工業技術研究院
Industrial Technology Research Institute

Peregrine: An All-Layer-2 Container Computer Network

Tzi-cker Chiueh

Cloud Computing Research Center for Mobile Applications (CCMA)
雲端運算行動應用研究中心

Cloud Service Models

- Infrastructure as a Service (IaaS)
  - A set of virtual machines with storage space and external network bandwidth ➔ unfurnished apartment
  - Example: Amazon Web Service
- Platform as a Service (PaaS)
  - An operating environment including (application-specific) libraries and supporting services (DBMS, AAA) ➔ furnished apartment
  - Example: Google's App Engine, Microsoft's Azure, IBM's XaaS
- Software as a Service (SaaS)
  - Turn-key software hosted on the cloud and accessible through the browser ➔ hotel
  - Example: salesforce.com, and all major desktop software vendors

Data Center as a Computer

- Containerization
  - Optimal HW building block granularity or packaging
  - More efficient power distribution and thermal design
  - Unification of computing, memory, network and storage resources
    - Virtualization of all HW resources: Software-definable boundaries
  - Faster deployment: no on-premise installation needed
  - Requires light-out operation
- Google-style data center
  - Army of commodity HW
  - Treat failure as a common case

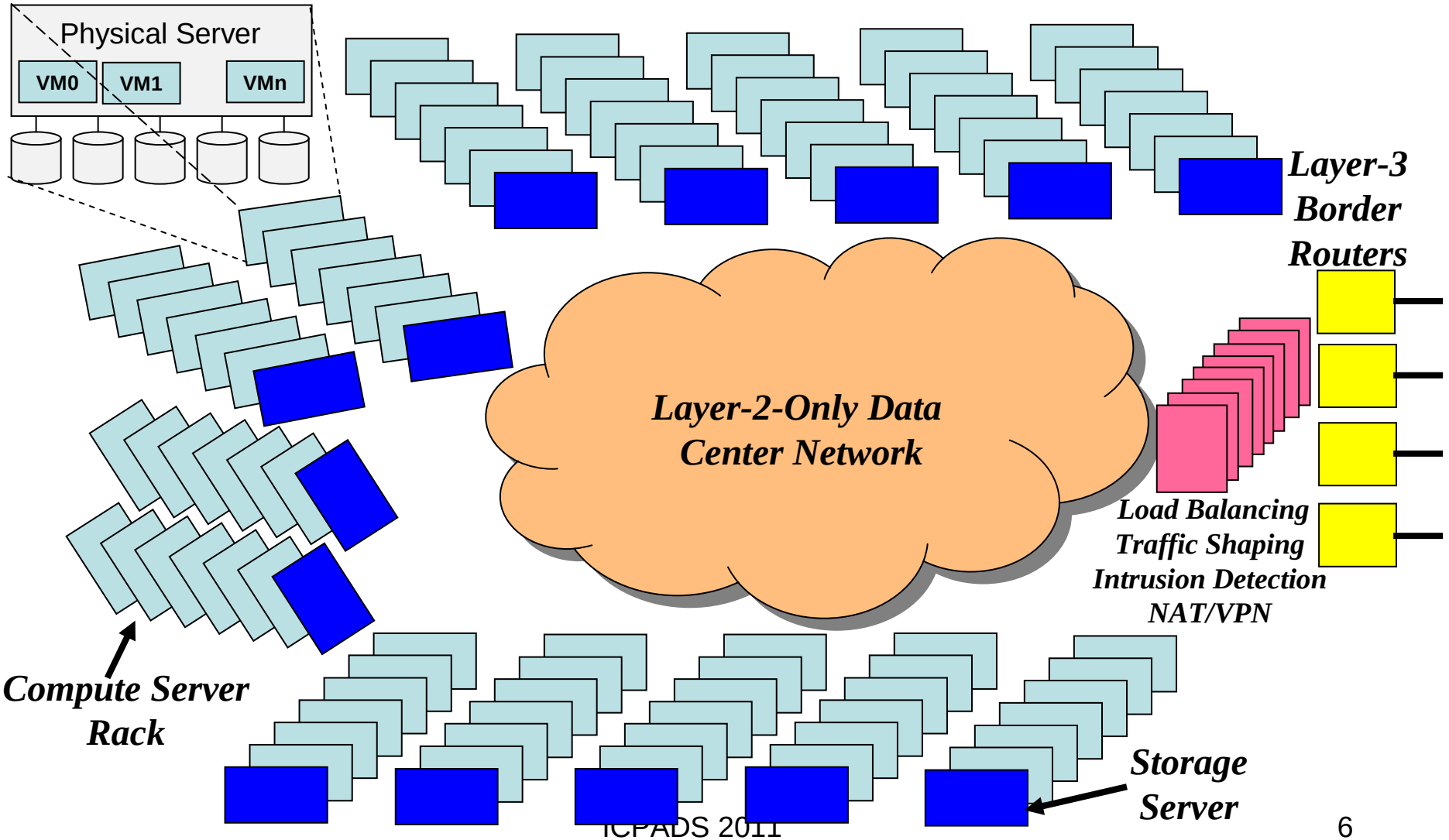
ITRI's Research Projects

- Container Computer 1.0
  - Manageable container computer for AWS-like IaaS
  - Differences between a set of servers/switches/storage boxes and a container computer?
    - Scalable storage/network architecture
    - Comprehensive monitoring and control
    - Energy-efficient cooling
- Cloud Operating System 1.0
  - Integrated data center software stack for supporting a AWS-like IaaS service on a set of commodity HW
  - Tight integration of storage, resource, security and system/network management

Cloud OS 1.0 Service Model

- Virtual data center consists of one or multiple virtual clusters, each of which comprises one or multiple VMs
  - Tiered architecture-based web services
- Users provide a Virtual Cluster specification
  - No. of VM instances each with CPU performance and memory size requirement
  - Per-VM storage space requirement
  - External network bandwidth requirement
  - Security policy
  - Backup policy
  - Load balancing policy
  - Network configuration, e.g. public IP address and private IP address range
  - OS image and application image

Container Computer 1.0 Architecture



Cloud-Scale Data Center

- Big
  - 5000 servers and up
  - HW failures are inevitable
- Shared
  - Virtualization: multiple virtual data centers running on a single physical data center
  - State isolation
    - Name space reuse
    - Visibility control
  - Performance isolation
    - Service level agreement (SLA) or Quality of service (QoS)

Design Issues of Cloud Data Center Network

- Scalable and available data center fabrics
- Internet Edge Logic:
  - Server load balancing
  - Multi-homing load balancing
  - Traffic shaping or Internet QoS guarantee
  - WAN traffic compression and caching
- Network support for hybrid cloud
- Rack area networking for I/O device consolidation and sharing

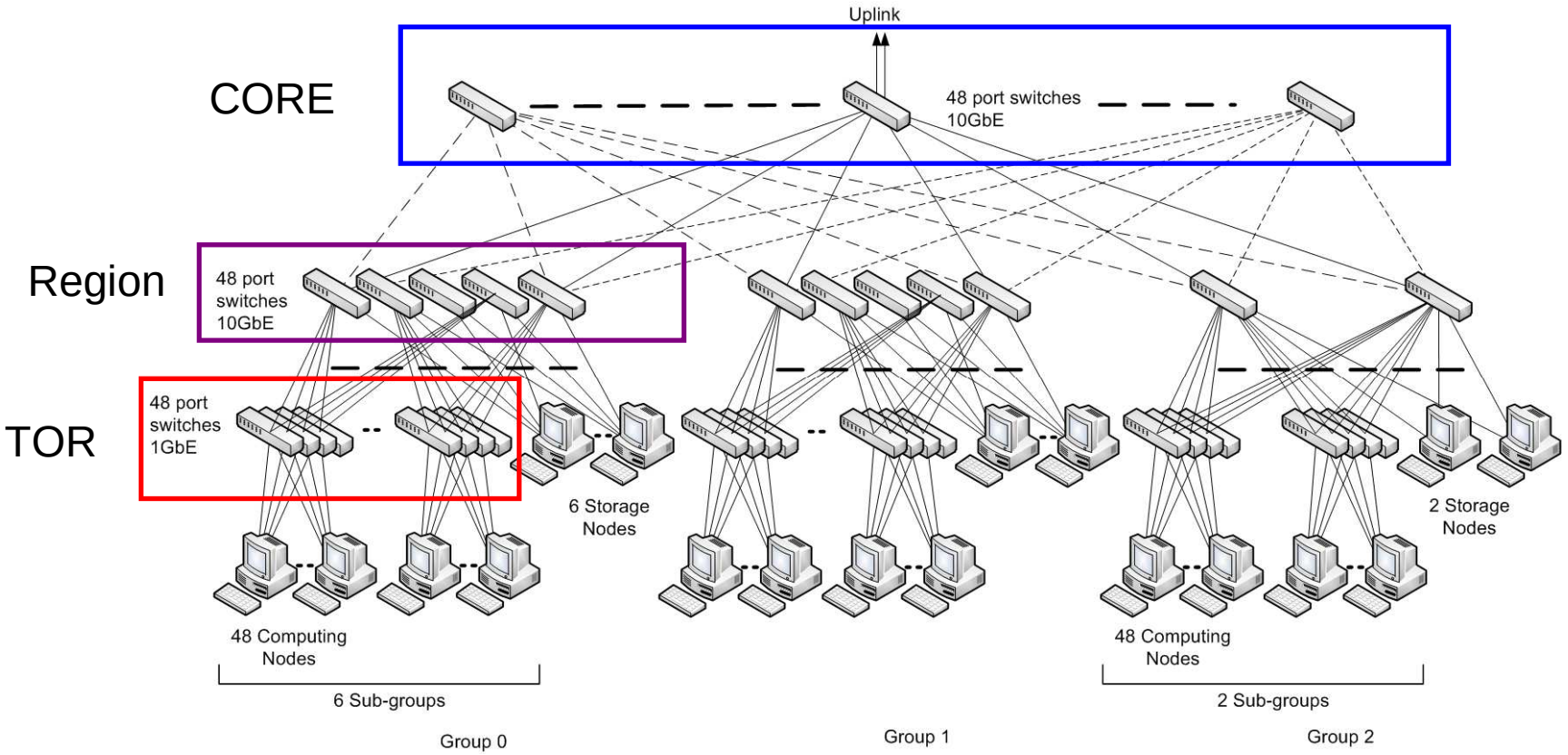
What's Wrong with Ethernet?

- Spanning tree-based
  - Not all physical links are used
  - No load-sensitive dynamic routing
  - Fail-over latency is high (> 5 seconds)
- Cannot scale to a large number of VMs (e.g. 1M)
  - Forwarding table is too small: 16K to 64K
- Does not support VM migration and visibility
- Does not support private IP address space reuse
- Limits scope of VM migration

Peregrine

- A unified network for LAN and SAN
- Layer-2 only
- Centralized control plane and distributed data plane
  - Asymmetric routing
- Commodity Ethernet switches only
  - Army of commodity switches vs. few high-port-density switches
  - OpenFlow is good to have but not needed
- No broadcast, flooding, source learning, access control list, etc.
- Careful architecting of all Internet Edge Logic

Peregrine's Clos Network Topology

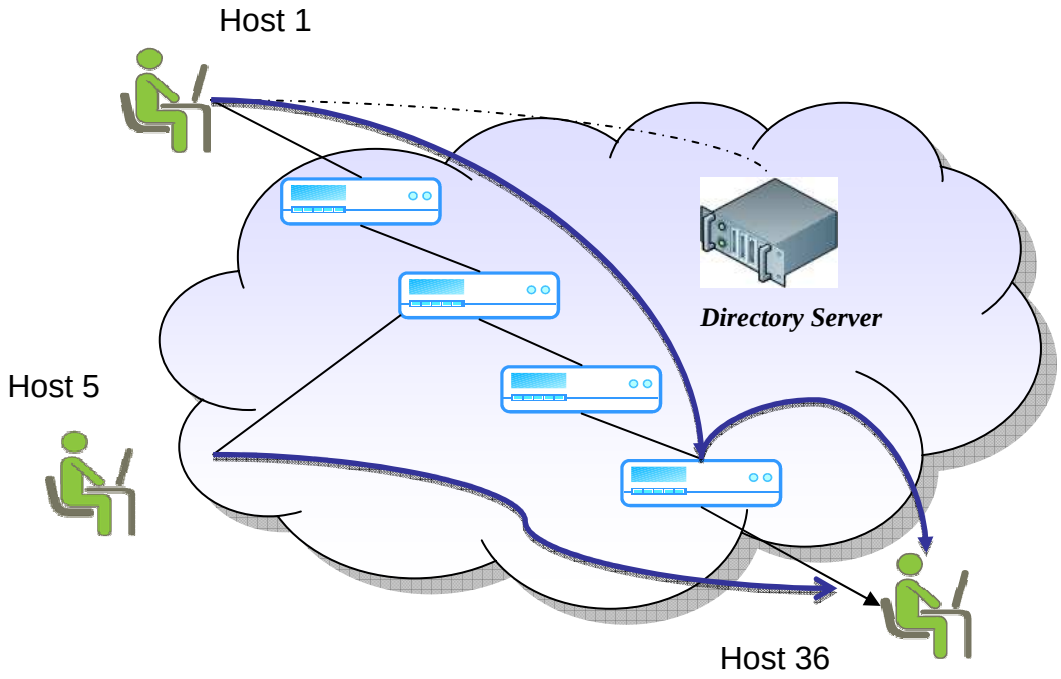


Scaling up to 1M VMs

- Problem: small forwarding table (< 64K)
- Solution: Two-stage forwarding
  - Source → Intermediate → Destination
- Problem: two-stage forwarding limits scalability and introduces latency penalty
- Solution: Dual-mode forwarding
  - Direct: source → destination for heavy-traffic pairs
  - Indirect: source → intermediate → destination for the rest

Two-Stage Forwarding

- Every Intermediate knows how to route to every VM in its scope
  - Intermediate needs to be notified when VM leaves or joins its scope
- Source → Intermediate → Destination
  - Intermediate: TOR_Swicth(Dest) or Physical_Machine (Dest)
- Directory Server: Host → Intermediate(Host) map

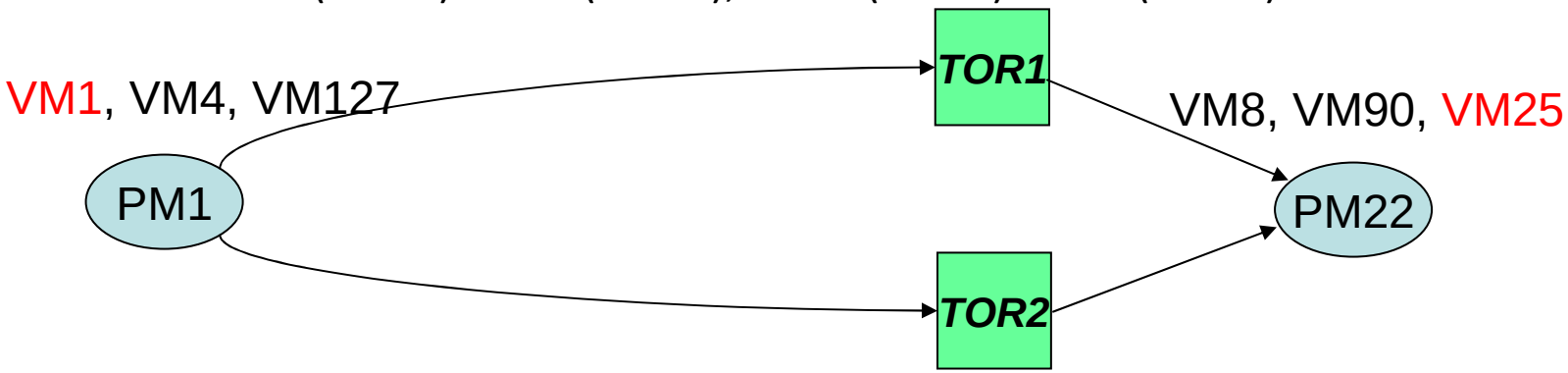


Fast Fail-Over

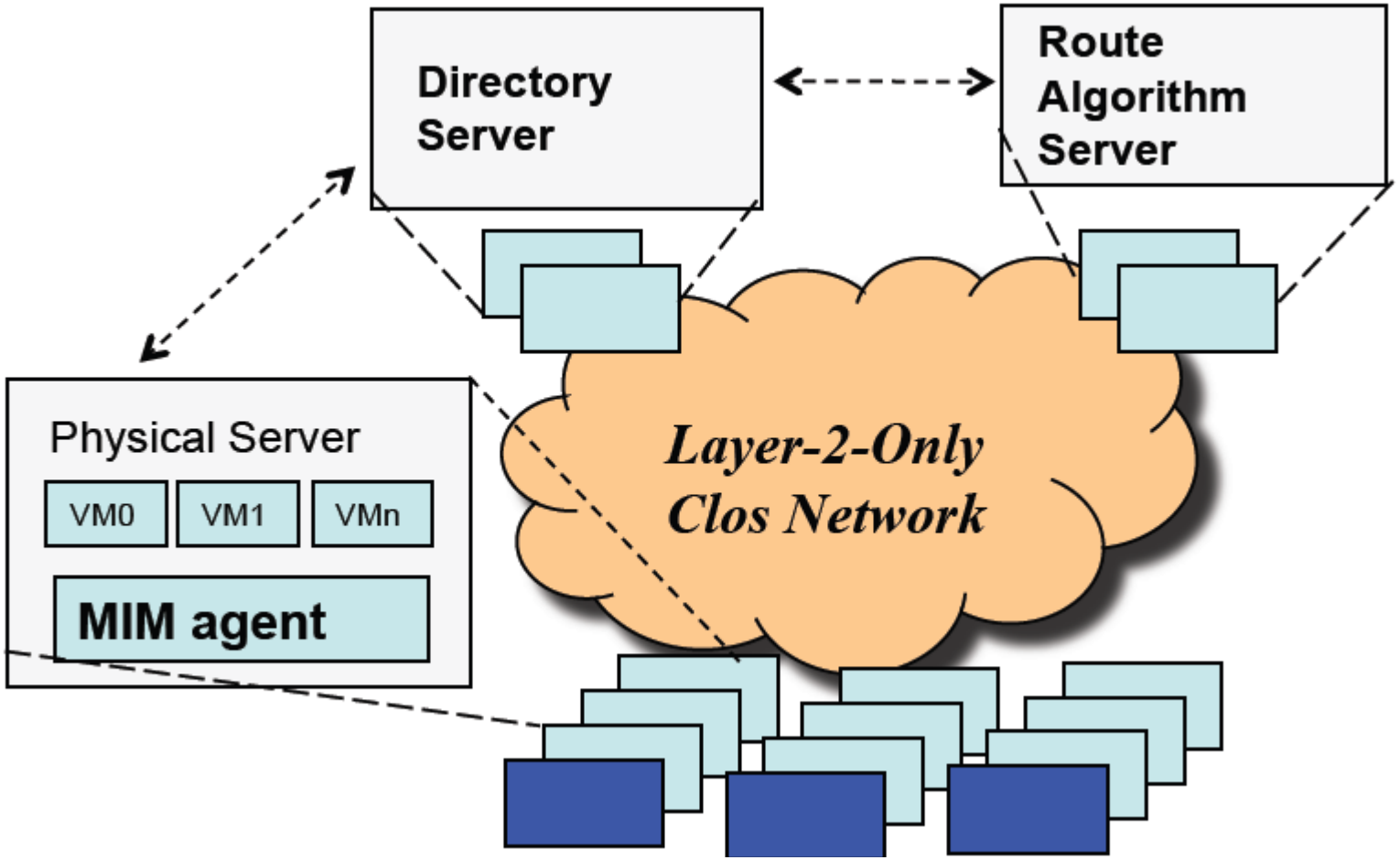
- Goal: Fail-over latency < 100 msec
- Strategy: Pre-compute a primary and backup route for each VM
  - Each VM has two virtual MACs
  - When a link fails, identify all affected routes
  - Notify hosts using affected primary routes that they should switch to corresponding backup routes
  - Re-compute new alternative routes

Interaction with Fail-Over Mechanism

- For each physical node P, routing algorithm computes two disjoint spanning trees, which enable other physical nodes to reach P
  - Direct routing: MAC1(VM25), MAC2(VM25)
  - Indirect routing:
    - MAC1(TOR1):MAC1(VM25); MAC1(TOR2):MAC2(VM25)
    - MAC1(PM22):MAC1(VM25); MAC2(PM22):MAC2(VM25)



Software Architecture



Directory Server (DS)

- Perform a generalized ARP query service
- A generalized ARP (GARP) map between a VM's IP address and
  - Primary MAC address and an availability bit
  - Backup MAC address and an availability bit
  - Primary intermediary's MAC address and an availability bit
  - Backup intermediary's MAC address and an availability bit
- Each GARP map entry keeps a list of caching clients and their expiration time
- Directory clients cache GARP entries using a lease-based cache consistency protocol

Route Algorithm Server (RAS)

- Collects VM migration, network element failure and congestion events
- Collects traffic matrix information at run time
- Includes a route engine that computes statically or incrementally routes for VM pairs
- Builds an inverse map that associates network links with all computed routes that travel on them

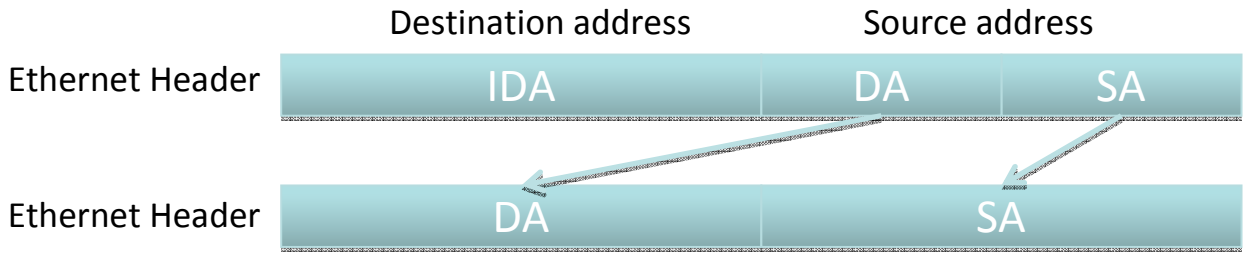
Mac-In-Mac (MIM) Kernel Agent

- Packet encapsulation and de-capsulation
- Caching GARP responses
- Intercepts ARP queries from VMs and services them by using GARP cache or contacting DS
- Check consistency between destination IP address and destination MAC address for each packet
- Collecting traffic matrix

Encapsulation and Decapsulation

- Encapsulation: Place DA and SA in the Ethernet source address field

- Decapsulation: extract DA and put it in the Ethernet destination address field



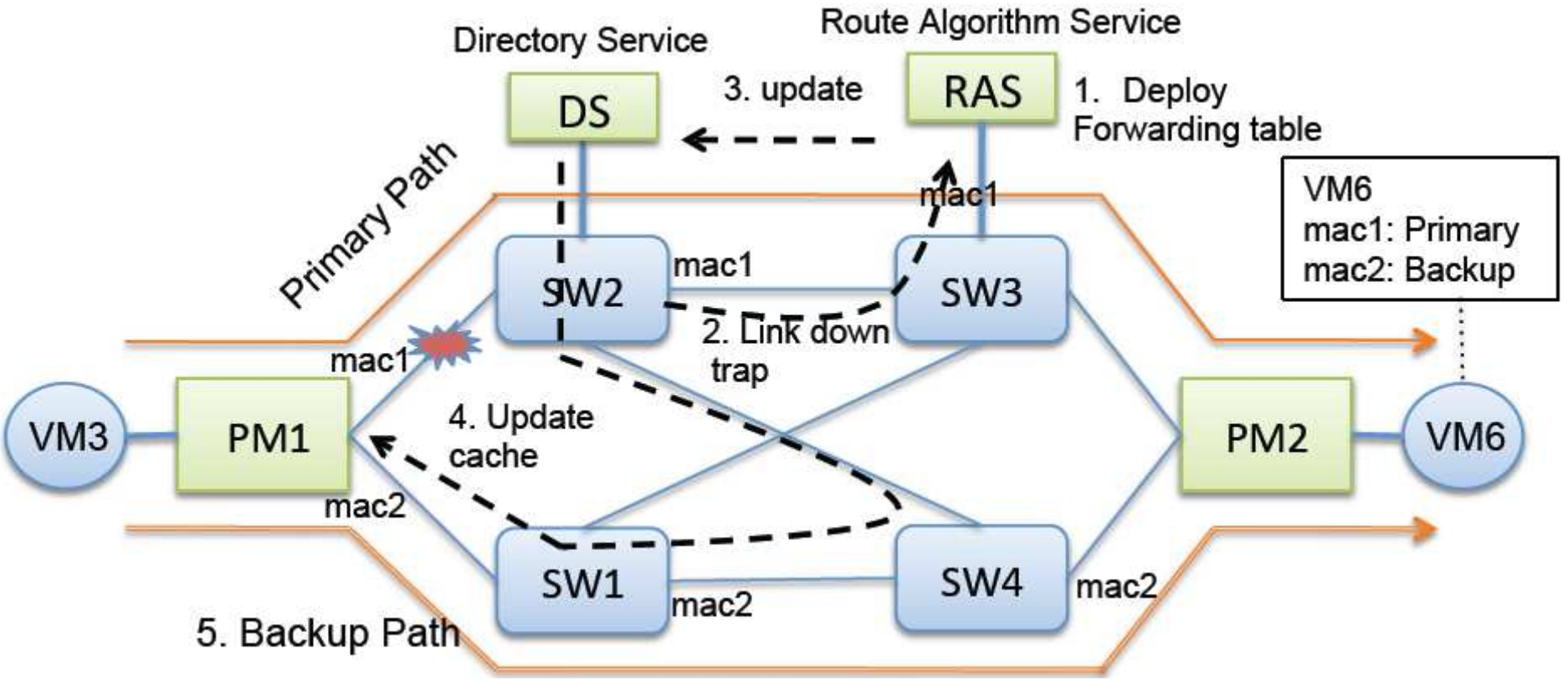
- Each VM's and PM's MAC address is effectively only 3 bytes long (as opposed to 6 bytes long)
- The most significant half of the Ethernet source address field signifies whether it is "encapsulated"

SA: Source address, DA: Destination address
IDA: Intermediate Destination address

Private IP Address Space Reuse

- Private IP Address Space Reuse (PASR)
  - Private IP address space is partitioned into a service node range and a non-service node range
  - The service node range is shared by all VDCs
  - The non-service node range could be resued across VDCs
  - Key used for GARP query: VDC ID + IP Address
    - The same IP address may be resolved into a different MAC address depending on its VDC ID
- Enforces inter-VDC isolation by checking outgoing packet's destination MAC matches its destination IP address
  - VDC: process
  - IP address → MAC address: Virtual address → Physical Address

When a Network Link Fails



Fail-Over Practice

- Witness switches ➔ RAS ➔ DS ➔ Affected VMs
- Witness switches ➔ RAS: send multiple SNMP traps to multiple SBMP managers
- RAS ➔ DS: RAS and DS are connected by dedicated links
- DS ➔ Affected VMs: use backup routes if necessary
- HA mechanism for RAS and DS
- Invalidate retired forwarding table entries
- Compute new alternative routes

When a VM X Moves

- Notifies RAS, which in turn notifies DS
- DS modifies X's GARP entry to reflect the MAC addresses of its new primary and backup intermediary
- DS invalidates X's GARP entry cached on all other VMs that communicate with it
- DS invalidates (asynchronously) all direct forwarding entries of this VM on the network

Centralized Load-Balancing Routing

- Ideal load-balancing routing algorithm
  - Compute the M shortest routes for every <u,v>
  - Distribute the traffic load of <u,v> among these M routes
  - Calculate the expected load of every physical network link
  - Weight of a link: EL(link)/RC(link)
- Approximate load-balancing routing algorithm
  1. Sort traffic matrix entries in decreasing order
  2. For each <u,v>, use weighted shortest path algorithm to find its primary and backup route
     - Only top N heavy-traffic node pairs accounting for Z% of all traffic
  3. Update the weights of links associated with <u,v> and its chosen path; go to 1

Internet Edge Logic

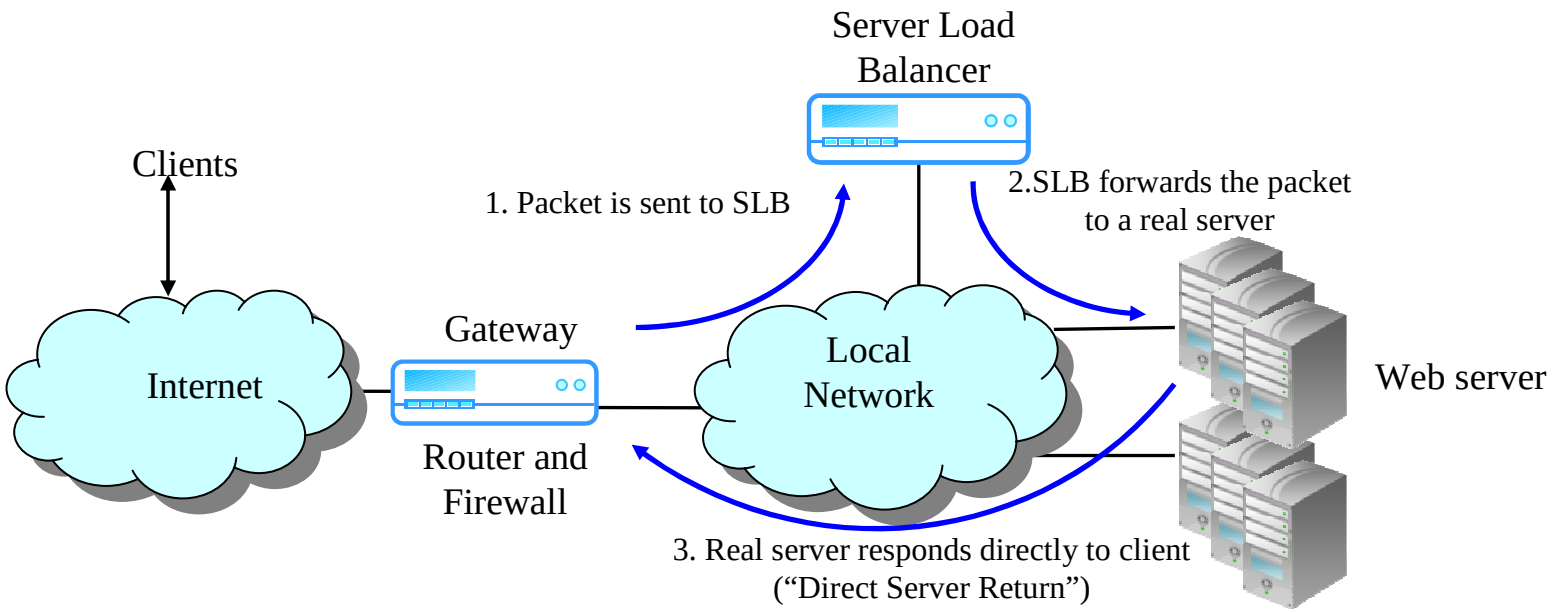
- Cluster-based implementation
- Server load balancing
- Firewalling and IDS/IPS
- Network Address Translation
- Multi-homing load balancing (Cloud OS 2.0)
- Internet traffic shaping (Cloud OS 2.0)
- VPN for hybrid cloud (Cloud OS 2.0)
- WAN traffic caching and compression (Cloud OS 2.0)

Server Load Balancer in Cloud OS 1.0

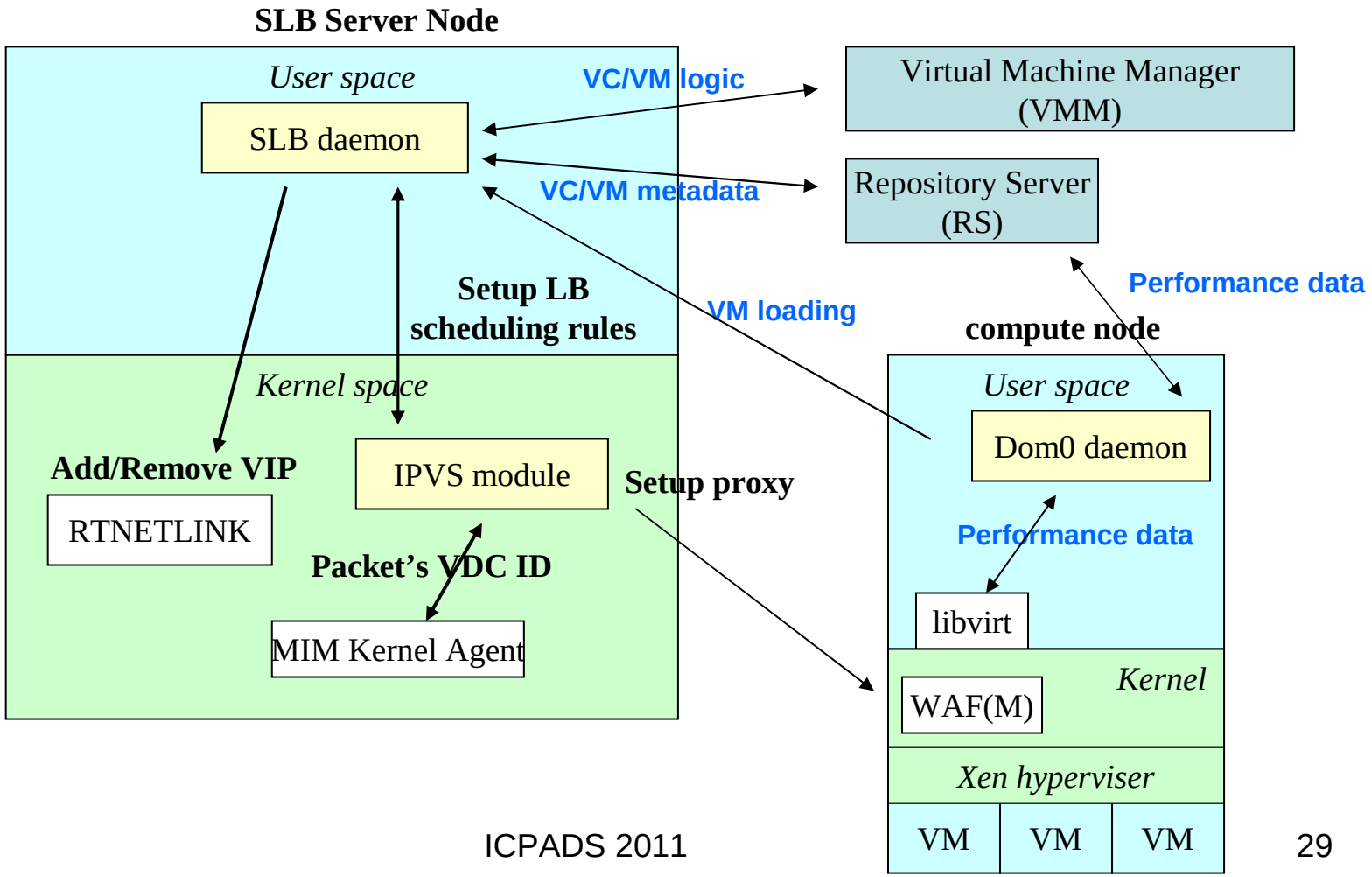
- Goal: distribution of input requests among virtual machines in an Internet-facing virtual cluster
- Layer 4 request distribution for services identified by TCP and UDP port numbers based on network and CPU load
- Cluster implementation with HA, load balancing and auto-scheduling
- Sticky session: source IP based
- Direct server return
- Support for auto-scheduling of virtual cluster

Direct Return of Response Traffic

- Intra-VC inter-VM load balancing
- Direct Server Return (DSR) reduces traffic by allowing web servers to send HTTP responses directly back to the requesting client

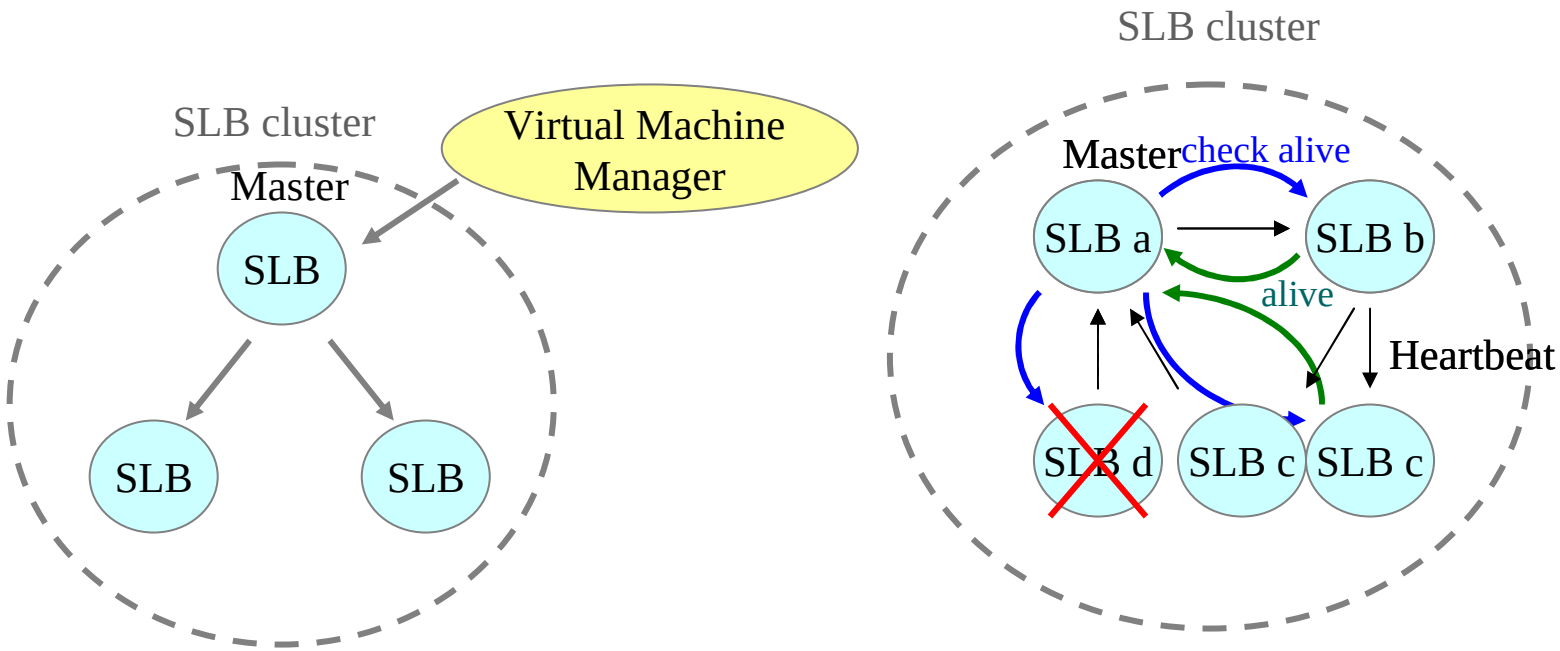


Software Architecture



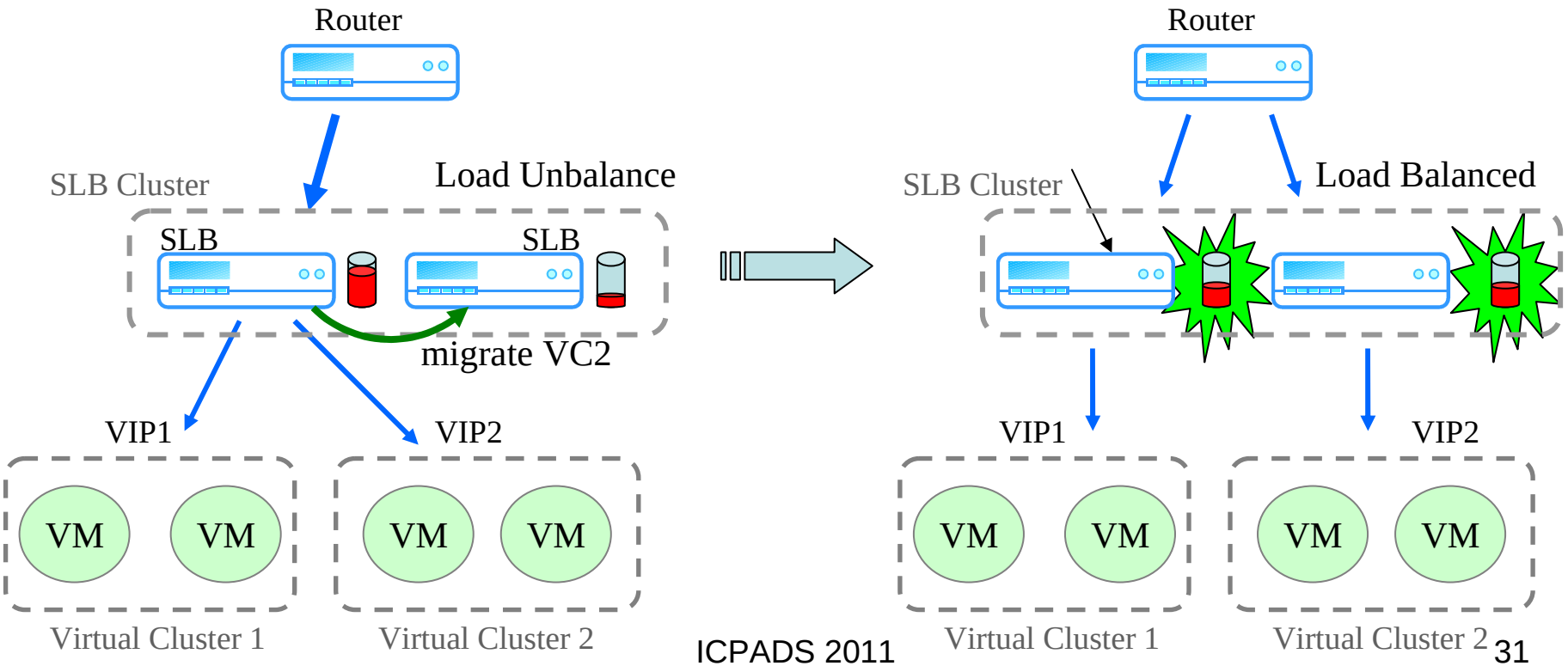
SLB Fault Tolerance

- Active-active HA rather than active-passive HA
- Ring structure SLB with each one monitored by its neighbor
- Master SLB synchronizes VC information to other SLB servers



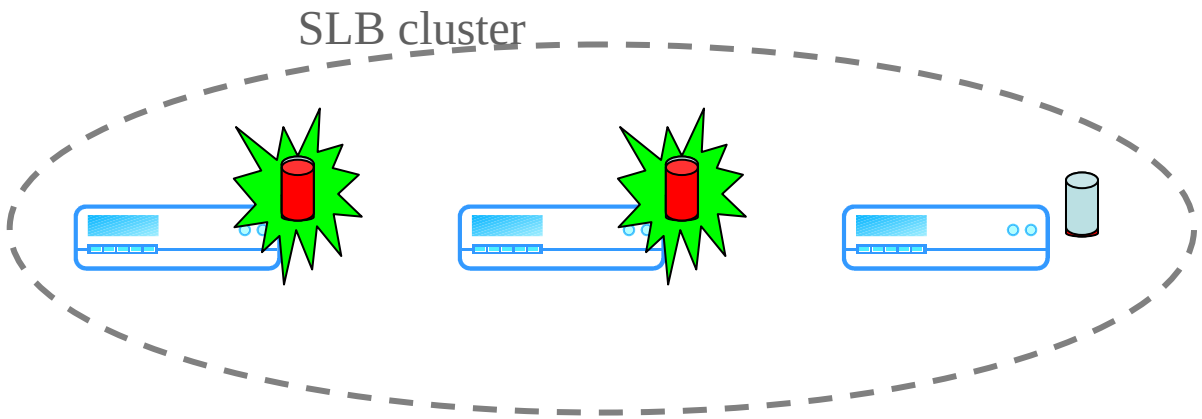
SLB Load Balancing

- Master SLB server periodically monitors each SLB server's load and balances the loads among SLB servers, by re-distributing VC request dispatching responsibilities among SLB servers



SLB Auto Scaling

- Add new SLB node:
  - When the average load of a SLB cluster is above a high watermark, the master SLB server asks PRM to power on one more SLB server
- Remove a SLB node:
  - When the average load of a SLB cluster is under a low watermark, the master SLB server asks PRM to power off one SLB server to save power



On-Going Work

- Performance Isolation between storage access traffic and application traffic
  - QoS-aware routing
- Scalability concerns:
  - Coverage of intermediate proxy
  - GARP query processing
  - GARP cache invalidations upon network failure or VM migration
    - Layer-2 broadcast/multicast support
  - Incremental routing

Conclusions

- Cloud data center network issues
  - All-L2 data center backbone (e.g. TRILL and SPB)
  - Internet edge logic
  - Rack area networking
  - Hybrid network support
- Existing solutions are fragmented or incomplete
- Plenty of room for innovation for a fully integrated data center network solution
- Current Status of Peregrine: 100-node testing is done, is going to move it to a 500-node container computer

Thank You!

Questions and Comments?

tcc@itri.org.tw