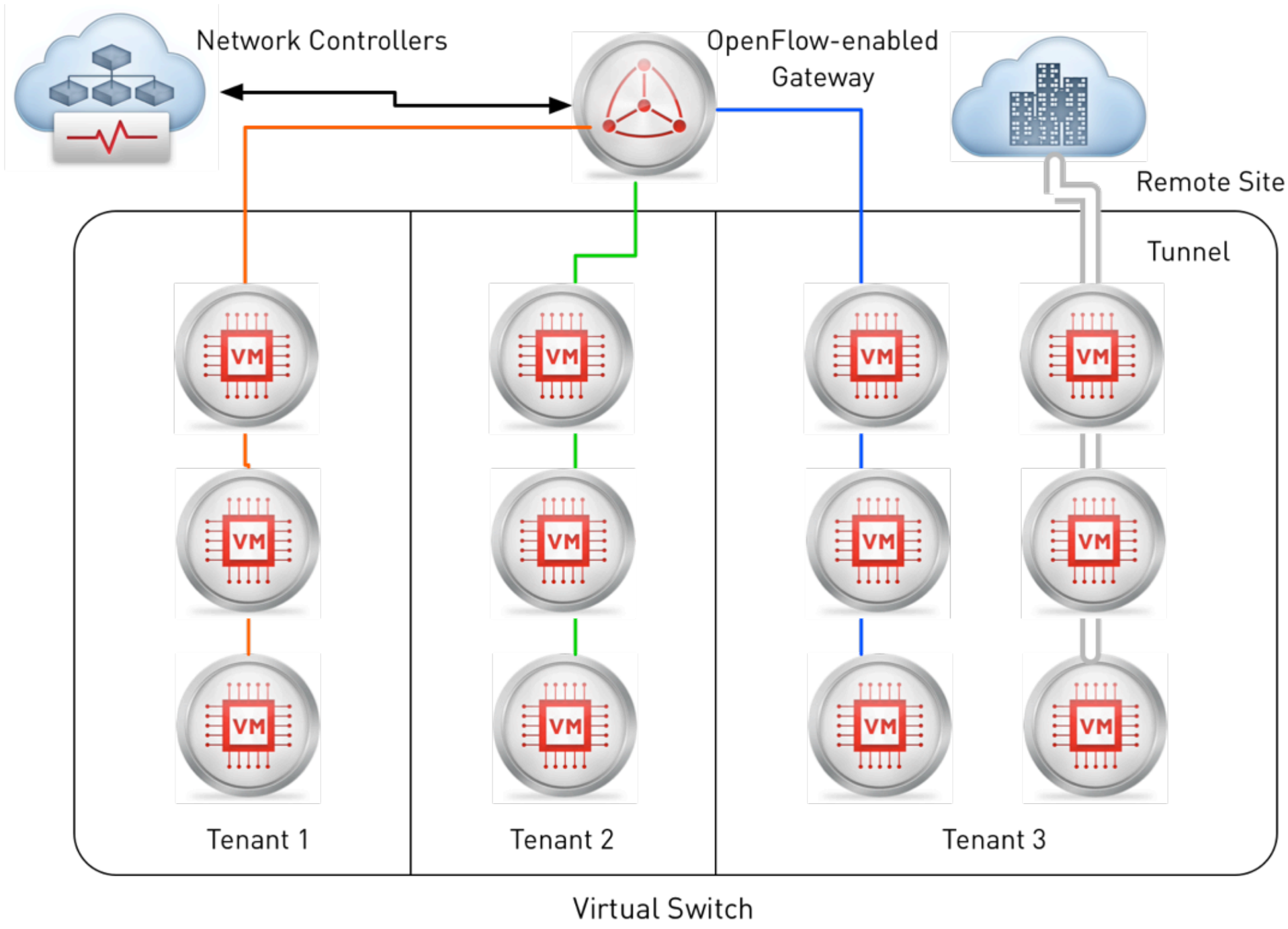
Lots Of Choices For Virtual Network/SDN Providers

- Open vSwitch. http://www.openvswitch.org/openstack/documentation
- Nicira NVP. quantum/plugins/nicira/nicira_nvp_plugin/README and http://www.nicira.com/support.
- Midokura. http://www.midokura.com/midonet/openstack/
- BigSwitch. http://www.bigswitch.com/sites/default/files/sdn_resources/openstack_aag.pdf
- Cisco. quantum/plugins/cisco/README and http://wiki.openstack.org/cisco-quantum
- Linux Bridge. quantum/plugins/linuxbridge/README and http://wiki.openstack.org/Quantum-Linux-Bridge-Plugin
- Ryu. quantum/plugins/ryu/README and http://www.osrg.net/ryu/using_with_openstack.html
- NEC OpenFlow. http://wiki.openstack.org/Quantum-NEC-OpenFlow-Plugin

Network Virtualization

Network Virtualization Use Case

Isolation and Network Tunnels in Multi-Tenant VM Host



Origin Of Virtual Networking

It makes more sense if you know where it came from

- Despite the appearance of new technologies, virtual networking and SDN have been around a long time

- Prior to the latest protocols, virtual networking was achieved through creative use of VLANs, proxies, dynamic routing, GRE tunnels, and expect scripts.

- MPLS was an attempt by Cisco and the telcom providers to do managed virtual networking

- SDN isn't new either, e.g. Ciscoworks, NetConf, Opnet

Future Of Virtual Networking

A few bold predictions

- Increased adoption of OpenFlow is an obvious inevitability
- The biggest change on the horizon: virtualized hardware
- OSS soon to be a viable option for elastic cloud networking
- Networking will be about programming, not configuring
- It is already easier to do virtual networking at scale, but soon it will also be cheaper, leading to massive disruption

The Path To The Future

- Virtualizing network hardware: VRFs, VM support on routers, VXLANs, virtual instances

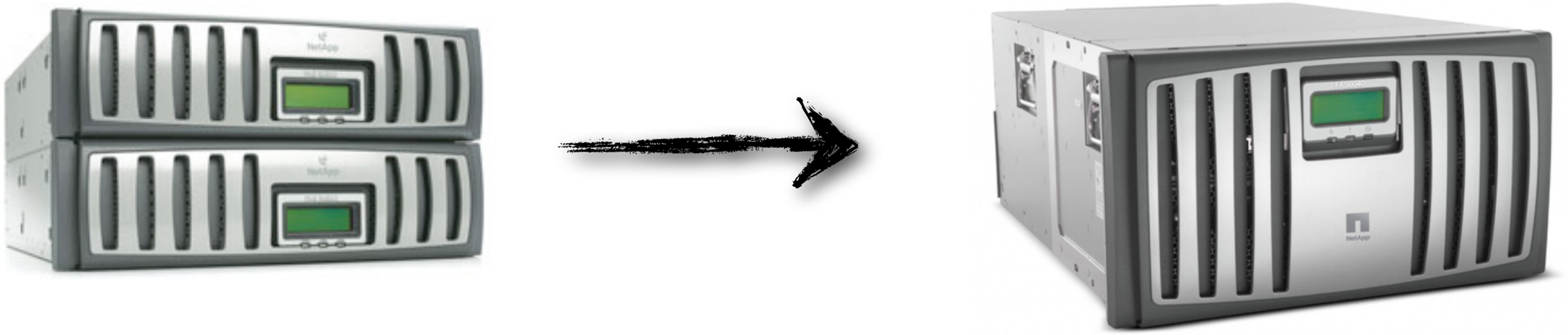
- SDN needs to be a software and hardware approach, device configuration should be done through common APIs

- A dramatic shift away from HA and failover, and toward distributed services with smaller failure domains and expectation of failure

Service Resiliency

What Do We Mean By "HA"?

We mean what most people mean ...



Two servers or network devices that look like one

“HA HA”?

HA pairs come in a couple flavors

Active / Passive

“HA HA”?

People like this flavor best, but it’s not always possible...

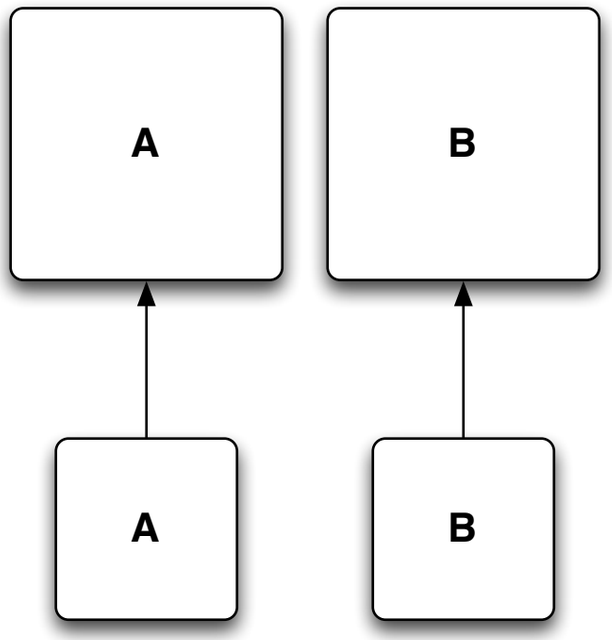
Active / Active

“HA HA HA HA HA”??

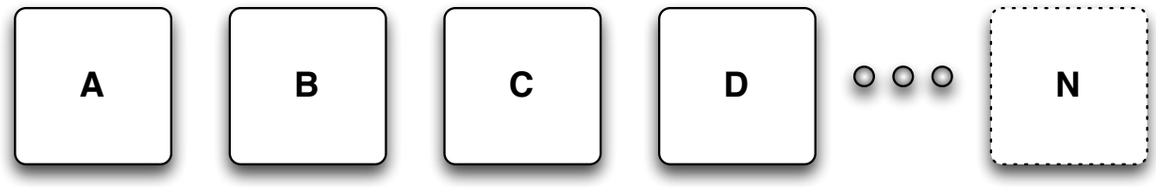
Many people wish they could get it more like this ...

HA cluster aka ‘massive operational nightmare’

What is Scale-out?



Scale-up - Make boxes bigger (usually an HA pair)



Scale-out - Make moar boxes

Scaling out is a mindset

Scaling up is like treating your servers as pets

bowzer.company.com

web001.company.com

Servers *are* cattle

"HA" Pairs Are an All-in Move

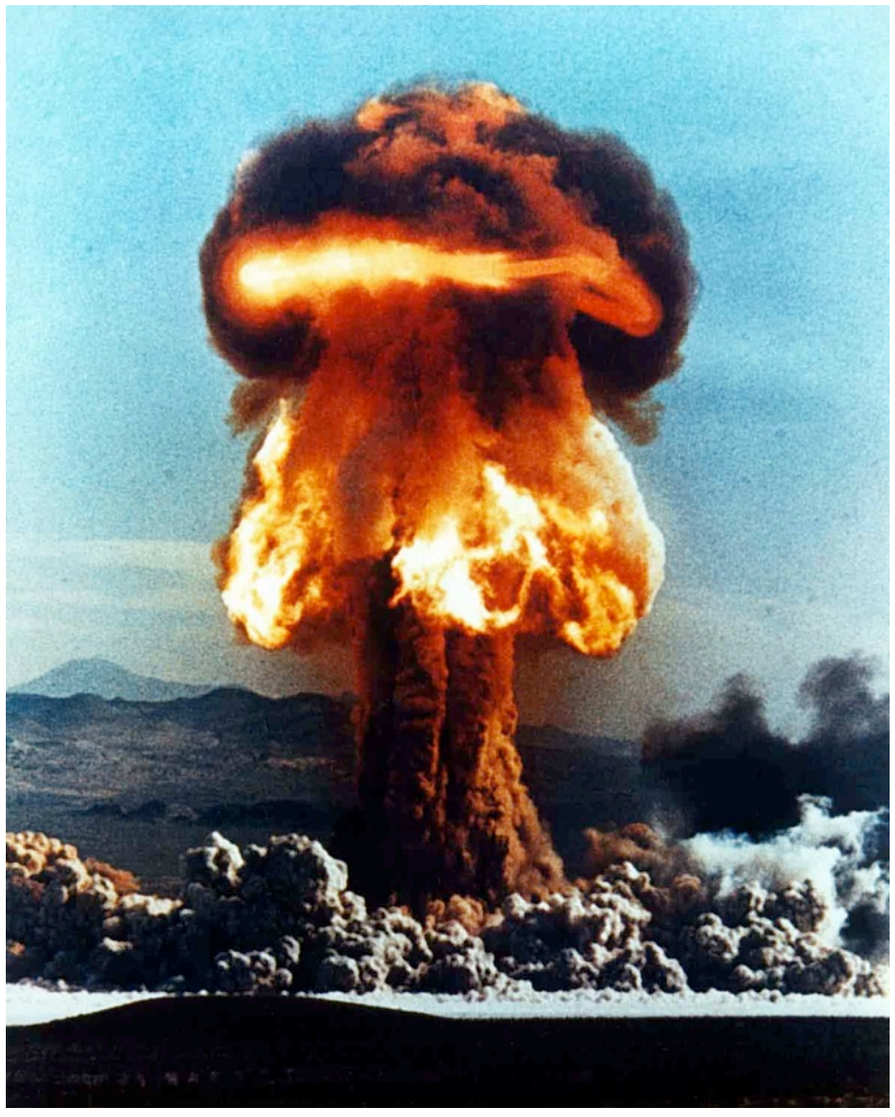
They better not fail …

Risk Reduction

Many small failure domains is usually better

Big failure domains vs. small

Would you rather have the whole cloud down or just a small bit for a short period of time?



Still a scale-up pattern ...
wouldn't you rather scale-out?

What's Usually an "HA" Pair in OpenStack?

Everything ...

Service Endpoints (APIs)

Messaging System (RPC)

Worker Threads (e.g. Scheduler, Networking)

Database (MySQL)

What needs to be an HA pair?

Not much needs state synchronization

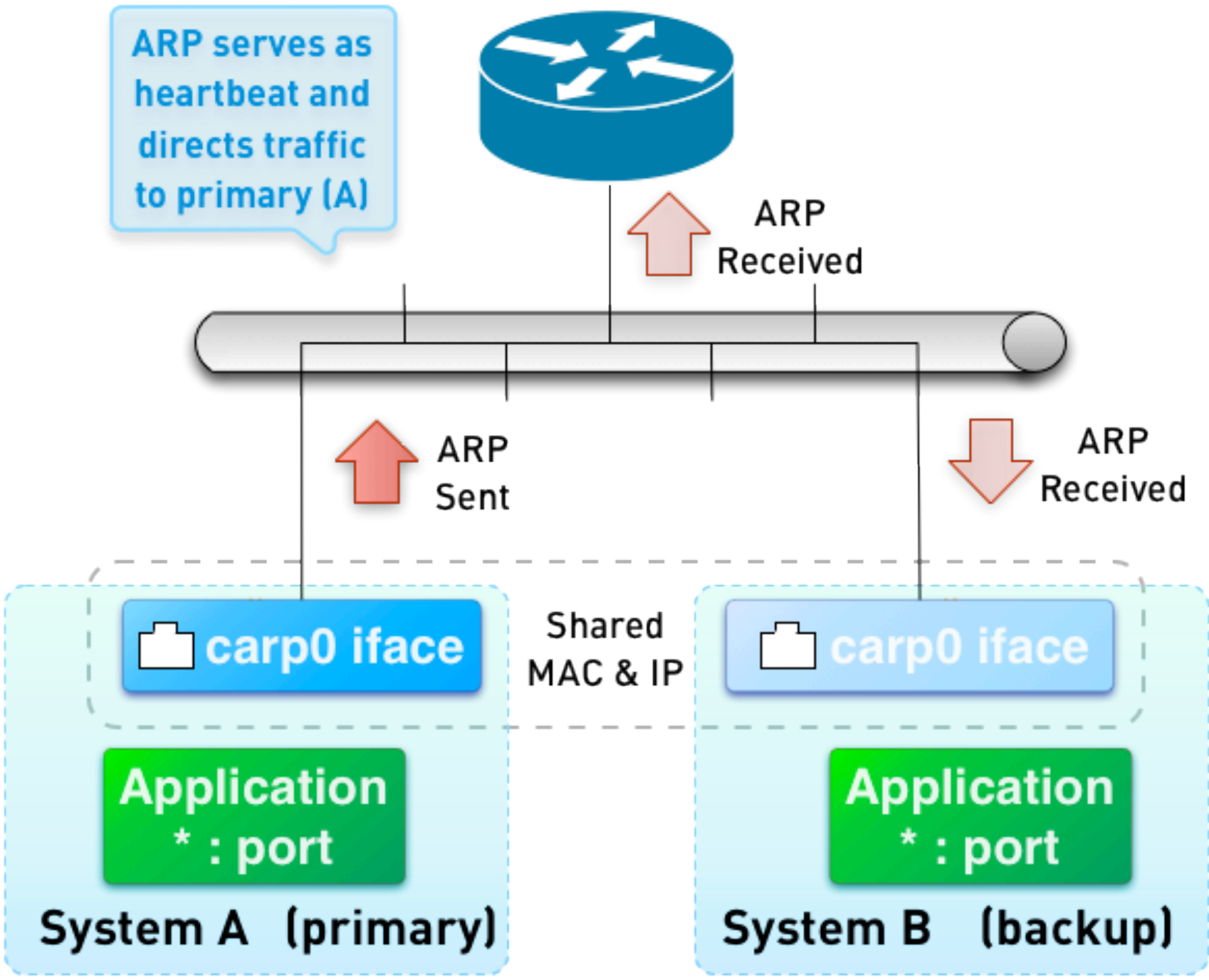
Service Endpoints (APIs)

Messaging System (RPC)

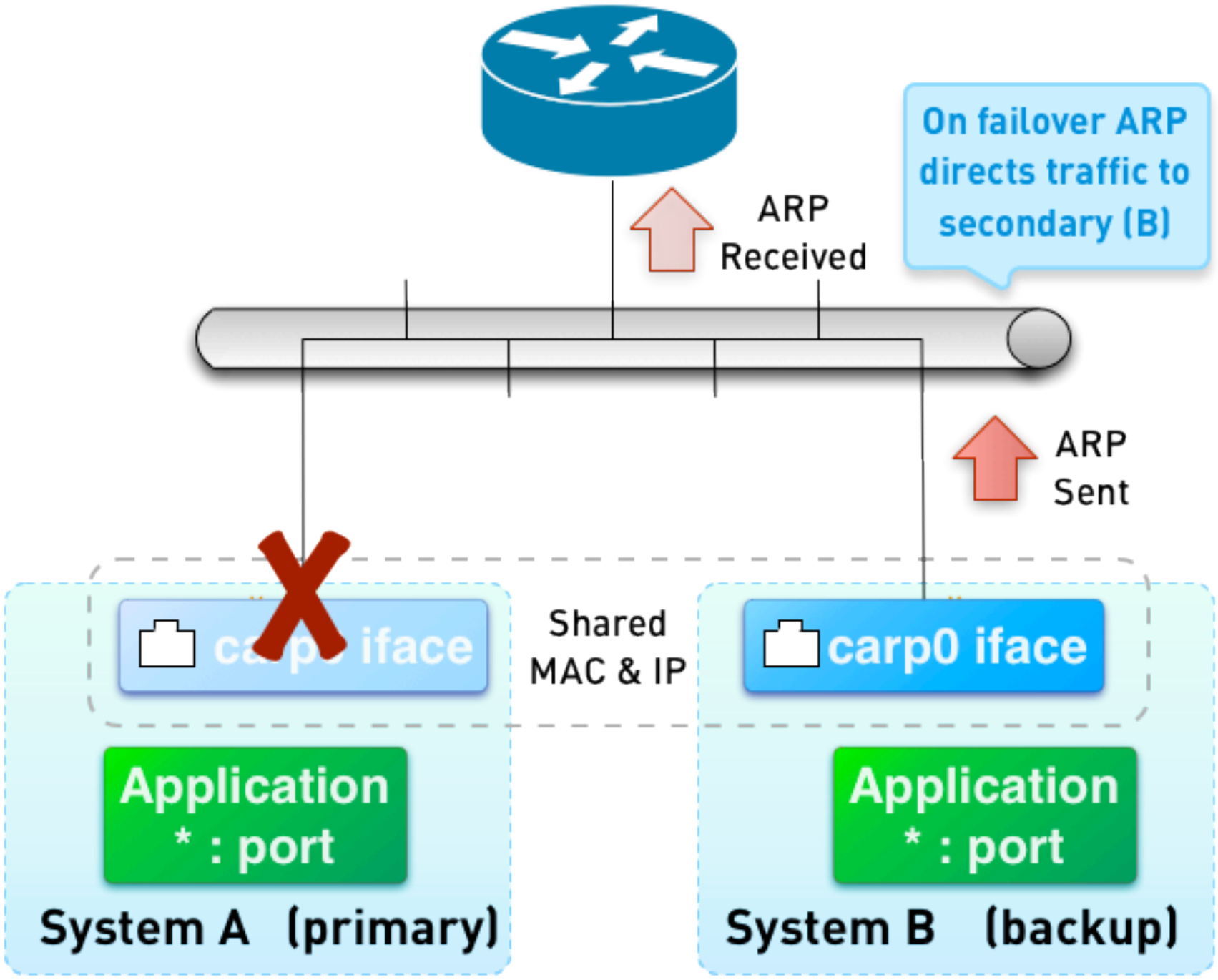
Worker Threads (e.g. Scheduler, Networking)

Database (MySQL)

Traditional HA Pair Failover



Traditional HA Pair Failover



cloudscaling

Fault Tolerance Methodologies

cloudscaling

Fault Tolerance in OCS

Service Distribution

High Availability Without Compromise

Resilient

Stateless

Scale-out

Service Distribution

Combines Standard Networking Technologies

OSPF	/etc/quagga/ospfd.conf	**router ospf** **ospf router-id 10.1.1.1** **network 10.1.255.1 area 0.0.0.0**
Anycast	/etc/quagga/zebra.conf	**interface lo:2** **description Pound listening address** **ip address 10.1.255.1/32**
Load-Balancing Proxy	/etc/pound/pound.conf	**ListenHTTP** **Address 10.1.255.1** **Port 8774** **xHTTP 1** **Service** **BackEnd** **Address 10.1.1.1** **Port 8774** **End** **BackEnd** **Address 10.1.1.2** **Port 8774** **End** **End** **End**

Resilient OpenStack

Horizontally Scalable, No Single Point Of Failure

Service Distribution

Service Endpoints (APIs)

ZeroMQ

Messaging System (RPC)

Service Distribution

Worker Threads (e.g. Scheduler, Networking)

MMR + HA

Database (MySQL)

Service Distribution Advantages

What Makes This a Superior Solution?

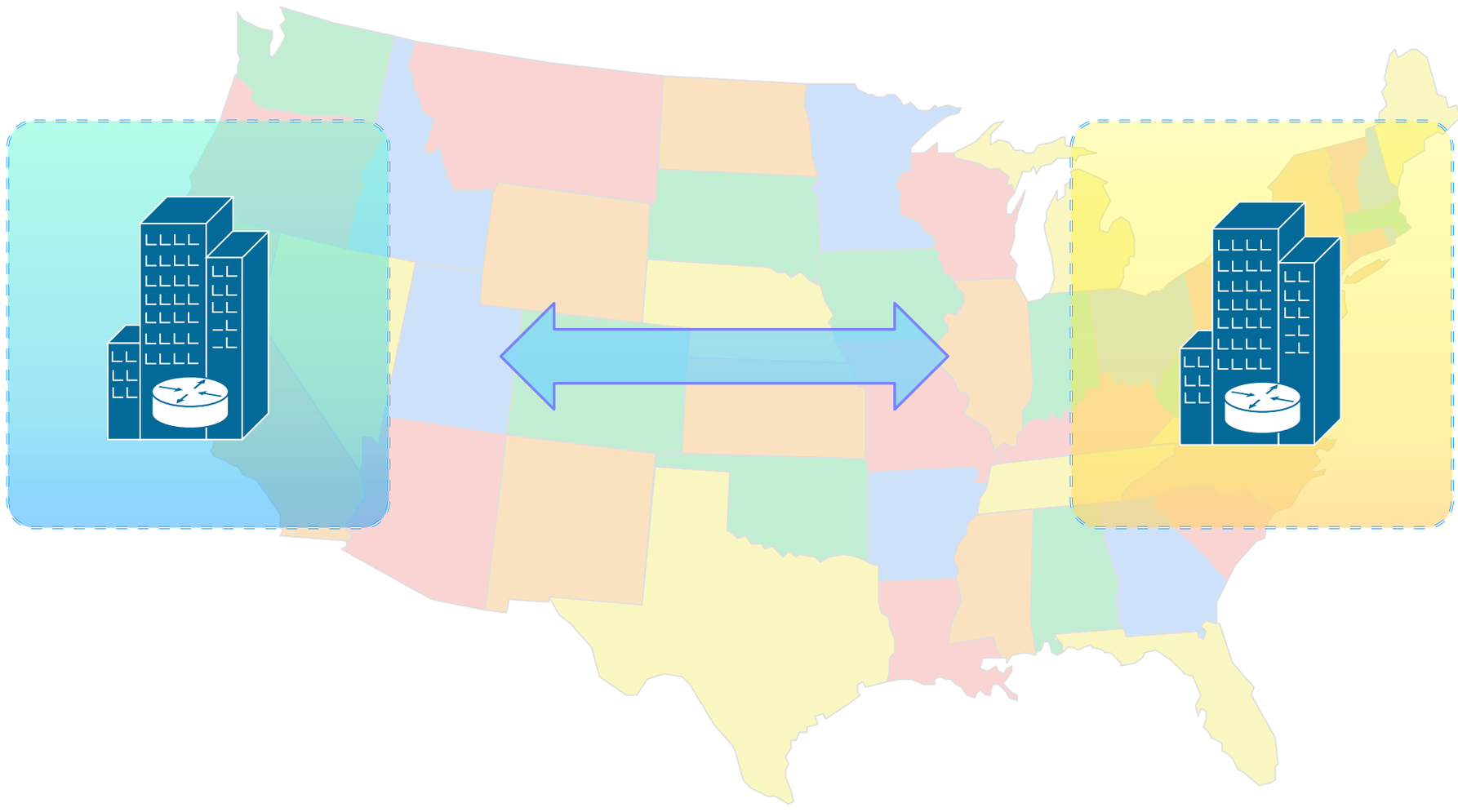
- True horizontal scalability with no centralized controller
- Services are always running, failover is nearly instant
- Reduced complexity, fewer idle resources
- No need for separate load balancers

Failover vs. Distributed Services

Site Failover and Global LB

Service Distribution Works With Multiple Sites

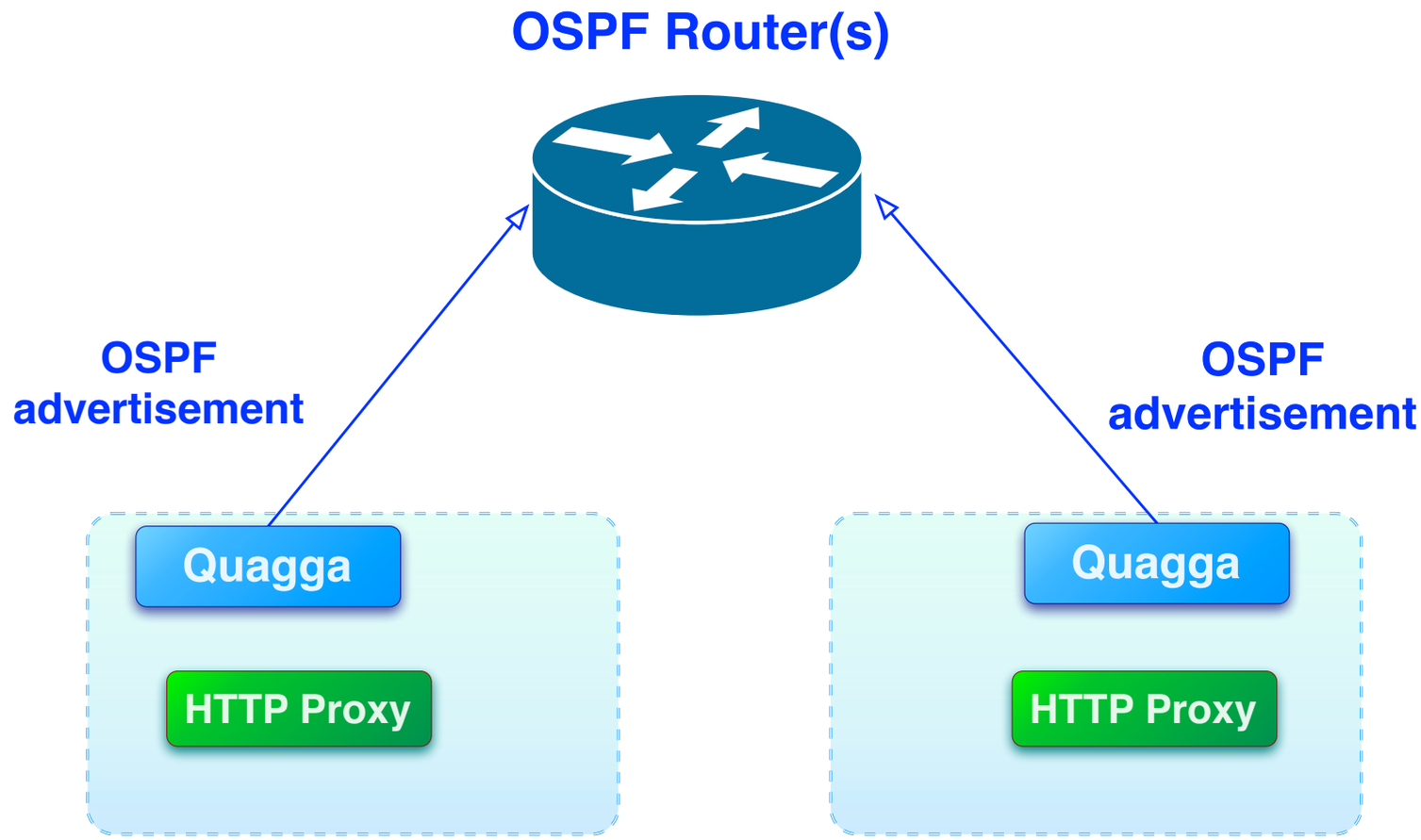
- Traditional HA pairs do not support cross-site resiliency
- Service Distribution fail across sites without DNS redirections



Service Distribution in Action

Example: Distributed Load Balancing

1) **OSPF**



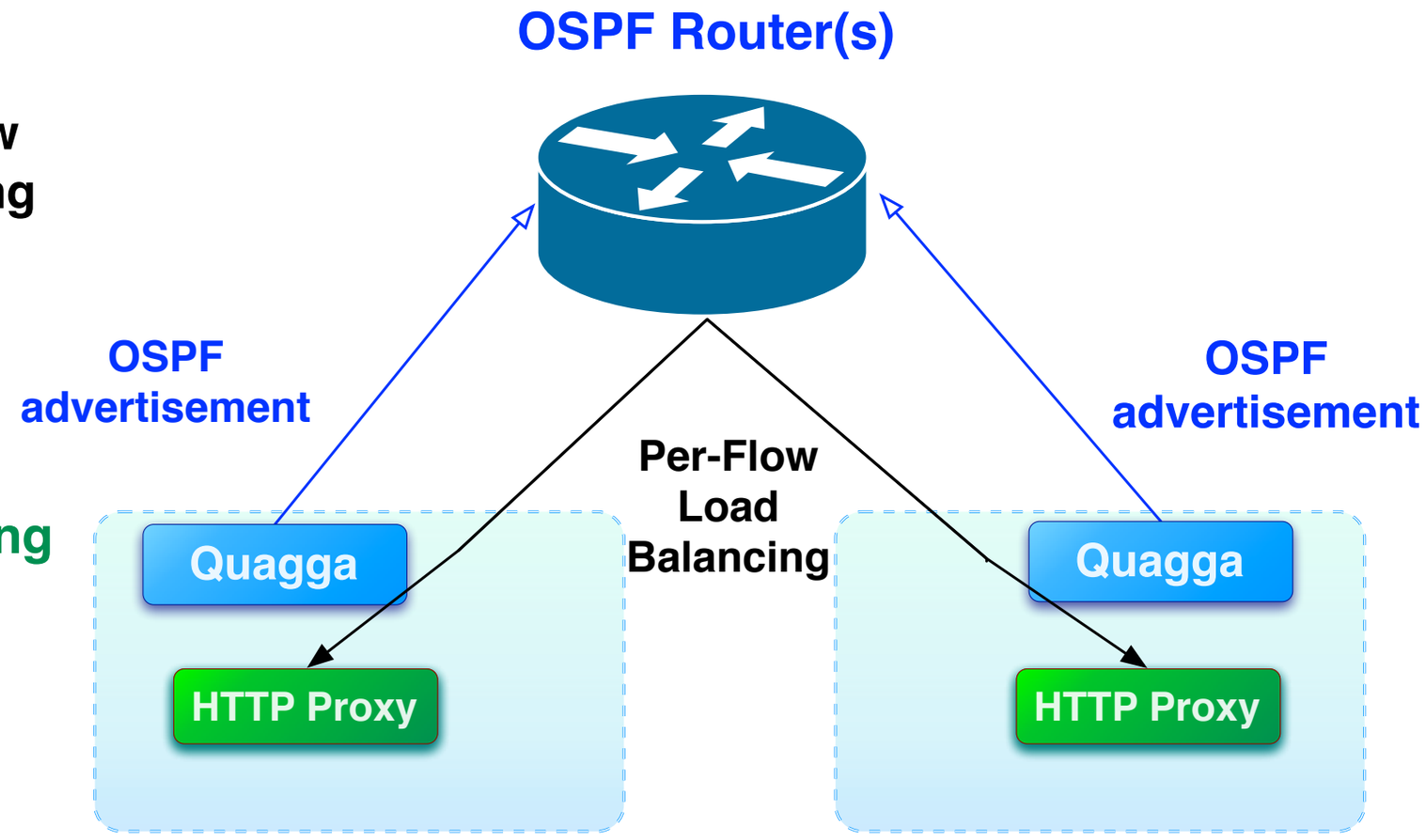
Service Distribution in Action

Example: Distributed Load Balancing

1) **OSPF**

2) **ECMP Per-flow Load Balancing**

3) **Load-balancing HTTP Proxy**



Service Distribution in Action

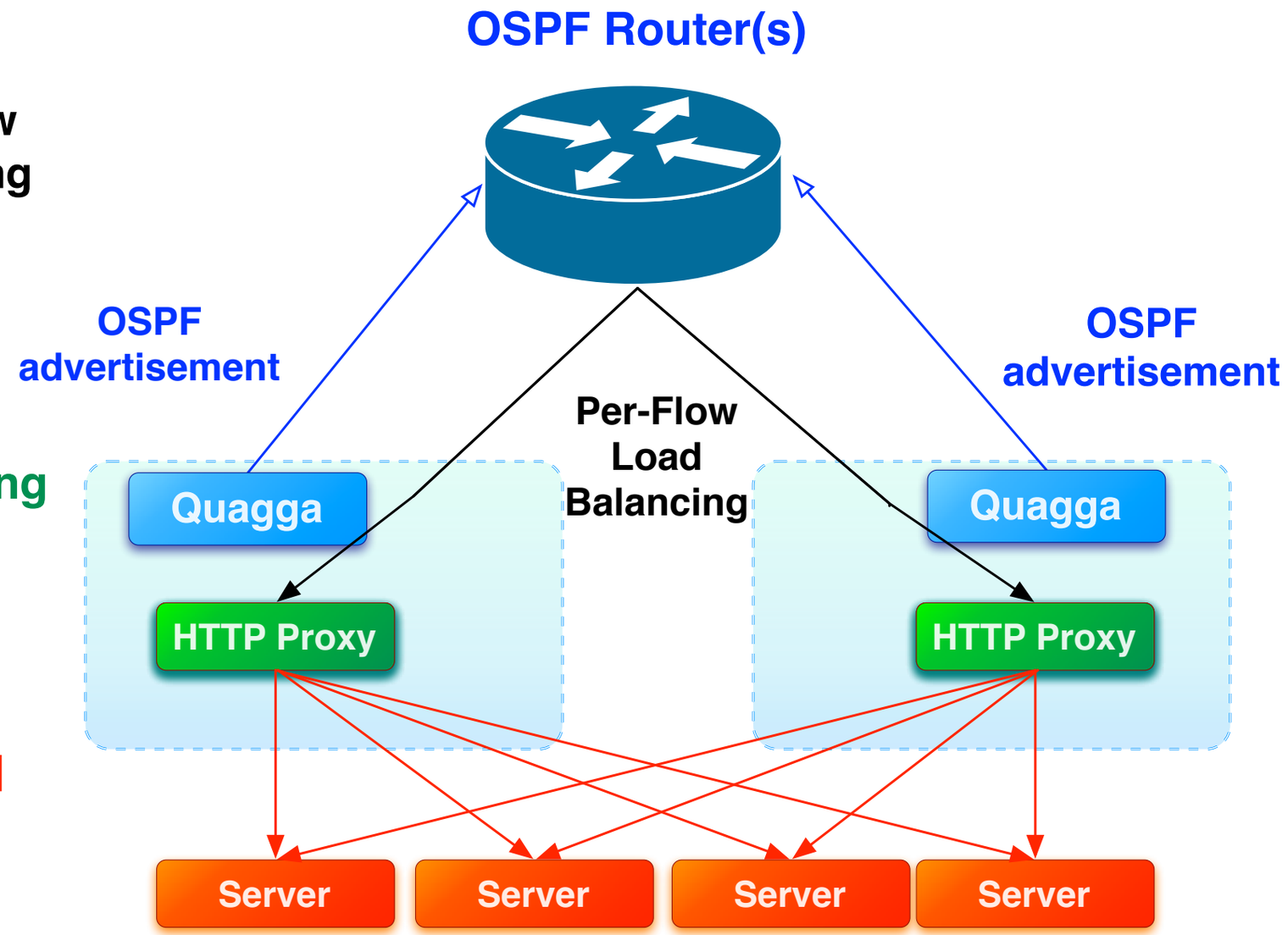
Example: Distributed Load Balancing

1) **OSPF**

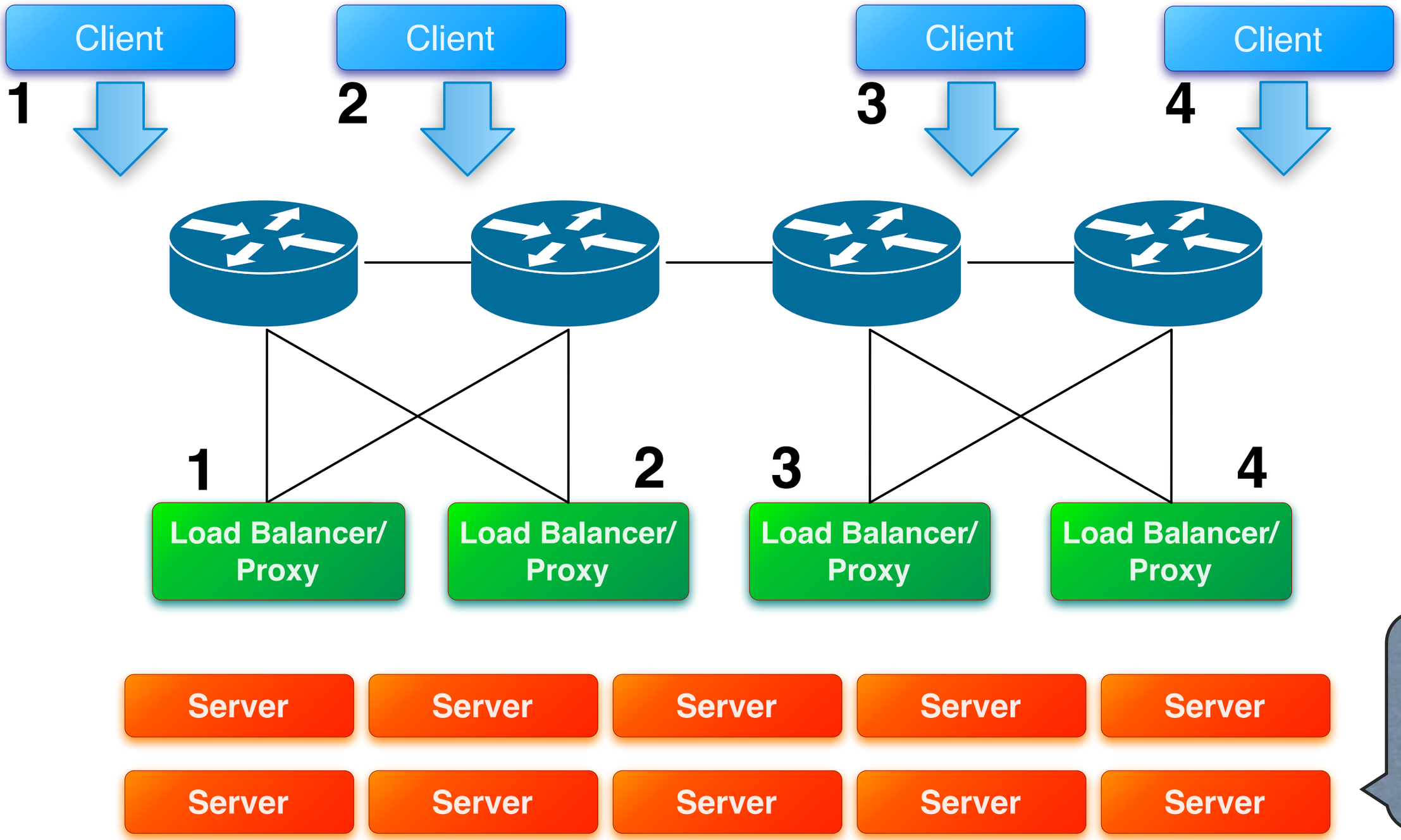
2) **ECMP Per-flow Load Balancing**

3) **Load-balancing HTTP Proxy**

4) **Unlimited # of Back-End Servers**

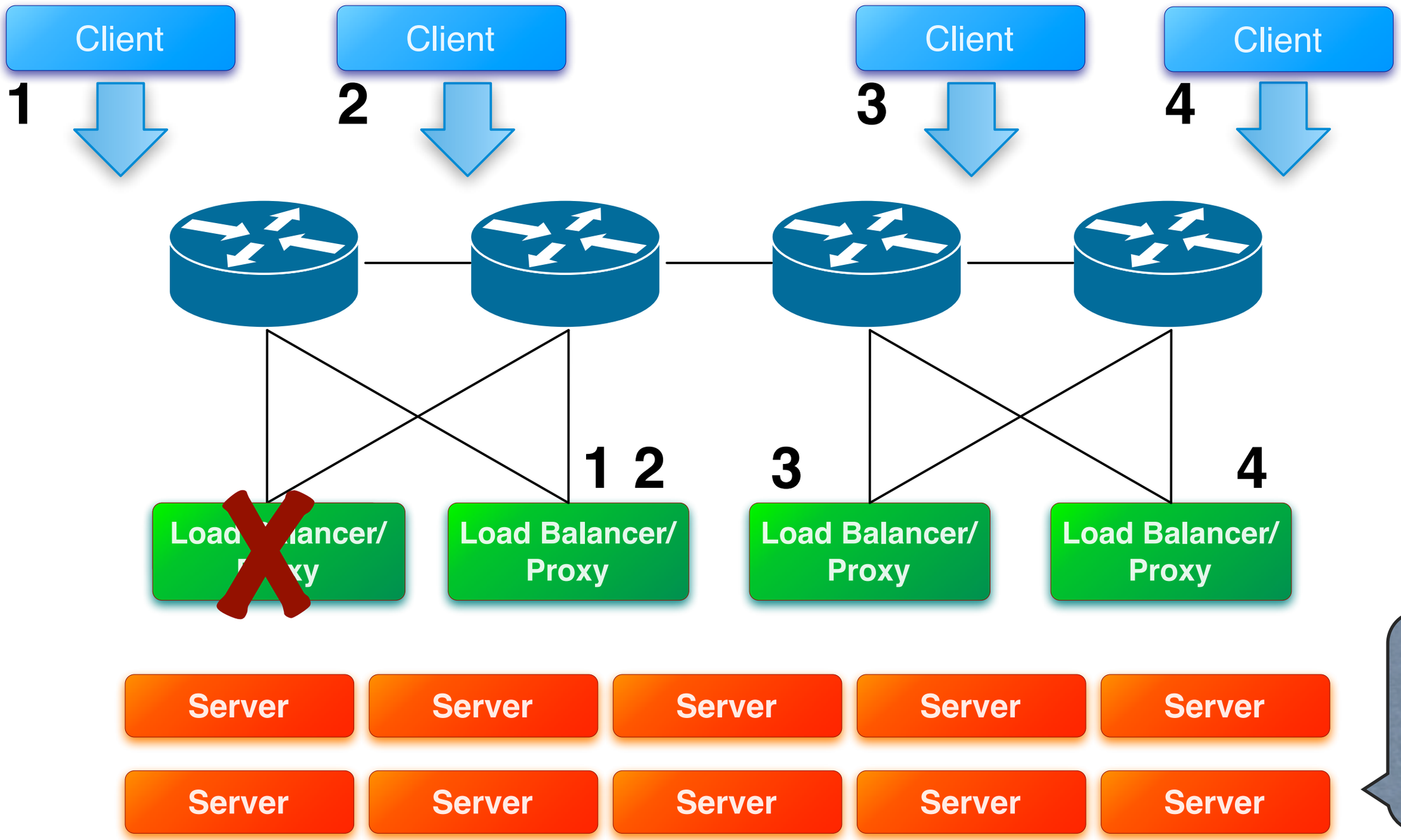


Failure Resiliency



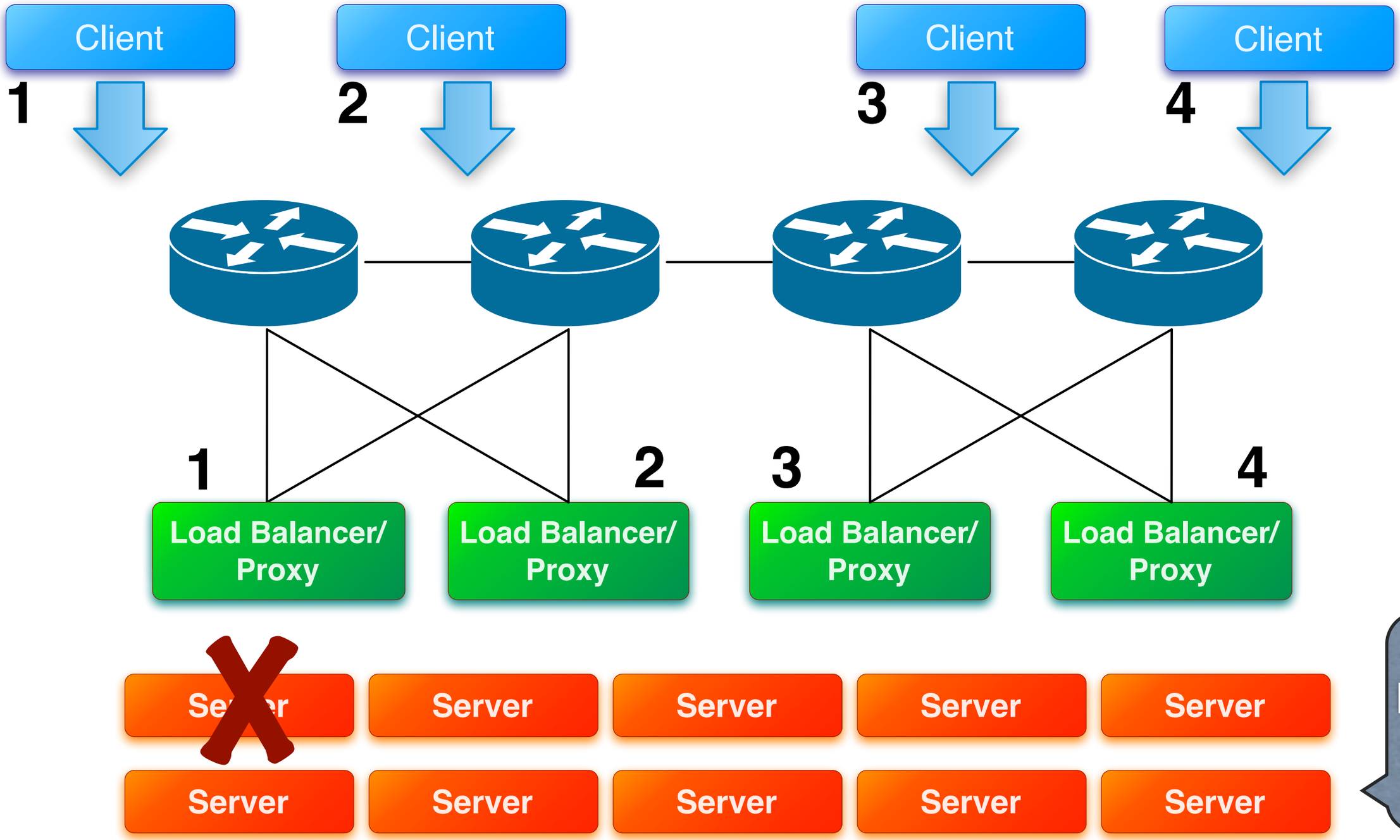
cloudscaling

Failure Resiliency



cloudscaling

Failure Resiliency



cloudscaling

cloudscaling