工業技術研究院
Industrial Technology Research Institute

Network Design Issues for Cloud Data Centers

Tzi-cker Chiueh **(闕志克)**

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

Cloud-Scale Data Center

- Main building blocks for Cloud Computing industry
- Technology components:
  - Modular cloud computer: Optimal HW building block for constructing a cloud data center
  - Cloud OS: An end-to-end software stack that runs cloud applications and operates a cloud data center
  - Non-ICT technology: seismic, fire, physical security, etc.
  - Integration/operation know-how: Operational experiences and expertise for putting together and running a cloud-scale data center

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
  - 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

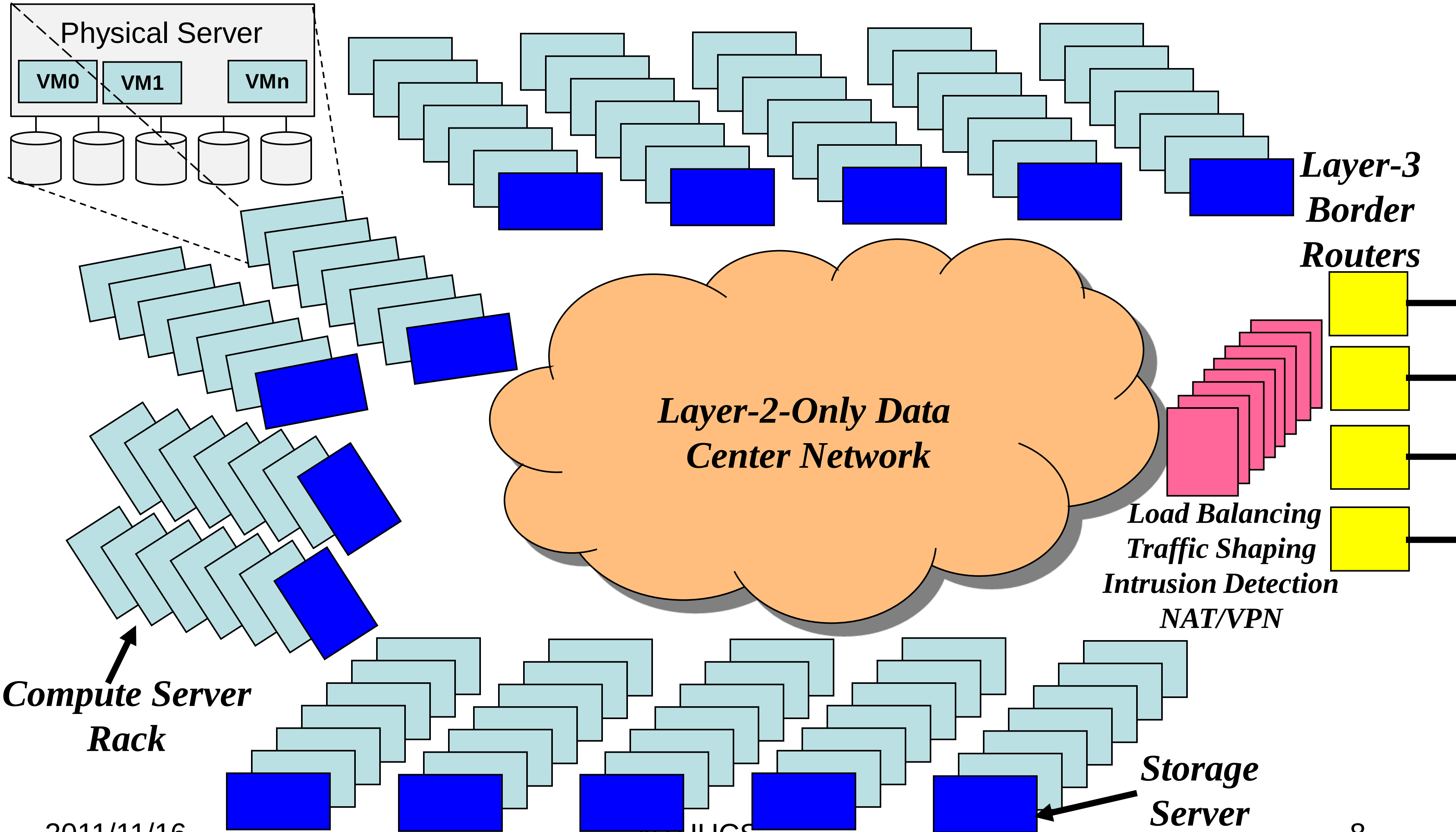
Cloud OS 1.0 Service Model

- 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

- Objective: Physical data center in a box
- Architecture Design Principles:
  - Commodity HW only
    - No storage box, appliance or accelerator
  - System-wide optimization
    - Component vs. self-contained system
    - server → container computer → warehouse computer
  - End-to-end redundancy
    - No HW element is indispensible
- Major features:
  - All-layer-2 data center network architecture
  - Scalable Internet edge appliance functionality
  - Touch cooling-based thermal management
  - Light-out management

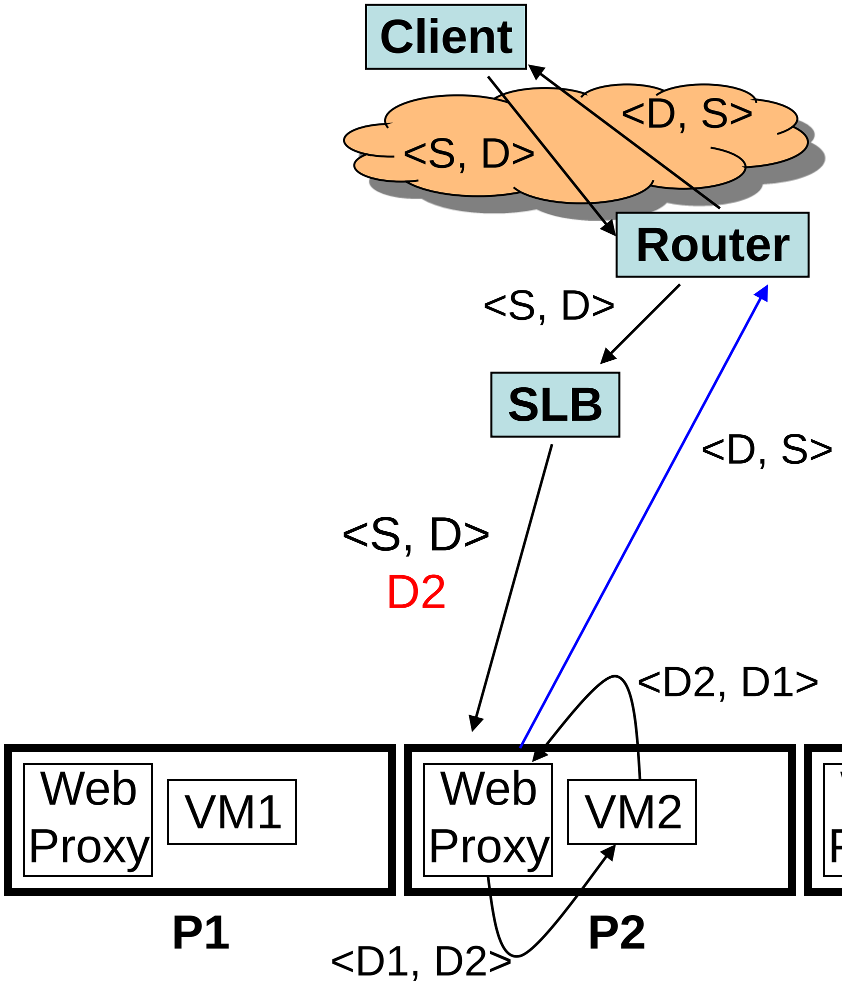
Container Computer 1.0 Architecture



Cloud Network Design Issues

- Internet Appliance Logic:
  - Server load balancing
  - Multi-homing load balancing
  - Traffic shaping or Internet QoS guarantee
  - WAN traffic compression and caching
- Network support for hybrid cloud
- “PCI bus” for data center computer
- Rack area networking for I/O device consolidation and sharing

NAT Support for Server Load Balancing



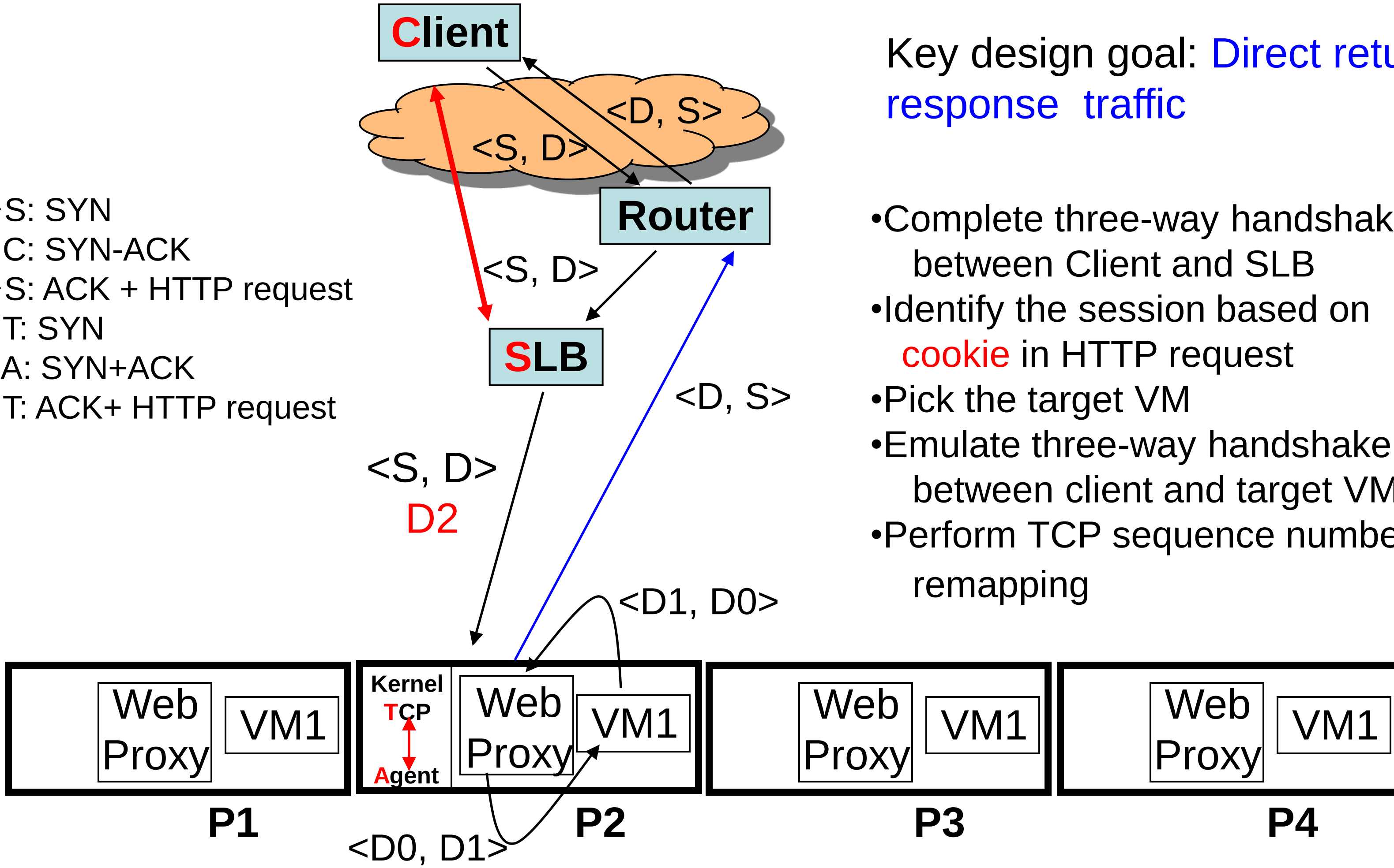
Server Load Balancing = Intra-VDC inter-VM Load Balancing

Key design goal: Direct return of response traffic

NAT Support for Session-Aware Server Load Balancing

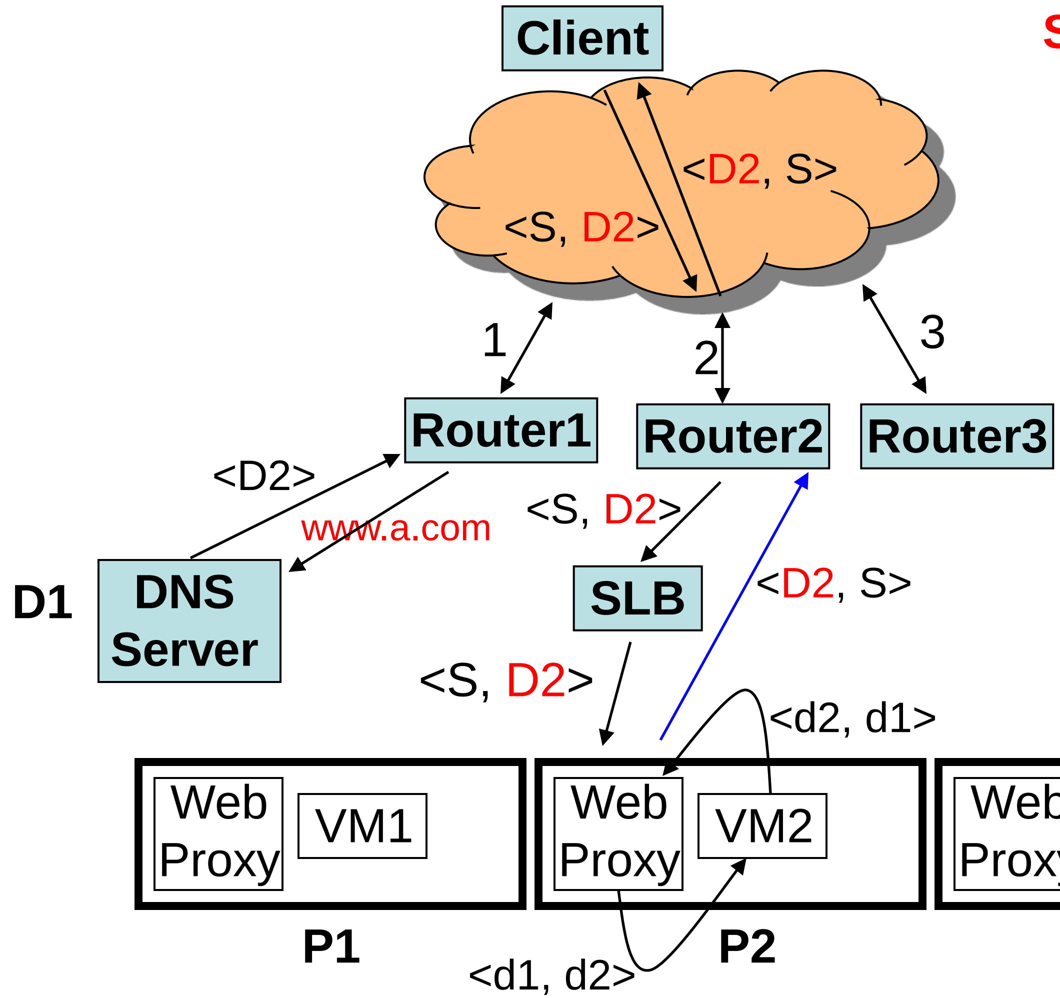
Key design goal: Direct return of response traffic

C→S: SYN
S→C: SYN-ACK
C→S: ACK + HTTP request
A→T: SYN
T→A: SYN+ACK
S→T: ACK+ HTTP request



- Complete three-way handshake between Client and SLB
- Identify the session based on cookie in HTTP request
- Pick the target VM
- Emulate three-way handshake between client and target VM
- Perform TCP sequence number remapping

Multi-Homing Load Balancing – Externally Initiated Connections

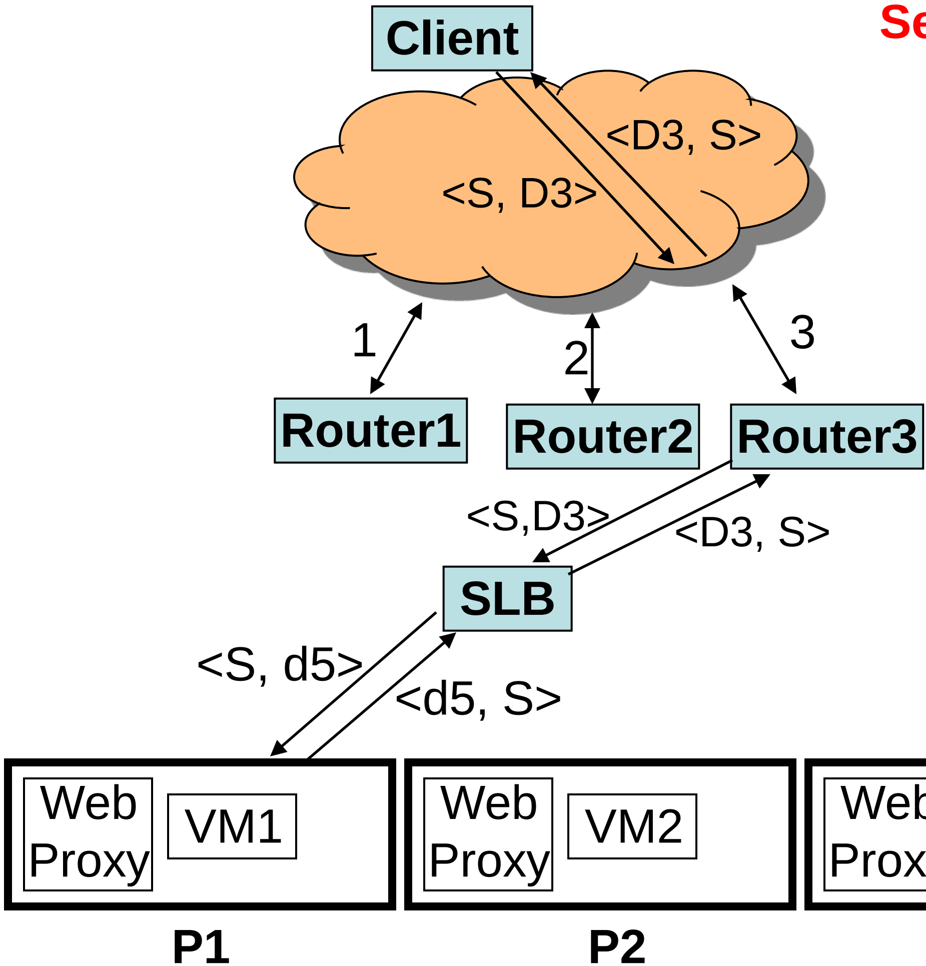


Session-oriented load balancing

Case 1: Each ISP assigns user a separate public IP address range

Case 2: User owns its own public IP address range ➔ needs to involve BGP for being multi-homed and public IP address range partitioning

Multi-Homing Load Balancing – Internally Initiated Connections



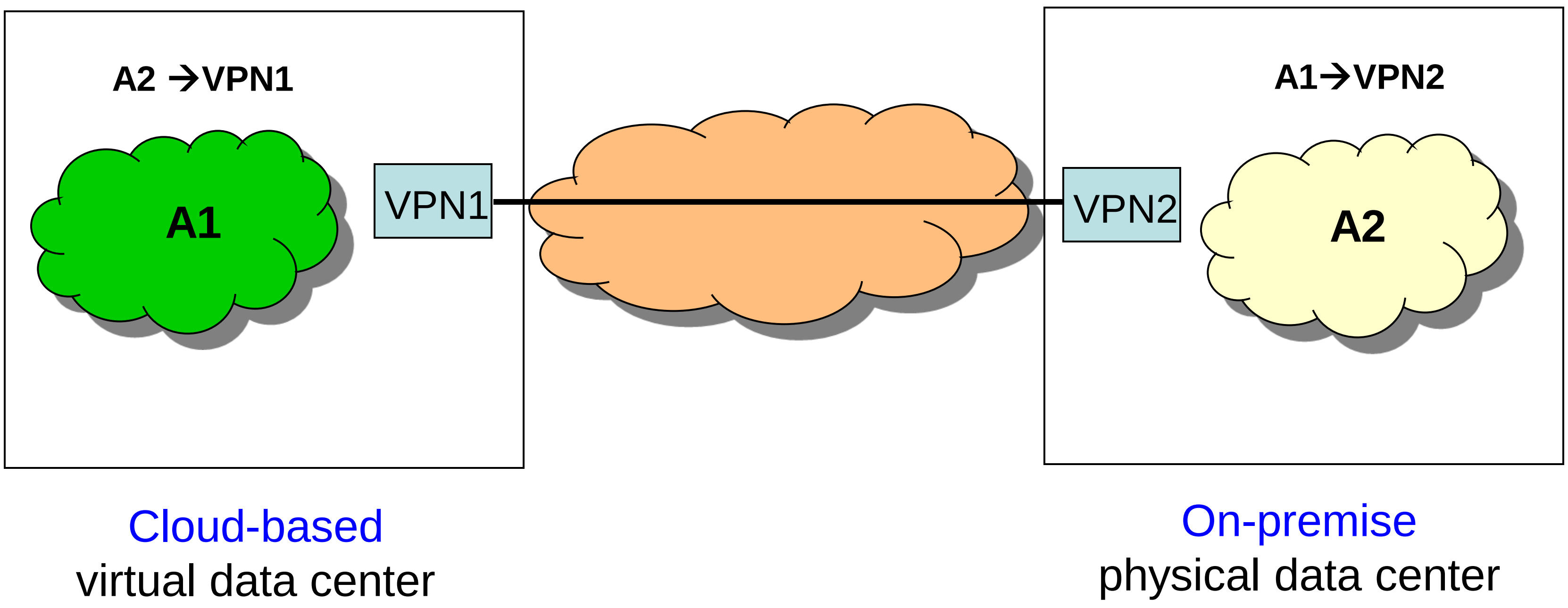
Session-oriented load balancing

Case 1: Each ISP assigns user a separate public IP address range

Case 2: User owns its own public IP address range ➔ needs to involve BGP for being multi-homed and public IP address range partitioning

Network Support for Hybrid Cloud

- Cloud-based VDC and on-premise physical data center share the same private IP address space
- Use a pair of VPN gateways to connect them. The VPN (VPN2 below) gateway on the on-premise data center cannot be modified



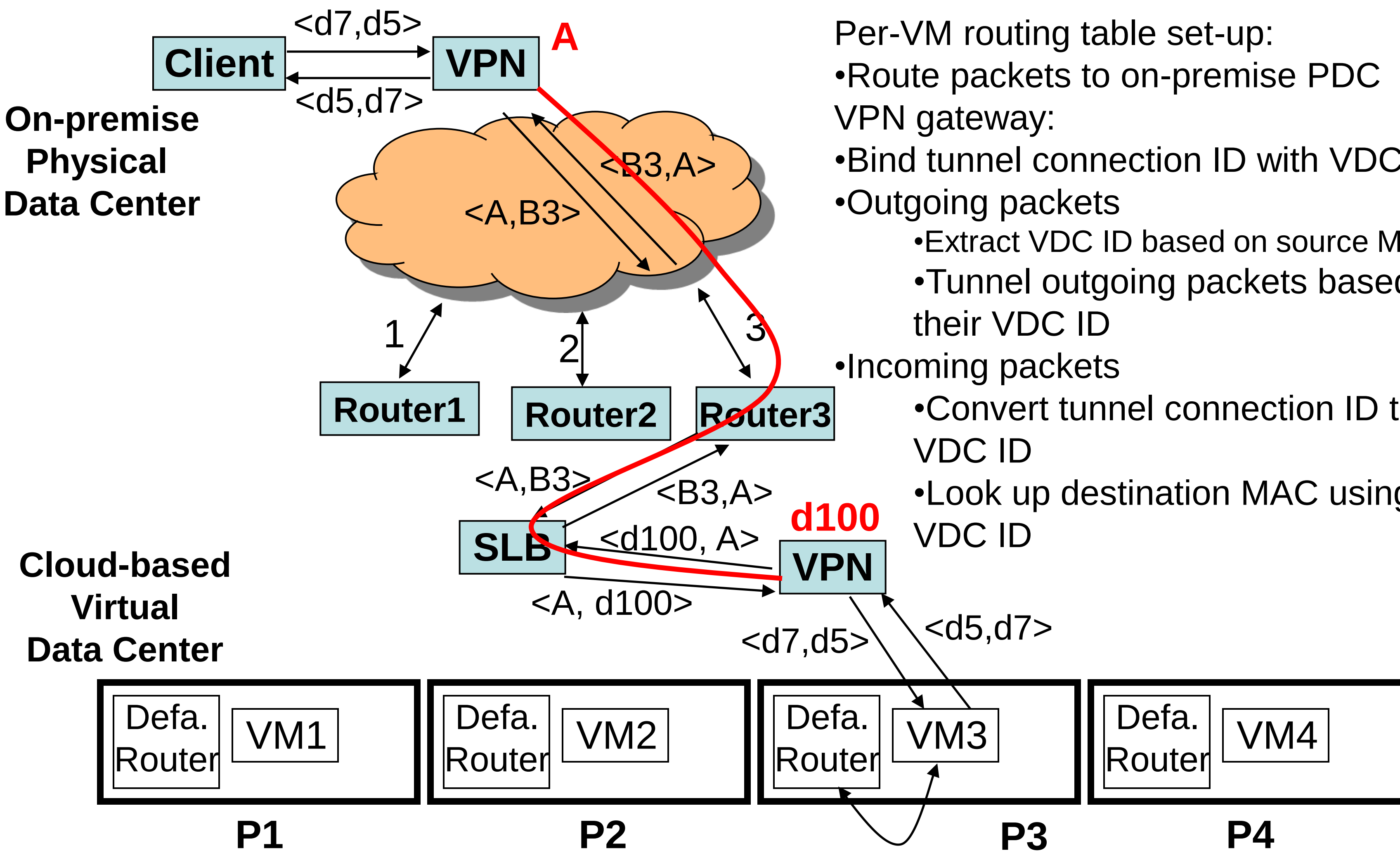
PASR: Private IP Address Reuse

- Every VDC has a VDC ID and its own full 24-bit private IP address space (10.x.x.x), even though multiple VDCs run on top of the same data center network
  - The data center network must be based ONLY on L2 or Ethernet switches
- Analogy
  - Virtual address = Private IP address; Physical address = MAC address
  - Service nodes are accessible to all VDCs and thus are given a special range of private IP addresses ➔ Kernel address space (3-4GB) is shared among all processes
- Translation provides both isolation and flexibility
  - VDC ID + private IP address ➔ MAC address
    - MAC address → VDC ID mapping is available
  - When to translate
    - Intercept ARP queries
    - Upon sending out each packet: protection

Multi-Tenancy

- Multiple virtual data centers share a single physical data center
  - How to give each virtual data center its own private IP address space?
  - How to set up and enforce management policies for each virtual data center separately?
  - How to account for resource usage for each virtual data center separately?
  - How to isolate the state and performance of one virtual data center from another?
- Generalization: multiple virtual data centers from multiple providers and multiple on-site physical data centers work as one

VPN + PASR + Multi-Homing LB



Per-VM routing table set-up:

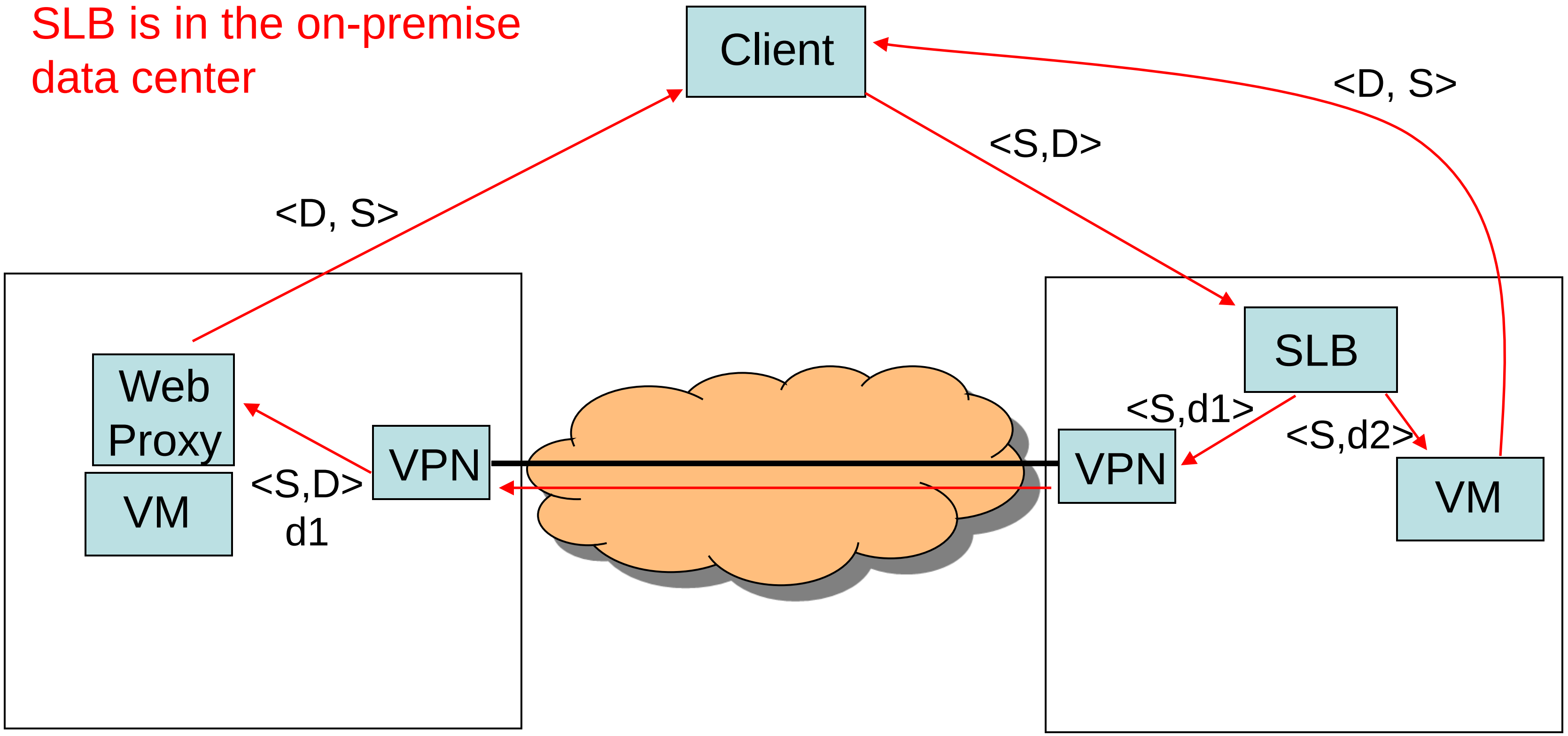
- Route packets to on-premise PDC VPN gateway:
- Bind tunnel connection ID with VDC ID
- Outgoing packets
  - Extract VDC ID based on source MAC
  - Tunnel outgoing packets based on their VDC ID
- Incoming packets
  - Convert tunnel connection ID to VDC ID
  - Look up destination MAC using its VDC ID

General Framework

- Each web service is uniquely identified by N combinations of a public IP address and a port number
- There could be M VMs behind each web service
  - N=1, M=1: port forwarding
  - N=1, M>1: server load balancing
  - N>1, M=1: multi-homing load balancing
  - N>1, M>1: server LB + multi-homing LB
- Multiple tunnels between two VPN gateways
  - Load balancing among multiple VPN tunnels

Hybrid Cloud + 1 Server Load Balancer Direct Return of Response Traffic

SLB is in the on-premise data center

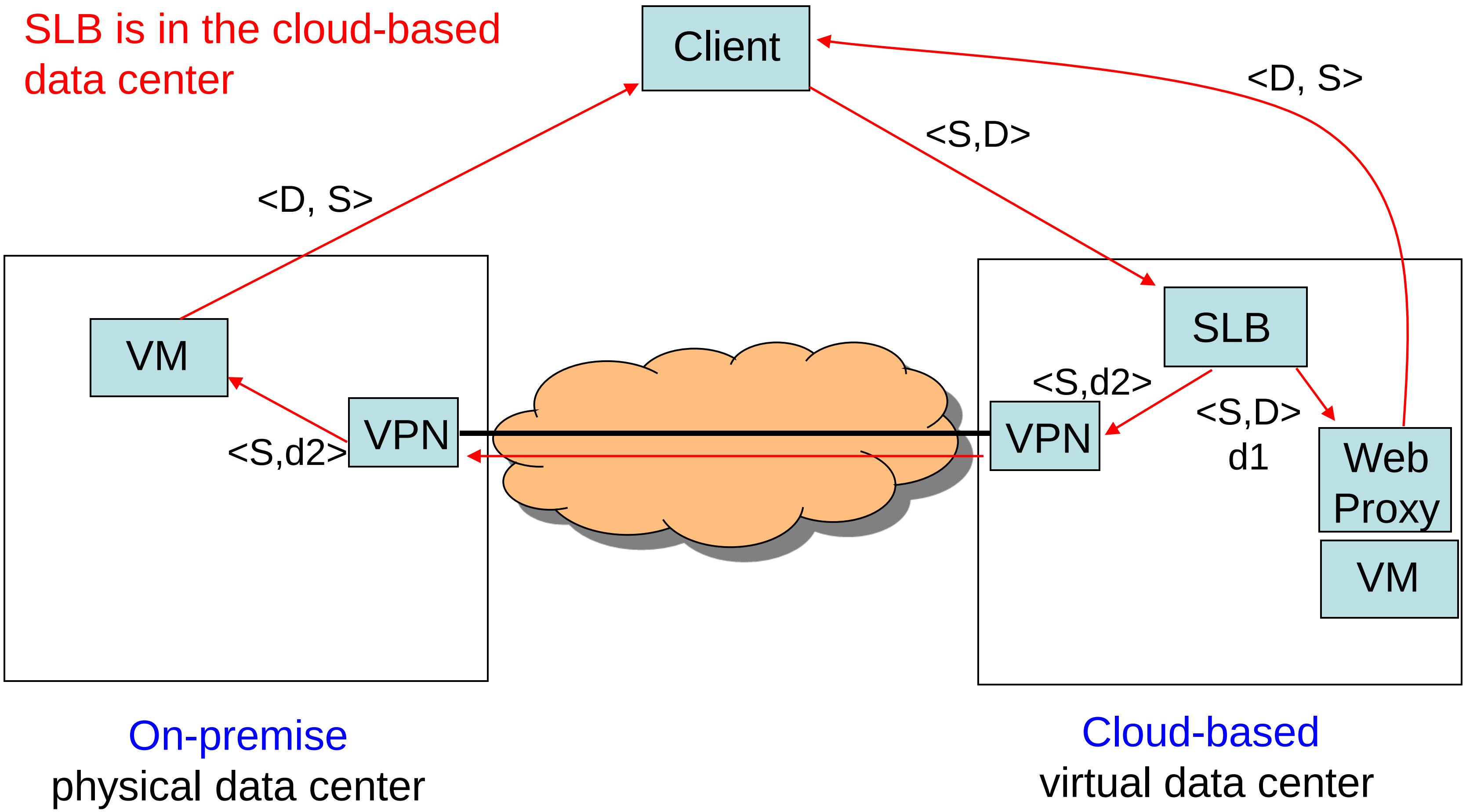


Cloud-based
virtual data center

On-premise
physical data center

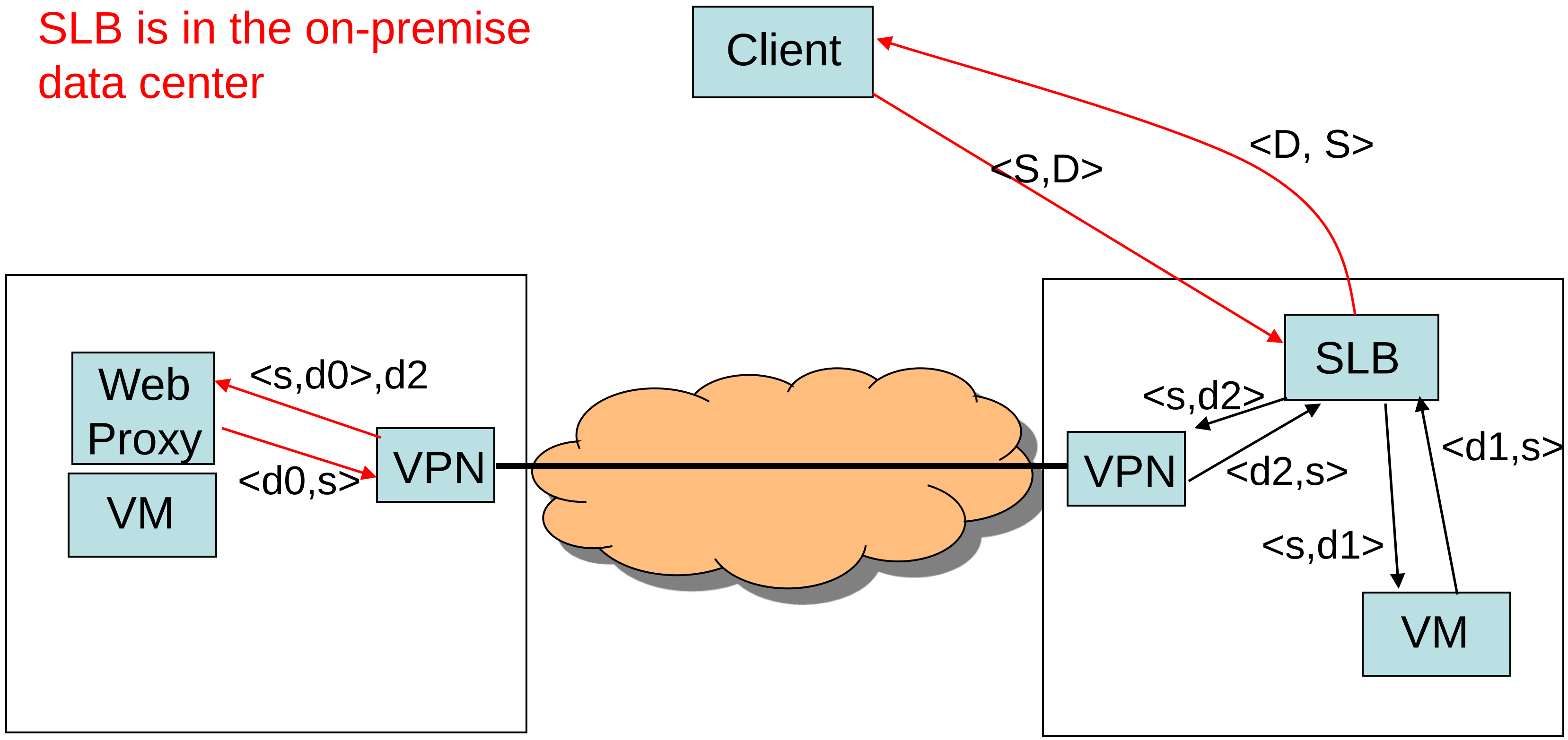
Hybrid Cloud + 1 Server Load Balancer Direct Return of Response Traffic

SLB is in the cloud-based data center



Hybrid Cloud + 1 Server Load Balancer No Direct Return of Response Traffic

SLB is in the on-premise data center

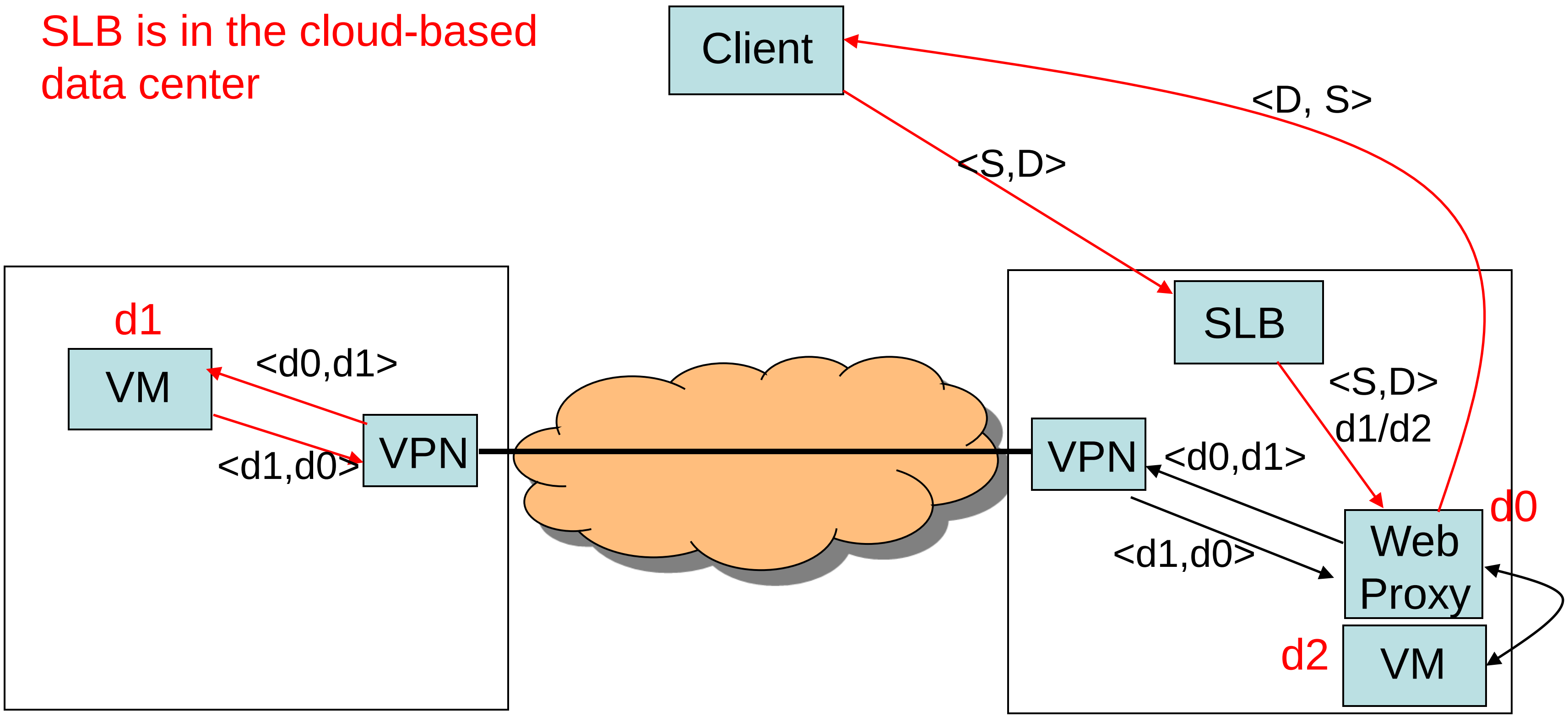


Cloud-based
virtual data center

On-premise
physical data center

Hybrid Cloud + 1 Server Load Balancer
No Direct Return of Response Traffic

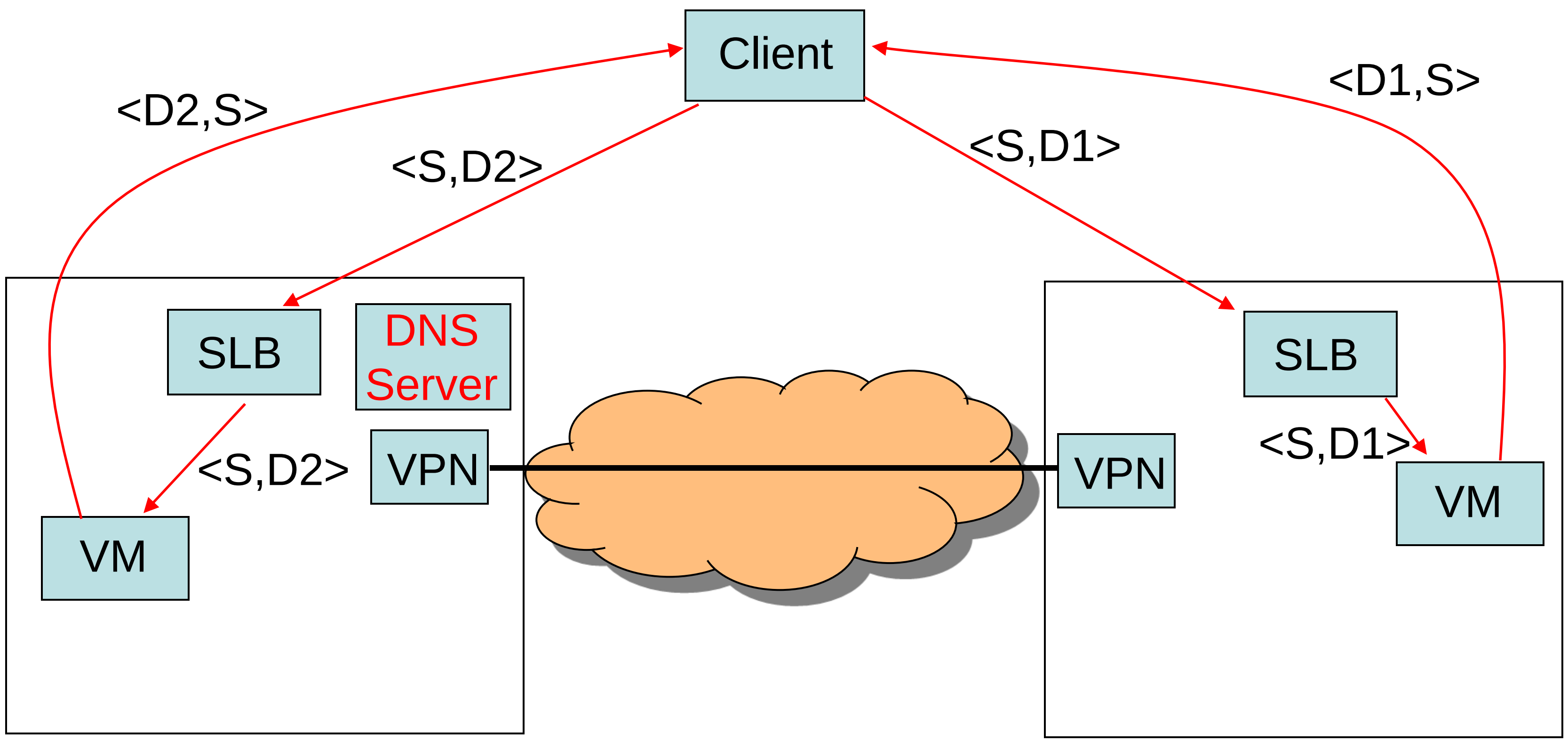
SLB is in the cloud-based data center



On-premise
physical data center

Cloud-based
virtual data center

Hybrid Cloud + 2 Server Load Balancers



Cloud-based
virtual data center

On-premise
physical data center

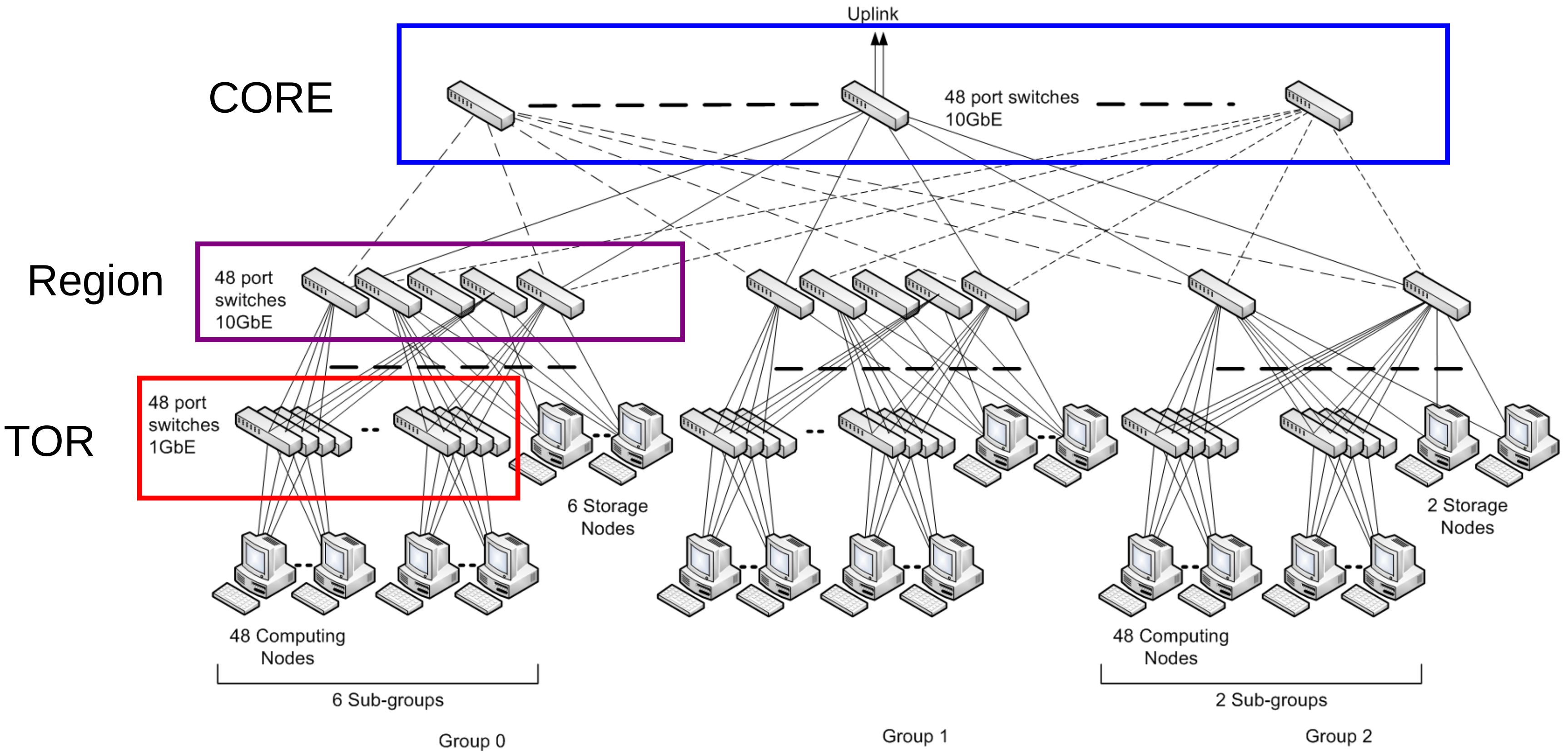
Distributed Traffic Shaping

- Centralized packet scheduling:
  - All traffic goes through a choke point
  - Provision a queue for all outgoing packets from a VDC
  - Schedule packets from multiple queues using a weighted round robin scheduler
    - Time granularity: 1000 bytes per msec vs. 1M bytes per second
    - Bounded credit accumulation
    - Deficit allowance: burst accommodation
- Distributed packet scheduling:
  - Enable direct return of response traffic
  - How to coordinate per-PM schedulers in a responsive and low-overhead manner
    - 10 Mbps shared among 100 VMs = 0.1 Mbps per VM?

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 end points (e.g. 1M)
  - Forwarding table is too small: 16K to 64K
- Does not support VM migration and visibility
- Lack of broadcast traffic scoping
- VM migration limited to a subnet

Peregrine's Network Topology

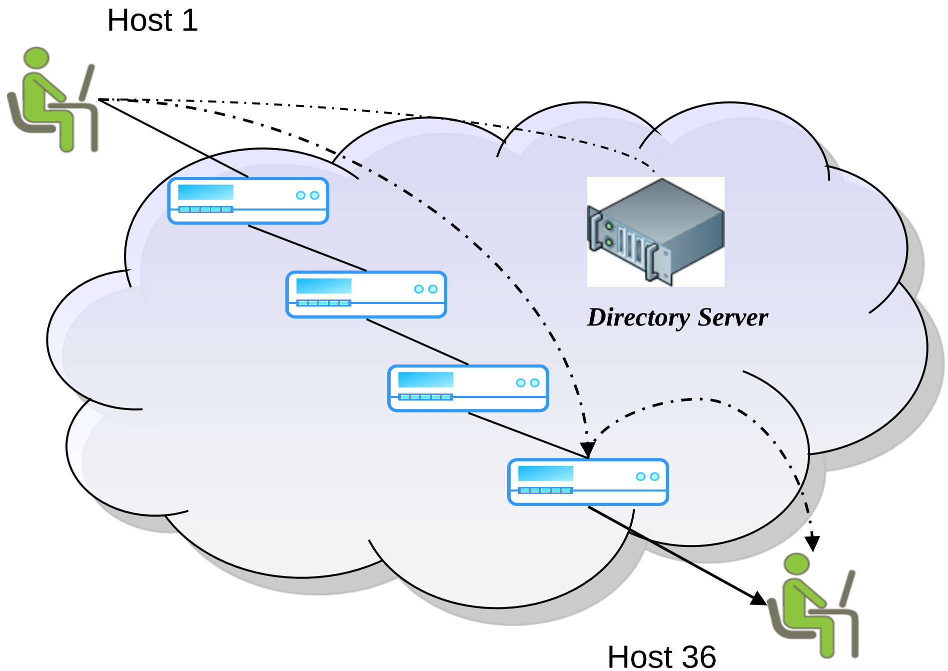


Scaling up to 1M VMs

- Routing vs. Forwarding
- 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
  - Indirect: source → intermediate → destination

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)

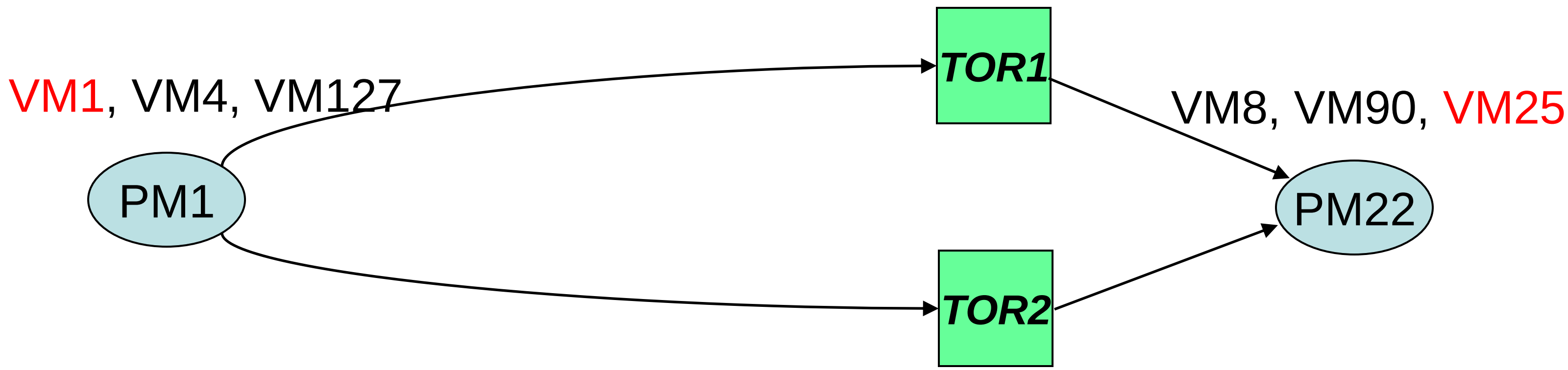


Fast Fail-Over

- Goal: Fail-over latency < 50 msec
- Strategy: Pre-compute a primary and backup route for each VM
  - Each VM has two virtual MACs
  - Asymmetric routing
  - When a link fails, notify hosts using affected primary routes that they should switch to corresponding backup routes
- Route computation is dynamic and aims to balance the loads on physical network links

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(TOR2) or MAC1(PM22), MAC2(PM22)



When a VM Moves

- Notify old and new Intermediaries
- Invalidate the ARP entry of this VM on all other VMs that communicate with it
- Invalidate (asynchronously) all direct forwarding entries of this VM on the network

Additional Issues

- Performance Isolation between storage access traffic and application traffic
- Scalability of directory server
- Relative effectiveness of random routing (e.g., Valiant load balancing) and load-aware routing
- Granularity of fail-over group: When a link fails, how many node pairs are affected
  - All node pairs whose route goes through the failed link
  - All per-node spanning trees that contain the failed link

PCIe-based Rack Area Networking

- Problems:
  - 10GE NIC is expensive and power hungry
    - Multiple 1GE NICs require too many cables
  - Directly attached disks should be accessible when the host CPUs are turned off or die
- Solution: I/O device consolidation or sharing
  - Single-root IOV: multiple VMs on the same host can share a set of I/O devices without conflicting with one another
  - Multiple-root IOV: multiple VMs from multiple hosts can share a set of I/O devices without conflicting with each other
- PCIe network is a promising candidate
  - Lower power consumption
  - How to use SR-IOV hardware to support MR-IOV
  - How to integrate PCIe network with Peregrine

Conclusions

- Cloud data center network issues
  - Rack area networking
  - All-L2 data center backbone (e.g. TRILL)
  - Internet edge logic
- Existing solutions are fragmented or incomplete
- Plenty of room for innovation for a fully integrated solution
- ITRI/CCMA is working at full steam on this

Thank You!

Questions and Comments?

tcc@itri.org.tw

Internet Edge Functionalities

- 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)

Symmetric vs. Asymmetric Routing

- Intermediary of a VM is its associated PM, which has three MAC addresses, I1, I2 and I3, and I3 never appears in any forwarding tables
- Source address check inside switch: a packet with source address A that comes in through port P1 but is supposed to routed via P2 will be dropped ➔ prevents asymmetric routing unless source address modification is used
- Direct forwarding: s ➔ d;   Indirect forwarding: s ➔ I(d) ➔ d
- Look-up: VDCid(d) + IPaddr(d) + I3(s)
  - I3(s): d1, d2, I1(d), I2(d) [0 0 1 1], [0 0 0 1], [0 1 0 1]

    first direct MAC address, second direct MAC address, first intermediary, and second intermediary
- PASR check on outgoing packets
  - I3(s) + VDCid(d) + IPaddr(d): d, I(d) = ARPcache_lookup(VDCid(d)+IPaddr(d))

    s → I3(s) ➔ guarantee source address never matches any forwarding table entry and thus enables asymmetric routing