Peregrine: An Ethernet-based Software Defined Network Architecture

Tzi-cker Chieueh

Cloud Computing Research Center for Mobile Applications (CCMA)
Cloud Data Center Architecture
Cloud Data Center Network

• Cloud data centers are **Big and Shared**
  – **Data center virtualization**: multiple virtual data centers (VDC) on a single physical data center

• **Scalable and available data center fabrics**
  – Use of all physical links in a load-balancing way
  – Fail-over latency is small (< 100 msec)

• **Network virtualization**: Each virtual data center (VDC) gets to define its own virtual network
  – Private IP address reuse
  – VDC-aware VPN and NAT
  – Cross-site global load balancing
Weaknesses of Ethernet’s Control Plane

• 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 (> 1M)
  – Forwarding table is too small: 16K to 64K
• Does not support VM migration and visibility
• Does not support network virtualization, e.g., private IP address reuse (PIAR)
• Does not support inter-VDC isolation: VMs in one VDC cannot ping VMs in another VDC
**Peregrine: Ethernet-based SDN**

- A unified Layer-2-only network for LAN and SAN using *centralized* control plane and *distributed* data plane
- A software defined network using *commodity* Ethernet switches
  - SDN ≠ OpenFlow
  - All Ethernet control plane functionalities are turned off: spanning tree, source learning, unknown DST flooding, source MAC check, etc.
- Centralized load-balancing and QoS-aware routing using real-time traffic matrix, traffic volume between each node pair
- Fast fail-over using pre-computed primary/back routes
- Centralized ARP server to control IP ➔ MAC address mapping
  - Minimum-route-change VM migration
  - Fast fail-over
  - Private IP address reuse
Software Architecture

DS

Directory Server

RAS

Route Algorithm Server

Physical Server

VM0  VM1  VMn

MIM agent

Layer-2-Only Clos Network
Load Balancing Routing

• Collection of real-time traffic matrix
  – Traffic volume between each pair of VMs
  – Traffic volume between each pair of PMs

• Load balancing routing algorithm
  – Loads on the physical links
  – Number of hops
  – Forwarding table entries
  – Prioritization: QoS considerations

• Computed routes are programmatically installed on switches
When a Network Link Fails

1. Deploy Forwarding table
2. Link down trap
3. Update
4. Update cache
5. Backup Path

VM6
mac1: Primary
mac2: Backup
Network Virtualization

• Multiple virtual networks running on a single physical network
• Why not VLAN? VLAN ID is too small; multiple VLANs per VDC
• The network of each virtual data center (VDC) consists of
  – VMs’ MAC addresses are pre-assigned and non-reusable
  – A complete reusable private IP address space, organized into multiple subnets each with its own broadcast domain
  – A set of public IP addresses and NAT/VPN end points
  – Its own DHCP and ARP server
  – Traffic shaping policy
  – Server load balancing policy
  – Cross-site load balancing policy
  – Intra-VDC and inter-VDC firewall policy
Hybrid Cloud Support

Cloud-based virtual data center

On-premise physical data center

A1, A2, A4 → VPN3

A2, A3, A4 → VPN1

A1, A3, A4 → VPN2

A1, A2, A3 → VPN4

A1, A2, A4 → VPN3

Internet
Private IP Address Space Reuse

- **Requirement**: 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; works across VPN connections

- **Two approaches**:
  - **Ethernet over TCP/UDP**:
    - Every Ethernet packet is encapsulated inside an TCP/UDP packet or TCP/UDP connection as an Ethernet link
    - Needs to implement in software such Ethernet switch functions as source learning, flooding, VLAN, etc.
    - Can work with arbitrary IP networks
  - **Multi-tenancy-aware IP-MAC mapping**: our approach
    - VDC ID + private IP address ➔ MAC address
    - MAC address ➔ VDC ID
    - Destination MAC address enforcement
    - Runs directly on L2 networks, no need for Ethernet switch emulation
Peregrine in SDN Framework

- **Data plane**: Ethernet switches vs. OpenFlow switches
- **Southbound API or control protocol**: SNMP/CLI vs. OpenFlow protocol
- **Controller**:
  - RAS for SNMP trap processing and switch configuration vs. OpenFlow controller
- **Applications**:
  - Dynamic load balancing routing
  - Fast fail-over
  - Network virtualization
- **Northbound API**: Quantum from OpenStack
Peregrine Summary

• Peregrine is a network system technology, not a network device technology, and consists of
  – A hypervisor agent running on every compute node
    • L7/Web application firewall and out-going traffic NAT and shaping
  – A centralized route server and ARP server
  – A VDC-aware Internet Edge Logic cluster
    • Server load balancing, VPN, NAT, L4 firewall, and incoming traffic NAT and shaping

• Current Status: A fully operational Peregrine prototype that works on a 10-switch and 100-server test-bed

• Network virtualization technology that does not require tunneling

• A software defined network (SDN) architecture that runs on commodity Ethernet switches, and is able to manage both legacy Ethernet and OpenFlow switches
SDN ≠ OpenFlow

Thank You!

Questions and Comments?

tcc@itri.org.tw
SDN Problems and Functions

• Three impactful SDN functions
  – Dynamic load-balancing routing
  – Fast fail-over
  – Private IP address reuse

• Three important SDN problems
  – Applying SDN to COTS Ethernet switches
  – Having unified control over Ethernet and OpenFlow switches
  – In-band vs. out-of-band control network
Standardization Opportunities

- **SDN-friendly** Ethernet switches
- **Removal of unwanted functionalities**
  - Disables flooding of packets with unknown destination MAC address ➔ prevents loop storms
  - Disables source MAC address check ➔ enables asymmetric routing
- **Addition of desirable functionalities**
  - Interrupt-based rather than polling-based link/switch failure detection ➔ reduces failure detection time
  - Bulk forwarding table updates ➔ speeds up forwarding table programming
  - Multiple notification targets for each SNMP trap ➔ allows fail-over of control plane
  - MAC address aliasing ➔ gives multiple MAC addresses to each NIC