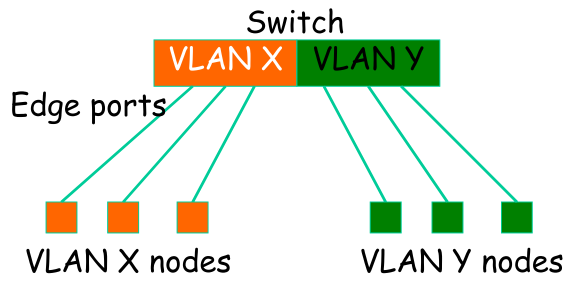
# VLAN & Data Center Networking

# Virtual LANs (VLANs)

- Allow us to split switches into separate (virtual) switches
- Only members of a VLAN can see that VLAN's traffic
  - Inter-vlan traffic must go through a router

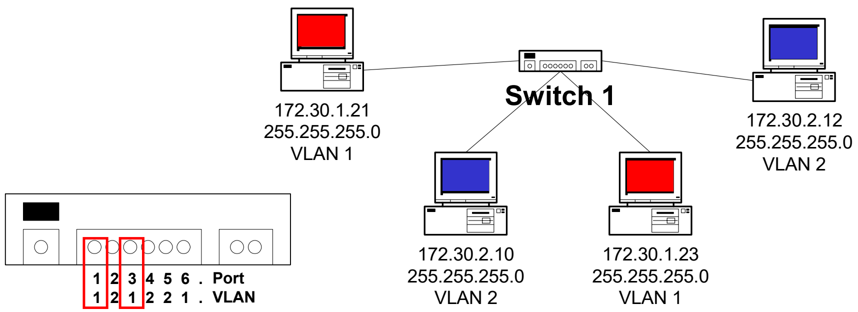



# Local VLANs

- 2 VLANs or more within a single switch
- ***Edge ports***, where end nodes are connected, are configured as members of a VLAN
- The switch behaves as several virtual switches, sending traffic only within VLAN members.
- Switches may not bridge any traffic between VLANs, as this would violate the integrity of the VLAN broadcast domain.
- Traffic should only be routed between VLANs.

# VLAN operation

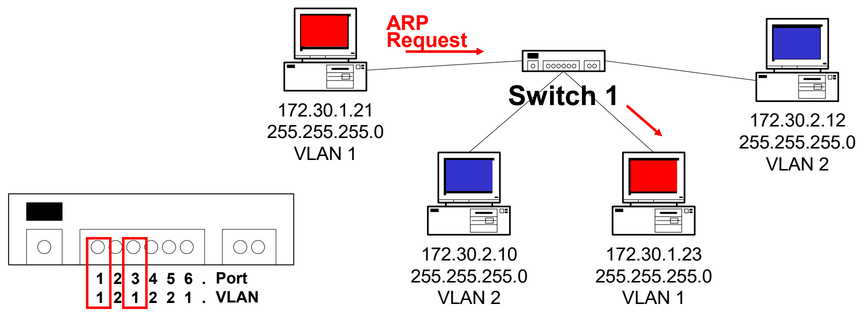
- As a device enters the network, it automatically assumes the VLAN membership of the port to which it is attached.
- The default VLAN for every port in the switch is VLAN 1 and cannot be deleted.
- Ports on the switch may be reassigned to alternate VLANs.




Two **VLANs** = Two **subnets**

- Two Subnets

- Important notes on VLANs:
- VLANs are assigned to switch ports. There is no "VLAN" assignment done on the host (usually).
- In order for a host to be a part of that VLAN, it must be assigned an IP address that belongs to the proper subnet. *Remember: **VLAN = Subnet***
- Assigning a host to the correct VLAN is a 2-step process:
  - Connect the host to the correct port on the switch.
  - Assign to the host the correct IP address depending on the VLAN membership

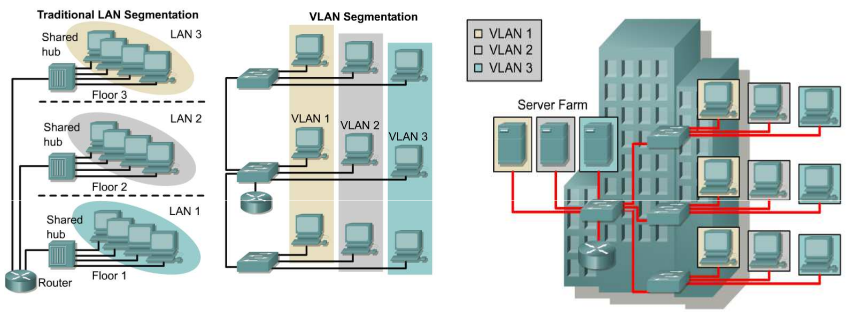



Two **VLANs** = Two **subnets**

- Two Subnets

- VLANs separate broadcast domains!
  e.g. without VLAN the ARP would be seen on all subnets.

# VLANs

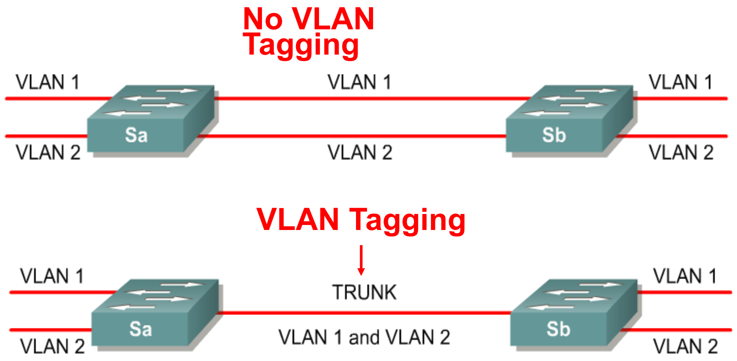



- VLANs logically segment switched networks based on the functions, project teams, or applications of the organization regardless of the physical location or connections to the network.
- All workstations and servers used by a particular workgroup share the same VLAN, regardless of the physical connection or location.

# VLANs across switches

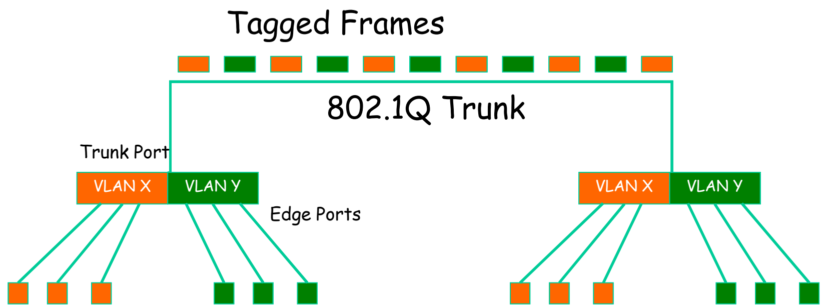
- Two switches can exchange traffic from one or more VLANs
- Inter-switch links are configured as ***trunks***, carrying frames from all or a subset of a switch's VLANs
- Each frame carries a ***tag*** that identifies which VLAN it belongs to

# VLANs across switches




- VLAN tagging is used when a single link needs to carry traffic for more than one VLAN.

# VLANs across switches




This is called "VLAN Trunking"

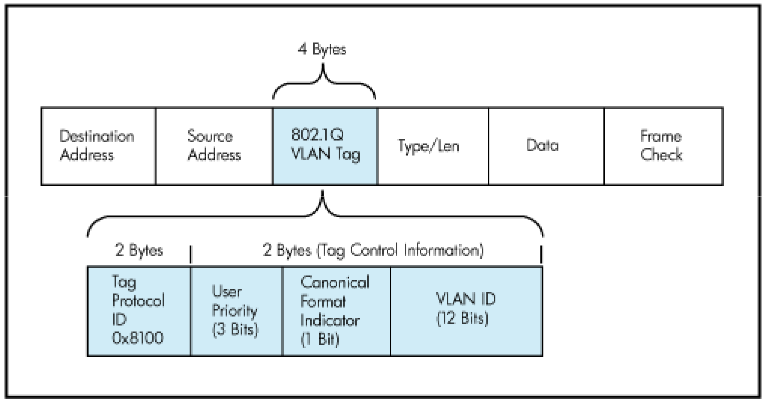
# Tagged vs. Untagged

- Edge ports are not tagged, they are just "members" of a VLAN
- You only need to tag frames in switch-to-switch links (trunks), when transporting multiple VLANs
- A trunk can transport both tagged and untagged VLANs
  - As long as the two switches agree on how to handle those

# 802.1Q

- The IEEE standard that defines how ethernet frames should be ***tagged*** when moving across switch trunks
- This means that switches from *different vendors* are able to exchange VLAN traffic.

# 802.1Q tagged frame




# 802.1Q Header

- A 4-byte tag header containing a tag protocol identifier (TPID) and tag control information (TCI) with the following elements:

**TPID**

- A 2-byte TPID with a fixed value of 0x8100.
- This value indicates that the frame carries the 802.1Q/802.1p tag information.

**TCI**

- A TCI containing the following elements:
  - Three-bit user priority (8 priority levels, 0 thru 7)
  - One-bit canonical format (CFI indicator), 0 = canonical, 1 = noncanonical, to signal bit order in the encapsulated frame (www.faqs.org/rfcs/rfc2469.html - "A Caution On the Canonical Ordering of Link-Layer Addresses")
  - Twelve-bit VLAN identifier (VID)-Uniquely identifies the VLAN to which the frame belongs, defining 4,096 VLANs, with 0 and 4095 reserved.

# VLANS increase complexity

- You can no longer "just replace" a switch
  - Now you have VLAN configuration to maintain
  - Field technicians need more skills
- You have to make sure that all the switch-to-switch trunks are carrying all the necessary VLANs
  - Need to keep in mind when adding/removing VLANs

# Data Center Networking

Major Theme:

What are new networking issues posed by large-scale data centers?

- Network Architecture?
- Topology design?
- Addressing?
- Routing?
- Forwarding?

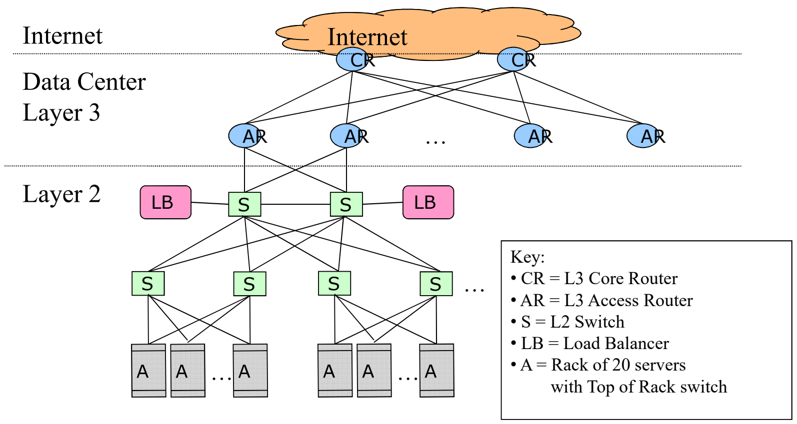
# Data Center Interconnection Structure

- Nodes in the system: racks of servers
- How are the nodes (racks) inter-connected?
  - Typically a hierarchical inter-connection structure
- Today's typical data center structure

  Cisco recommended data center structure:

  starting from the bottom level
  - rack switches
  - 1-2 layers of (layer-2) aggregation switches
  - access routers
  - core routers
- Is such an architecture good enough?

# Cisco Recommended DC Structure: Illustration






# Data Center Costs

| Amortized Cost* | Component | Sub-Components |
|---|---|---|
| ~45% | Servers | CPU, memory, disk |
| ~25% | Power infrastructure | UPS, cooling, power distribution |
| ~15% | Power draw | Electrical utility costs |
| ~15% | Network | Switches, links, transit |

*3 yr amortization for servers, 15 yr for infrastructure; 5% cost of money

- Total cost varies
  - upwards of $1/4 B for mega data center
  - server costs dominate
  - network costs significant
- Long provisioning timescales:
  - new servers purchased quarterly at best

**Source: the Cost of a Cloud: Research Problems in Data Center Networks.** Sigcomm CCR 2009. Greenberg, Hamilton, Maltz, Patel.

# Data Center Design Requirements

- Data centers typically run two types of applications
  - outward facing (e.g., serving web pages to users)
  - internal computations (e.g., MapReduce for web indexing)
- Workloads often unpredictable:
  - Multiple services run concurrently within a DC
  - Demand for new services may spike unexpected
    - Spike of demands for new services mean success!
    - But this is when success spells trouble (if not prepared)!
- Failures of servers are the norm
  - GFS, MapReduce, etc., resort to dynamic re-assignment of chunkservers, jobs/tasks (worker servers) to deal with failures; data is often replicated across racks, …
  - "Traffic matrix" between servers are constantly changing

# Overall Data Center Design Goal

Agility – Any service, Any Server

- Turn the servers into a single large fungible pool
  - Let services "breathe" : dynamically expand and contract their footprint as needed
- Benefits
  - Increase service developer productivity
  - Lower cost
  - Achieve high performance and reliability

These are the three motivators for most data center infrastructure projects!

# Achieving Agility …

- Workload Management
  - means for rapidly installing a service's code on a server
  - *dynamical cluster scheduling and server assignment* ☑
    - *E.g., MapReduce, …*
  - *virtual machines, disk images* ☑
- Storage Management
  - means for a server to access persistent data
  - *distributed file systems (e.g., GFS)* ☑
- Network Management
  - Means for communicating with other servers, regardless of where they are in the data center
  - Achieve high performance and reliability

# Networking Objectives

1. Uniform high capacity
   - Capacity between servers limited only by their NICs
   - No need to consider topology when adding servers

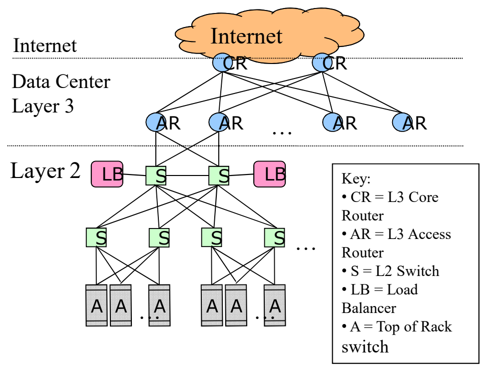
   => In other words, high capacity between two any servers no matter which racks they are located !
2. Performance isolation
   - Traffic of one service should be unaffected by others
3. Ease of management: "Plug-&-Play" (layer-2 semantics)
   - Flat addressing, so any server can have any IP address
   - Server configuration is the same as in a LAN
   - Legacy applications depending on broadcast must work

# Is Today's DC Architecture Adequate?

- Hierarchical network; 1+1 redundancy
- Equipment higher in the hierarchy handles more traffic
  - more expensive, more efforts made at availability
  - Servers connect via 1 Gbps UTP to Top-of-Rack switches
- Other links are mix of 1G, 10G; fiber, copper

- Uniform high capacity?
- Performance isolation?

  typically via VLANs
- Agility in terms of dynamically adding or shrinking servers?
- Agility in terms of adapting to failures, and to traffic dynamics?
- Ease of management?



# A Scalable, Commodity Data Center Network Architecture

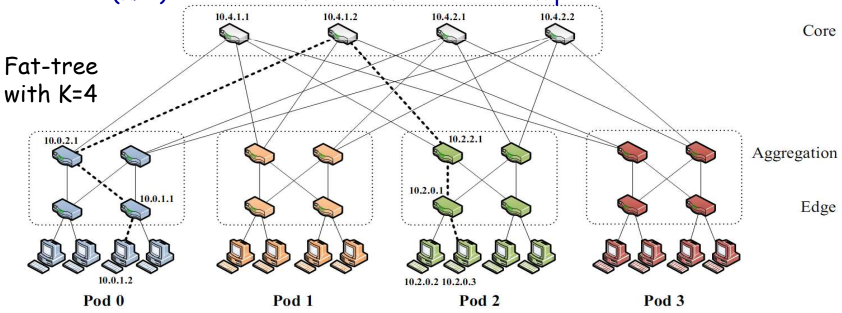
- Main Goal: addressing the limitations of today's data center network architecture
  - sing point of failure
  - over subscript of links higher up in the topology
    - trade-offs between cost and providing
- Key Design Considerations/Goals
  - Allows host communication at line speed
    - no matter where they are located!
  - Backwards compatible with existing infrastructure
    - no changes in application & support of layer 2 (Ethernet)
  - Cost effective
    - cheap infrastructure
    - and low power consumption & heat emission

# Fat-Tree Based DC Architecture

- Inter-connect racks (of servers) using a fat-tree topology
- Fat-Tree: a special type of Clos Networks (after C. Clos)

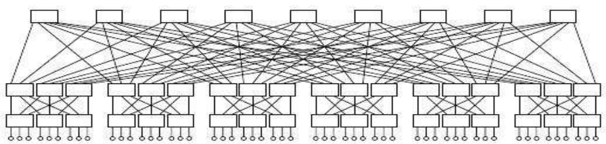
  K-ary fat tree: three-layer topology (edge, aggregation and core)
  - each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches
  - each edge switch connects to k/2 servers & k/2 aggr. switches
  - each aggr. switch connects to k/2 edge & k/2 core switches
  - $(k/2)^2$ core switches: each connects to k pods

Fat-tree with K=4



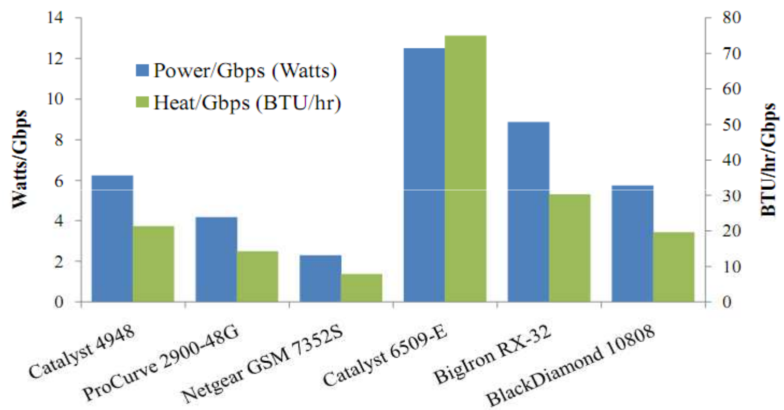

# Fat-Tree Based Topology …

- Why Fat-Tree?
  - Fat tree has identical bandwidth at any bisections
  - Each layer has the same aggregated bandwidth
- Can be built using cheap devices with uniform capacity
  - Each port supports same speed as end host
  - All devices can transmit at line speed if packets are distributed uniform along available paths
- Great scalability: k-port switch supports $k^3/4$ servers



Fat tree network with K = 6 supporting 54 hosts

# Cost of Maintaining Switches



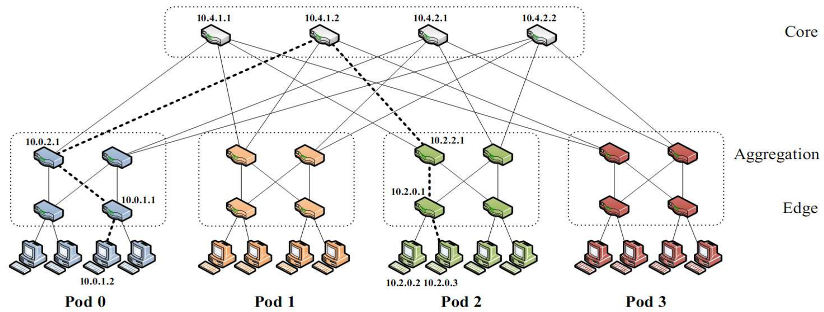
# Fat-tree Topology is Great, But …

Does using fat-tree topology to inter-connect racks of servers in itself sufficient?

- What routing protocols should we run on these switches?
- Layer 2 switch algorithm: data plane flooding!
- Layer 3 IP routing:
  - shortest path IP routing will typically use only one path despite the path diversity in the topology
  - if using equal-cost multi-path routing at each switch independently and blindly, packet re-ordering may occur; further load may not necessarily be well-balanced
  - Aside: control plane flooding!

# Addressing Scheme

- Enforce a special (IP) addressing scheme in DC
  - Allocate IP addresses within the private block 10.0.0.0/8 block
  - Switch address: 10.pod.switch.1
    - pod ∈[0,…,k-1] left to right, switch ∈[0,…,k-1] left to right, bottom to top
  - Core switch address: 10.k.j.i, j,i ∈[1,(k/2)]
  - Host address: 10.pod.switch.ID, ID ∈[2,(k/2)+1]



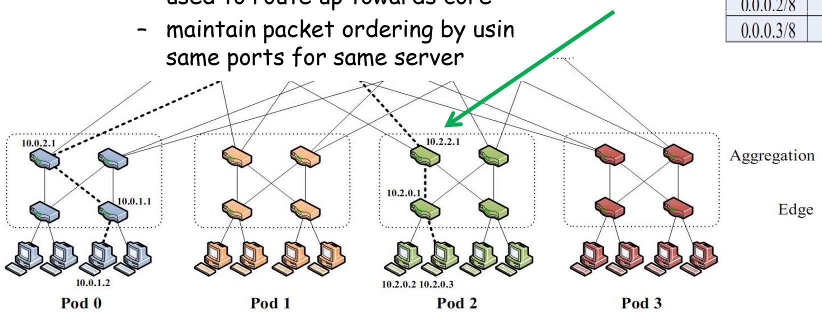

# FAT-Tree Modified

- Use two level look-ups to distribute traffic and maintain packet ordering
  - First level is prefix lookup
    - used to route down the topology to servers
  - Second level is a suffix lookup
    - used to route up towards core
    - maintain packet ordering by usin same ports for same server

10.2.2.1 table

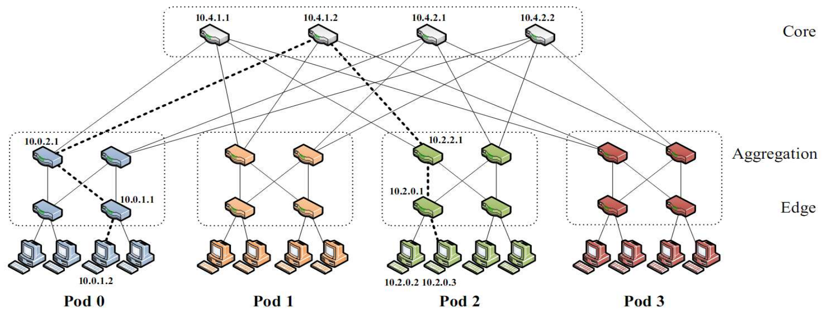
| Prefix | Output port |
|---|---|
| 10.2.0.0/24 | 0 |
| 10.2.1.0/24 | 1 |
| 0.0.0.0/0 | |

| Suffix | Output port |
|---|---|
| 0.0.0.2/8 | 2 |
| 0.0.0.3/8 | 3 |




# FAT-Tree Modified

- Use two level look-ups to distribute traffic and maintain packet ordering
  - Core Switches: terminating 1st level prefixes for all network ID, pointing to the appropriate ID
    - 1 link from each core to each pod
    - /16 prefix (10.pod.0.0/16, port)



# More on Fat-Tree DC Architecture

## Diffusion Optimizations

- Flow classification
  - Eliminates local congestion
  - Assign traffic to ports on a per-flow basis instead of a per-host basis
- Flow scheduling
  - Eliminates global congestion
  - Prevent long lived flows from sharing the same links
  - Assign long lived flows to different links

# PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric

## In a nutshell:

- PortLand is a single "logical layer 2" data center network fabric that scales to millions of endpoints
- PortLand internally separates host identity from host location
  - uses IP address as host identifier
  - introduces "Pseudo MAC" (PMAC) addresses internally to encode endpoint location
- PortLand runs on commodity switch hardware with unmodified hosts

# Design Goals for Network Fabric

## Support for Agility!

- Easy configuration and management: plug-&-play
- Fault tolerance, routing and addressing: scalability
- Commodity switch hardware: small switch state
- Virtualization support: seamless VM migration

What are the limitations of current layer-2 and layer-3?

- layer-2 (Ethernet w/ flat-addressing) vs.
- layer-3 (IP w/ prefix-based addressing):
  - plug-&-play?
  - scalability?
  - small switch state?
  - seamless VM migration?

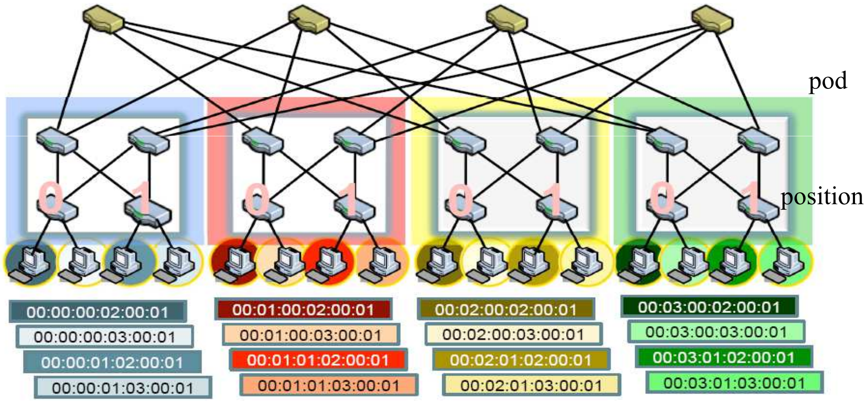
# PortLand Solution

## Assuming: a Fat-tree network topology for DC

- Introduce "pseudo MAC addresses" to balance the pros and cons of flat- vs. topology-dependent addressing
- PMACs are "topology-dependent," hierarchical addresses
  - But used only as "host locators," not "host identities"
  - IP addresses used as "host identities" (for compatibility w/ apps)
- Pros: small switch state & Seamless VM migration
- Pros: "eliminate" flooding in both data & control planes
- But requires a IP-to-PMAC mapping and name resolution
  - a location directory service
- And location discovery protocol & fabric manager
  - for support of "plug-&-play"

# PMAC Addressing Scheme

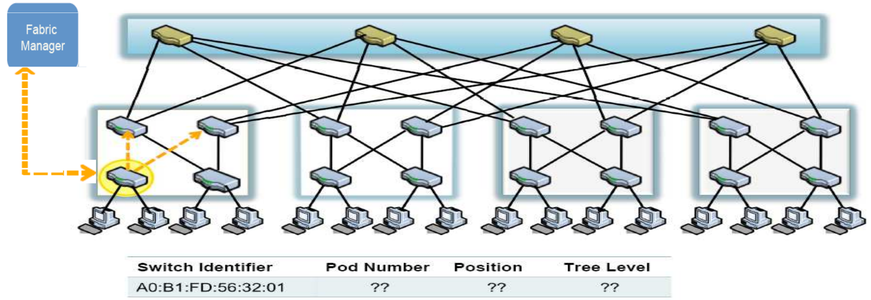
- PMAC (48 bits): pod.position.port.vmid
  - Pod: 16 bits; position and port (8 bits); vmid: 16 bits
- Assign only to servers (end-hosts) – by switches




# Location Discovery Protocol

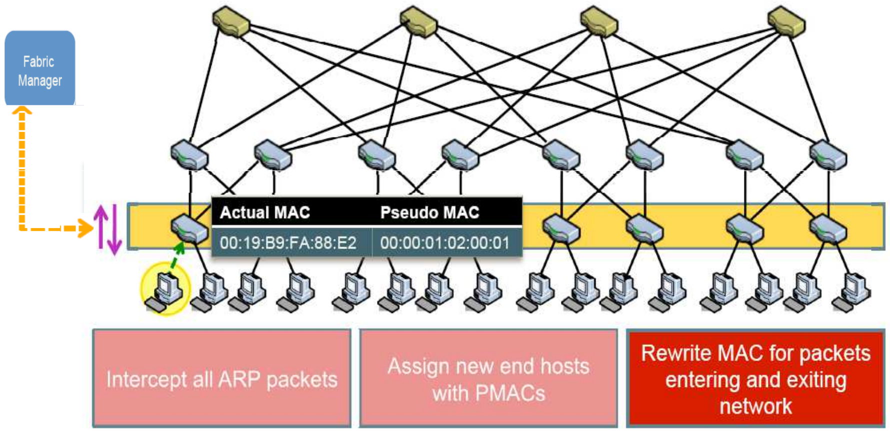
- Location Discovery Messages (LDMs) exchanged between neighboring switches
- Switches self-discover location on boot up

| Location Characteristics | Technique |
|---|---|
| Tree-level (edge, aggr. , core) | auto-discovery via neighbor connectivity |
| Position # | aggregation switch help edge switches decide |
| Pod # | request (by pos. 0 switch only) to fabric manager |



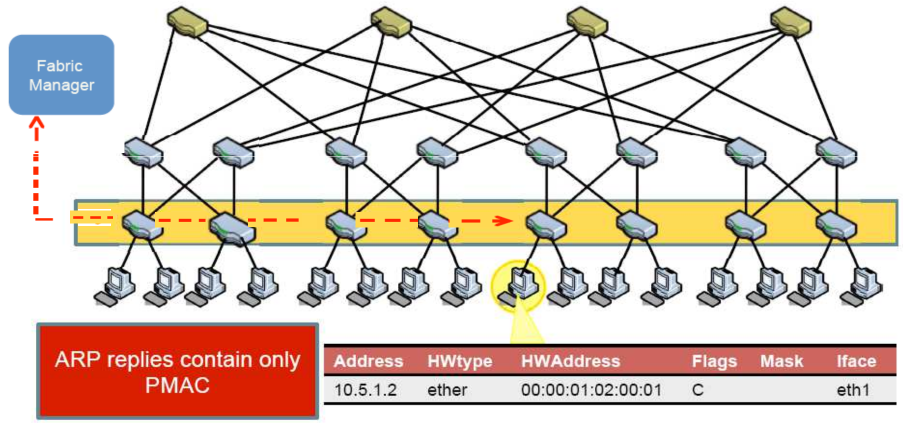

# PortLand: Name Resolution

- Edge switch listens to end hosts, and discover new source MACs
- Installs <IP, PMAC> mappings, and informs fabric manager




# PortLand: Name Resolution …

- Edge switch intercepts ARP messages from end hosts
- send request to fabric manager, which replies with PMAC



# PortLand: Fabric Manager

- fabric manager: logically centralized, multi-homed server
- maintains topology and <IP,PMAC> mappings in "soft state"