Scale and Efficiency in Data Center Networks

Amin Vahdat
Computer Science and Engineering
Center for Networked Systems
UC San Diego

vahdat@cs.ucsd.edu

UC San Diego
Center for Networked Systems

Member Companies

Center Faculty

Research Interests

Project Proposals

Diverse Research Projects

-Multiple faculty
-Multiple students
-Multidisciplinary
-CNS Research Theme

Current Team

- Amin Vahdat
- Mohammad Al-Fares
- Hamid Bazzaz
- Harshit Chitalia
- Sambit Das
- Shaya Fainman
- Nathan Farrington
- Brian Kantor
- Harsha Madhyastha
- Pardis Miri
- Radhika Niranjan Mysore
- George Papen
- George Porter
- Sivasankar Radhakrishnan
- Erik Rubow
- Aram Shahinfard
- Vikram Subramanya
- Malveeka Tewari
- Meg Walraed-Sullivan

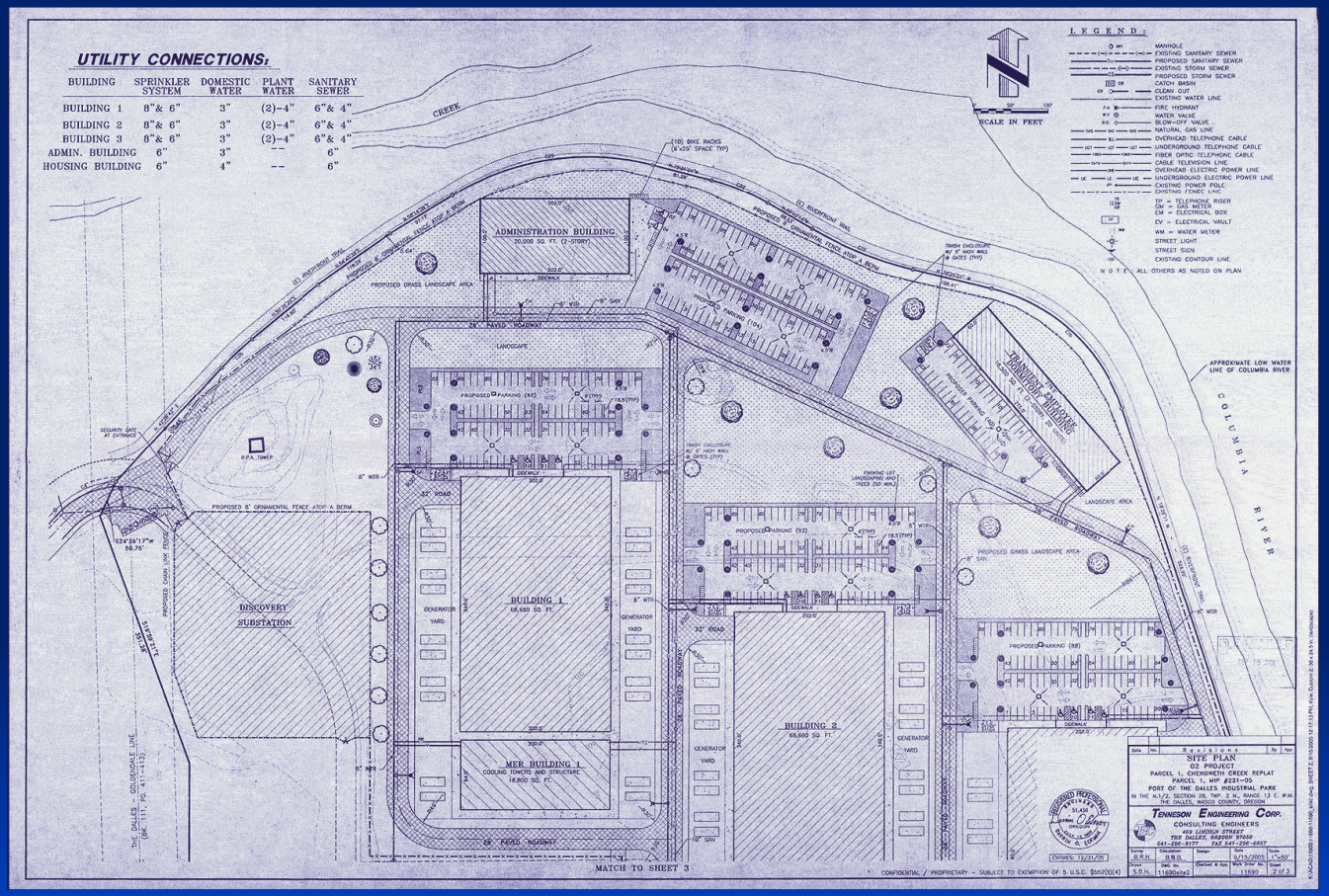
Acknowledgements

- Industry
  - Ericsson, HP, Cisco, Facebook, Yahoo!, Google, Broadcom
- Government
  - National Science Foundation: building ~300 node “data center network” prototype
- Academia
  - Stanford (NetFPGA + OpenFlow)

Motivation

Walmart's Data Center

Motivation

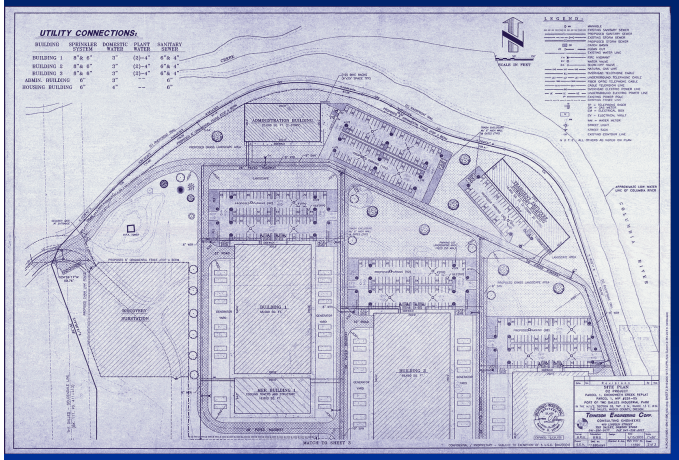


Blueprints for 200k sq. ft. Data Center in OR

Motivation

Google's 36 Worldwide Data Centers

Motivation



- Commoditization in the data center
  - Inexpensive, commodity PCs and storage devices
  - But network still highly specialized
- Data center is not a “small Internet”
  - One admin domain, not adversarial, limited policy routing, etc.
- Bandwidth is often the bottleneck
  - “Cloud” Computing
  - Service-oriented Architectures
  - Data Analysis (MapReduce)

Network Design Goals

- Scalable interconnection bandwidth
  - Full bisection bandwidth between all pairs of hosts

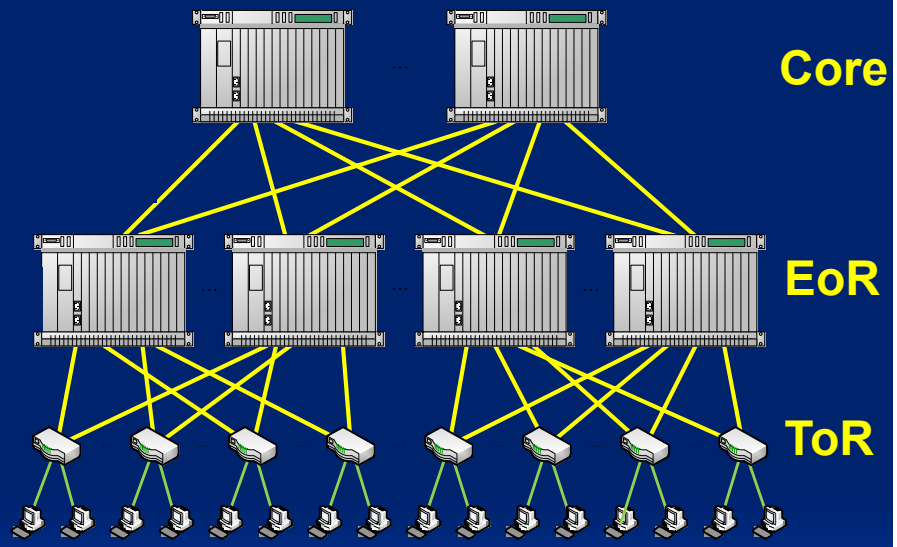
    Aggregate bandwidth = # hosts × host NIC capacity
- Economies of scale
  - Price/port constant with number of hosts
  - Must leverage commodity merchant silicon
- Single network fabric
  - Support Ethernet and IP without end host modifications
- Management
  - Modular design
  - Avoid actively managing 100's-1000's network elements

Scale Out Networking

- Advances toward *scale out* computing and storage
  - Aggregate computing and storage grows linearly with the number of commodity processors and disks
  - Small matter of software to enable functionality
  - Alternative is *scale up* where weaker processors and smaller disks are *replaced* with more powerful parts
- Today, no technology for scale out networking
  - Modules to expand number of ports or aggr BW
  - No management of individual switches, VLANs, subnets

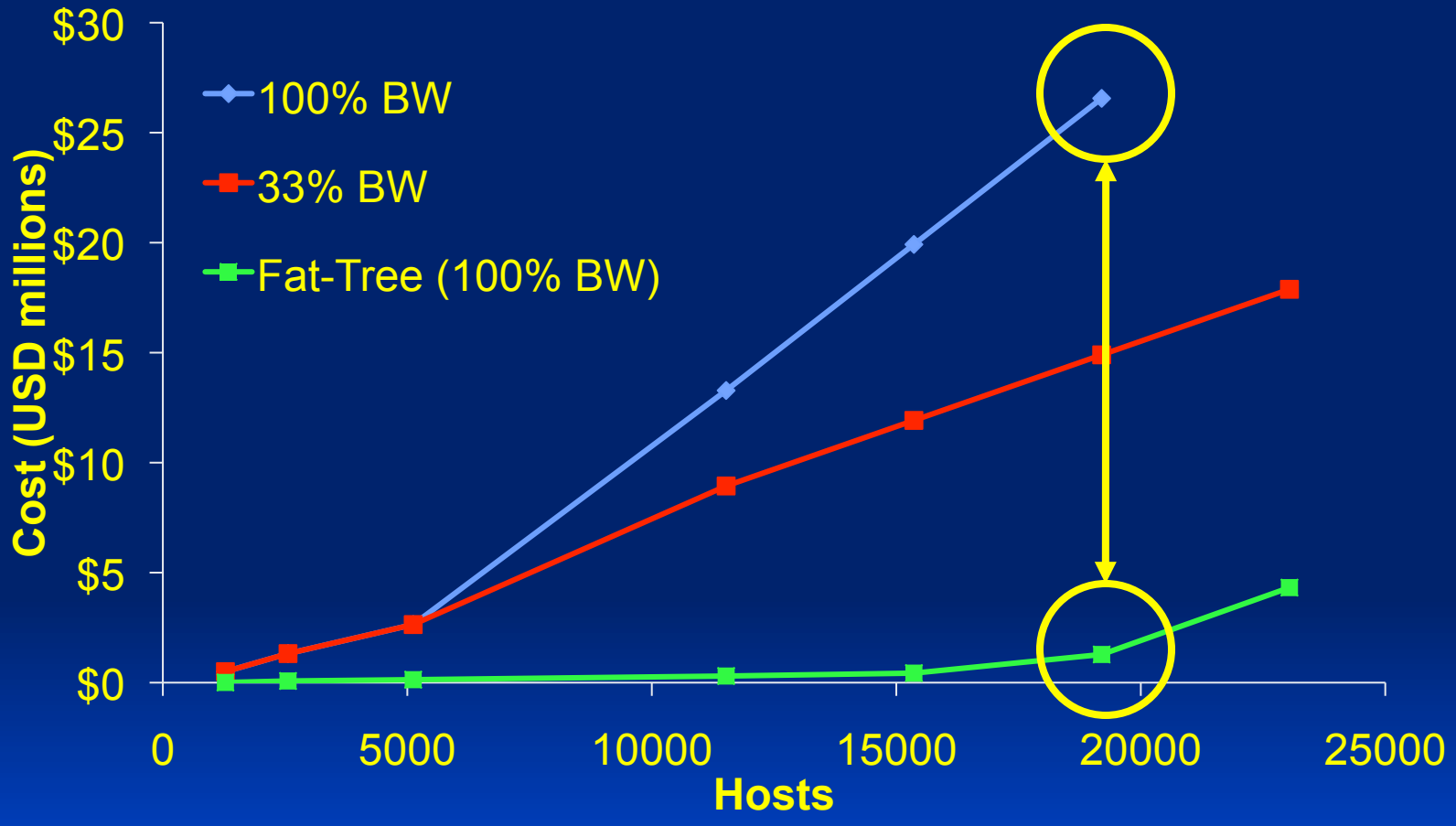
Current Data Center Topologies

- Edge hosts connect to 1G Top of Rack (ToR) switch
- ToR switches connect to 10G End of Row (EoR) switches
- Large clusters: EoR switches to 10G core switches
  - Oversubscription of 2.5:1 to 8:1 typical in guidelines
- No story for what happens as we move to 10G to the edge



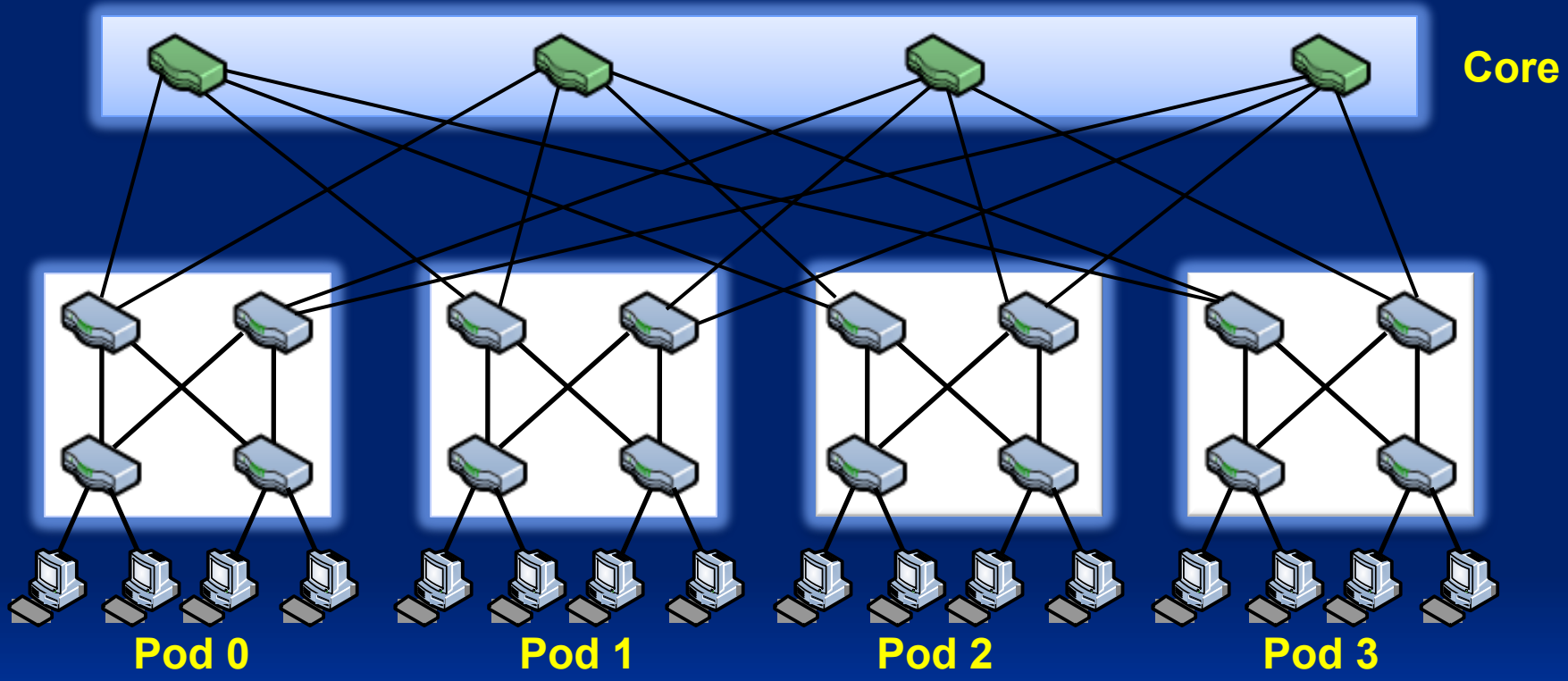
➡ Key challenges: performance, cost, routing, energy, cabling

Cost of Data Center Networks



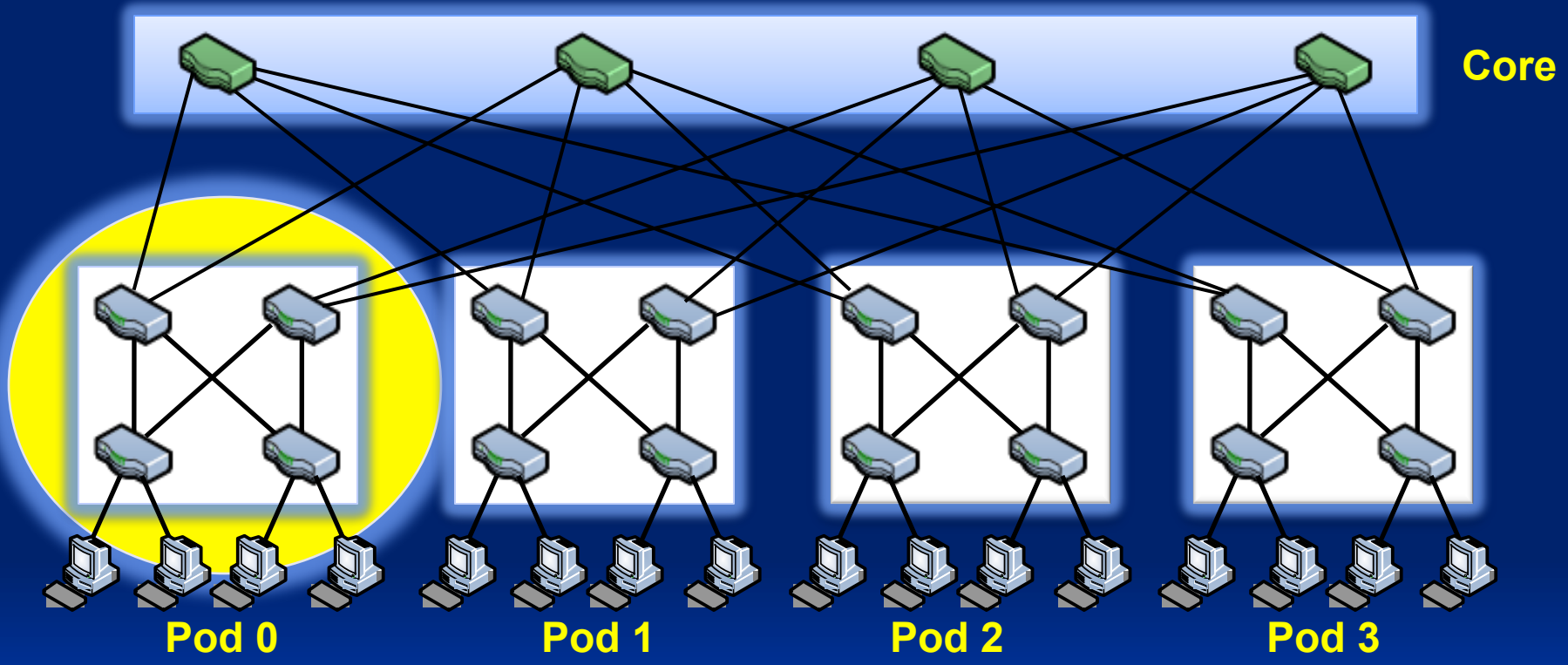
- Factor of 10+ price difference between traditional approach and proposed architecture

Scalability Using Identical Network Elements



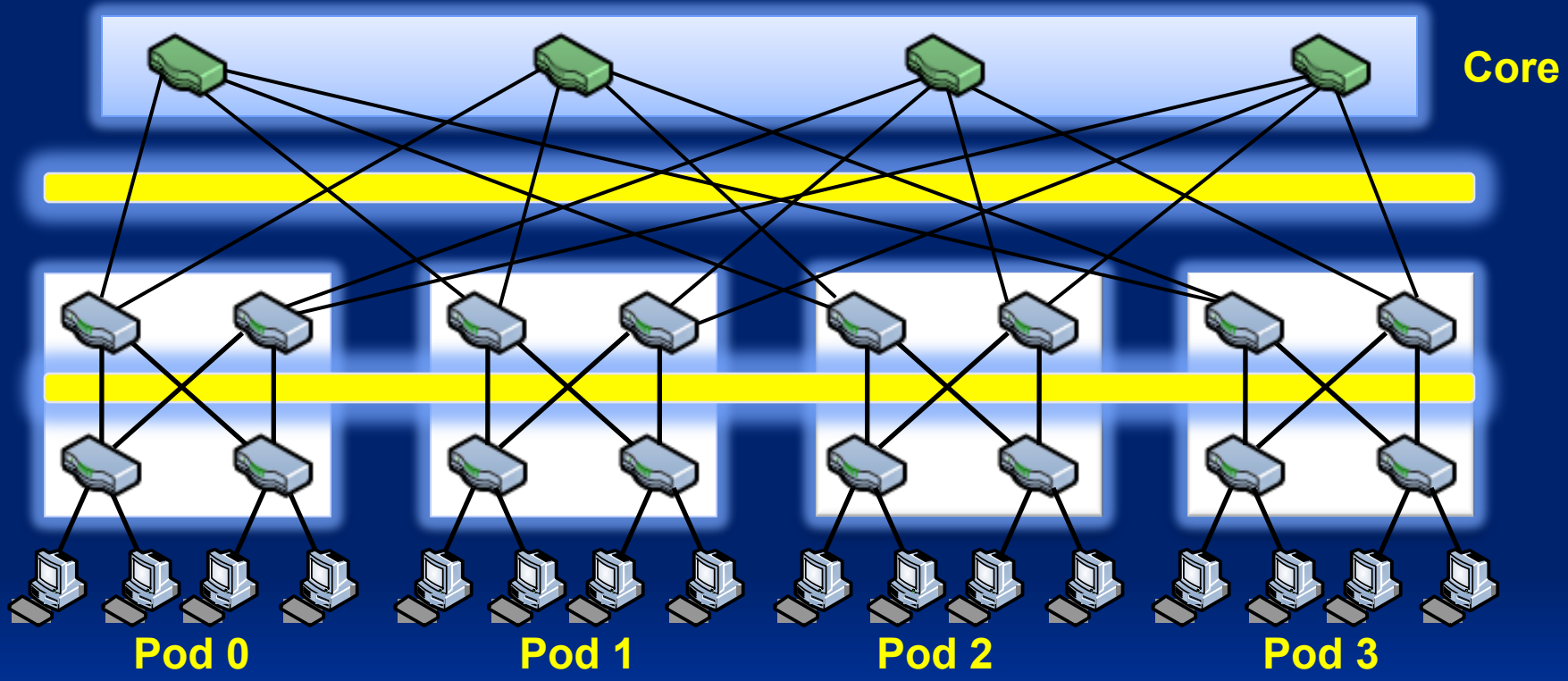
- Fat tree built from 4-port switches

Scalability Using Identical Network Elements



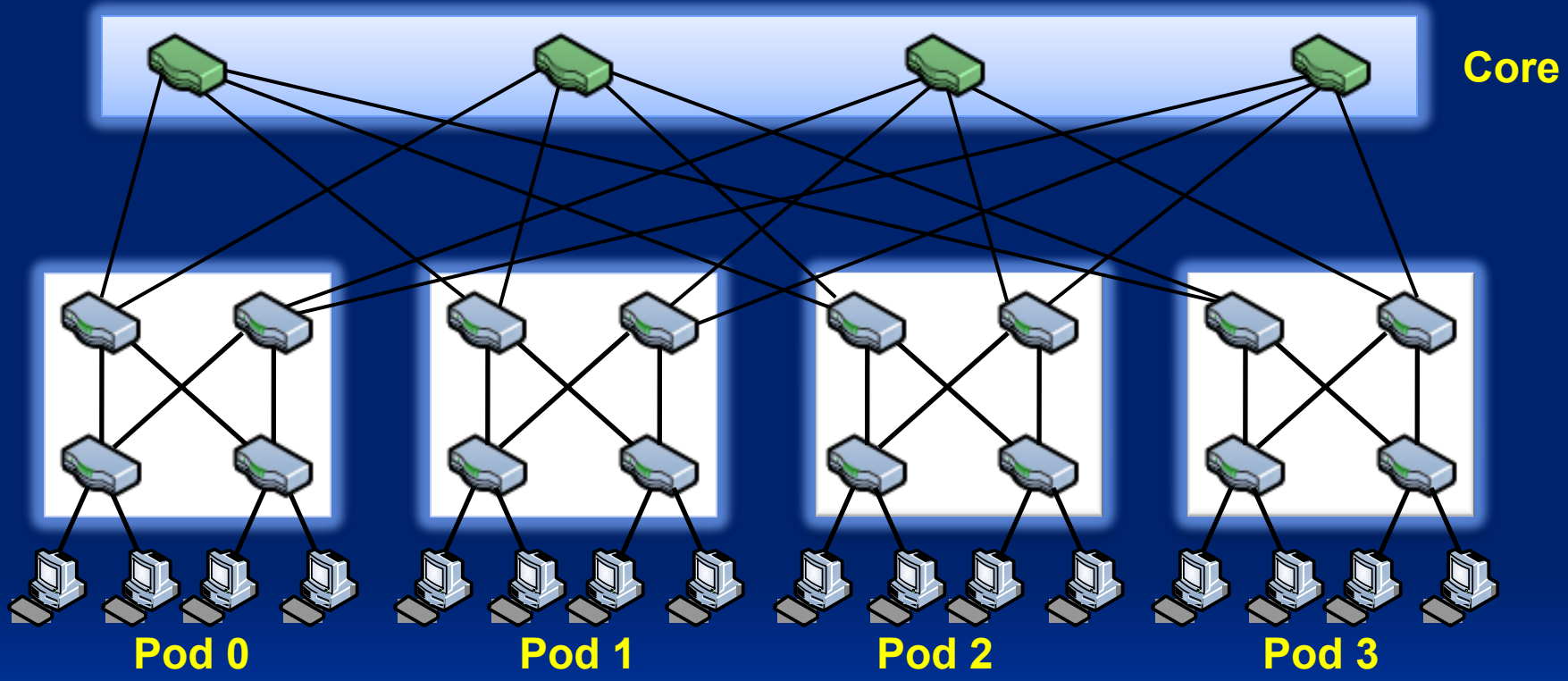
- Support 16 hosts organized into 4 pods
  - Each pod is a 2-ary 2-tree
  - Full bandwidth among hosts directly connected to pod

Scalability Using Identical Network Elements



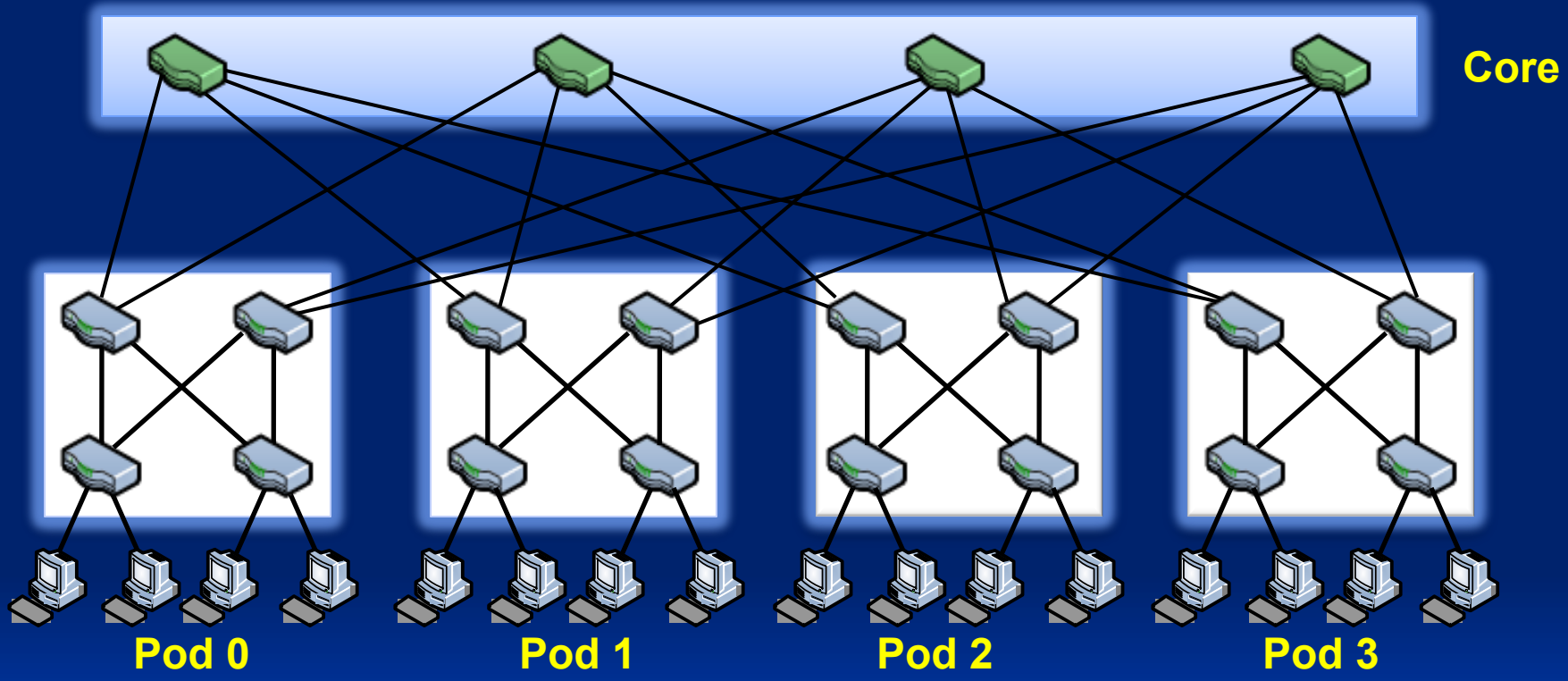
- Full bisection bandwidth at each level of fat tree
  - Rearrangeably Nonblocking
  - Entire fat-tree is a 2-ary 3-tree

Scalability Using Identical Network Elements



- *(5k²/4) k*-port switches support *k³/4* hosts
  - 48-port switches: 27,648 hosts using 2,880 switches
- Critically, approach scales to 10 GigE at the edge

Scalability Using Identical Network Elements



- Regular structure simplifies design of network protocols
- Opportunities: performance, cost, energy, fault tolerance, incremental scalability, etc.

Why Hasn't This Done Before?

- Needs to be backward compatible with IP/Ethernet
  - Existing routing and forwarding protocols do not work for fat tree
  - Scalability challenges with millions of end points
- Management
  - Thousands of individual elements that must be programmed individually
- Cabling explosion at each level of the fat tree
  - Tens of thousands of cables running across data center?

Our Work

- Switch Architecture [SIGCOMM 08]
- Cabling, Merchant Silicon [Hot Interconnects 09]
- Virtualization, Layer 2, Management [SIGCOMM 09]
- Routing/Forwarding [ongoing]
- Energy, Optics [ongoing]

- Take advantage of regularity of fat tree structure to simplify protocol design and improve performance

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

PortLand Goals

- Single Layer 2 fabric to 100k ports, 1M end hosts
  - VM migration while maintaining IP address, external connectivity and session state
- Toward zero configuration at deployment
  - No subnet, IP address, wiring information ,etc.
- No forwarding loops
- Rapid and efficient failure detection
- Native multicast support
- First class support for multipath forwarding

System Comparison

System	Topology	Forwarding		Routing	ARP	Loops	Multicast
		Switch State	Addressing				
TRILL	General	O(# of global hosts)	Flat: MAC in MAC encap	Switch broadcast	Switch maps MAC address to remote switch	TRILL header with TTL	ISIS, extensions based on MOSPF
SEATTLE	General	O(# of global hosts)	Flat	Switch broadcast	One-hop DHT	Unicast loops possible	New construct: groups
PortLand	Multi-rooted tree	O(# of ports)	Hierarchical	LDP: Fabric Manager for faults	Fabric Manager	Provably loop free; no extra encap	Broadcast free routing; Multi-rooted spanning trees

Design Overview

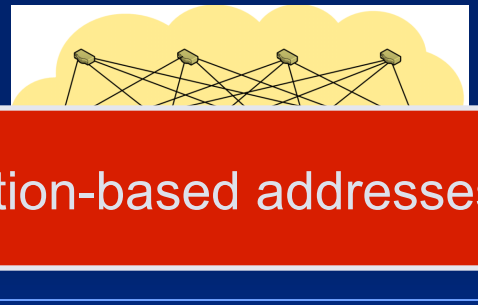
- Separate node location from node identifier
  - Host IP address: node identifier
  - *Pseudo MAC* (PMAC): node location
- *Fabric Manager*
  - Maintains IP→PMAC mapping
  - Facilitates fault tolerance
- PMAC sufficient to perform positional *forwarding*
- Location Discovery Protocol (LDP) for decentralized, zero-configuration *routing* and *forwarding*

Layer 2 versus Layer 3 Data Center Fabrics

Technique	Plug and play	Scalability	Small Switch State	Seamless VM Migration
Layer 2: Flat MAC Addresses	+	-	-	+
Layer 3: IP Addrs	-	~	+	-

Layer 2 versus Layer 3 Data Center Fabrics

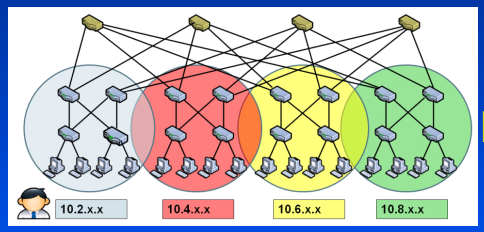
Technique	Plug and play	Scalability	Small Switch State	Seamless VM Migration
Layer 2: Flat MAC	+	-	-	+
Layer 3: IP Addrs	-	~	+	-



Host MAC Address	Out Port
9a:57:10:ab:6e	1

No change

Location-based addresses mandate manual configuration



LSR – Broadcast based

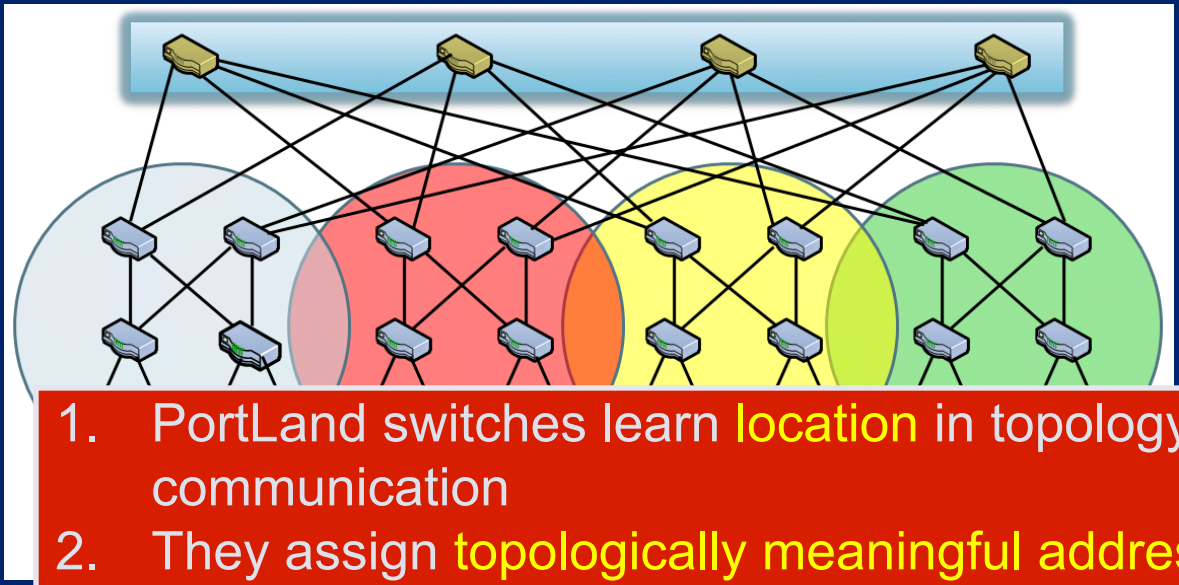
Host IP Address	Out Port
10.2.x.x	0
10.4.x.x	1

IP endpoint changes

Cost Consideration: Flat vs. Hierarchical Addresses

- Commodity switches today have ~640 KB of low latency, power hungry, expensive on chip memory
  - Stores 32 – 64 K flow entries
- 10 million virtual endpoints in 500k servers in data center
- Flat addresses → 10 million address mappings → ~100 MB on chip memory → ~150 times the memory size that can be put on chip today
- Location based addresses → 100 – 1000 address mappings→ ~10 KB of memory → easily accommodated in switches today

PortLand: Plug and Play + Small Switch State



\+ Pair-wise communication

1. PortLand switches learn location in topology using pair-wise communication
2. They assign topologically meaningful addresses to hosts using their location

PMAC Address	Out Port
0:2:x:x:x:x	0
0:4:x:x:x:x	1
0:6:x:x:x:x	2
0:8:x:x:x:x	3

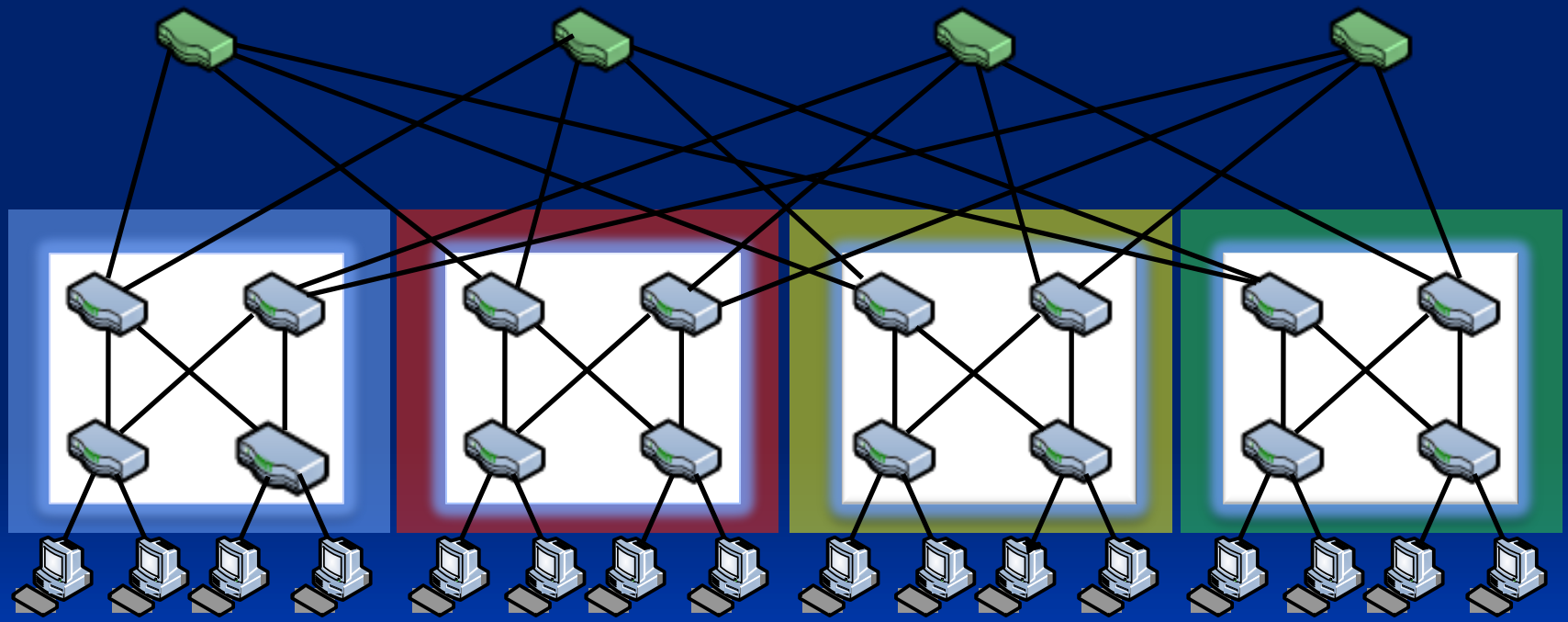
- 10 million virtual endpoints in 500,000 servers in data center

- 100 – 1000 address mappings ➔ ~10 KB of memory ➔ easily accommodated in switches today

PortLand: Layer 2 Scalability Challenges

Challenge	State Of Art
Address Resolution	Broadcast based
Routing	Broadcast based
Forwarding	Large switch state

Layering Hierarchy On A Multi-Rooted Tree



POD 0

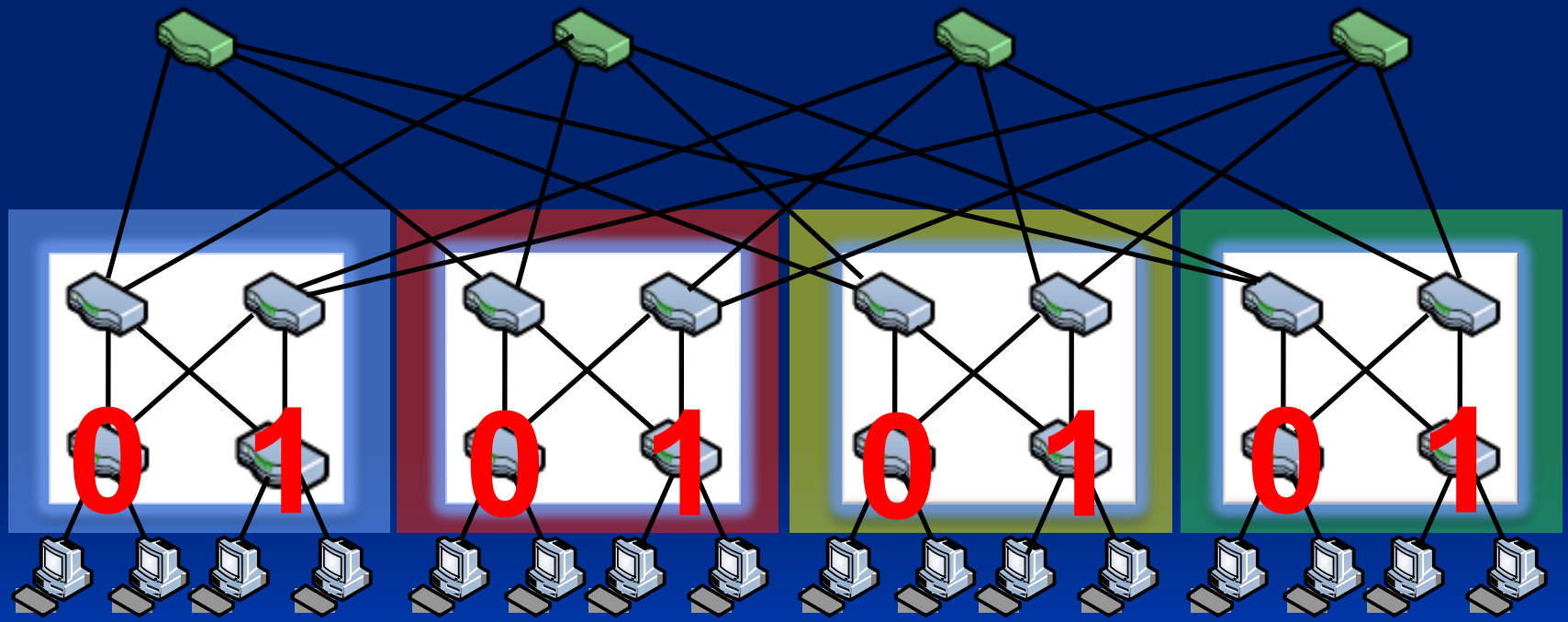
POD 1

POD 2

POD 3

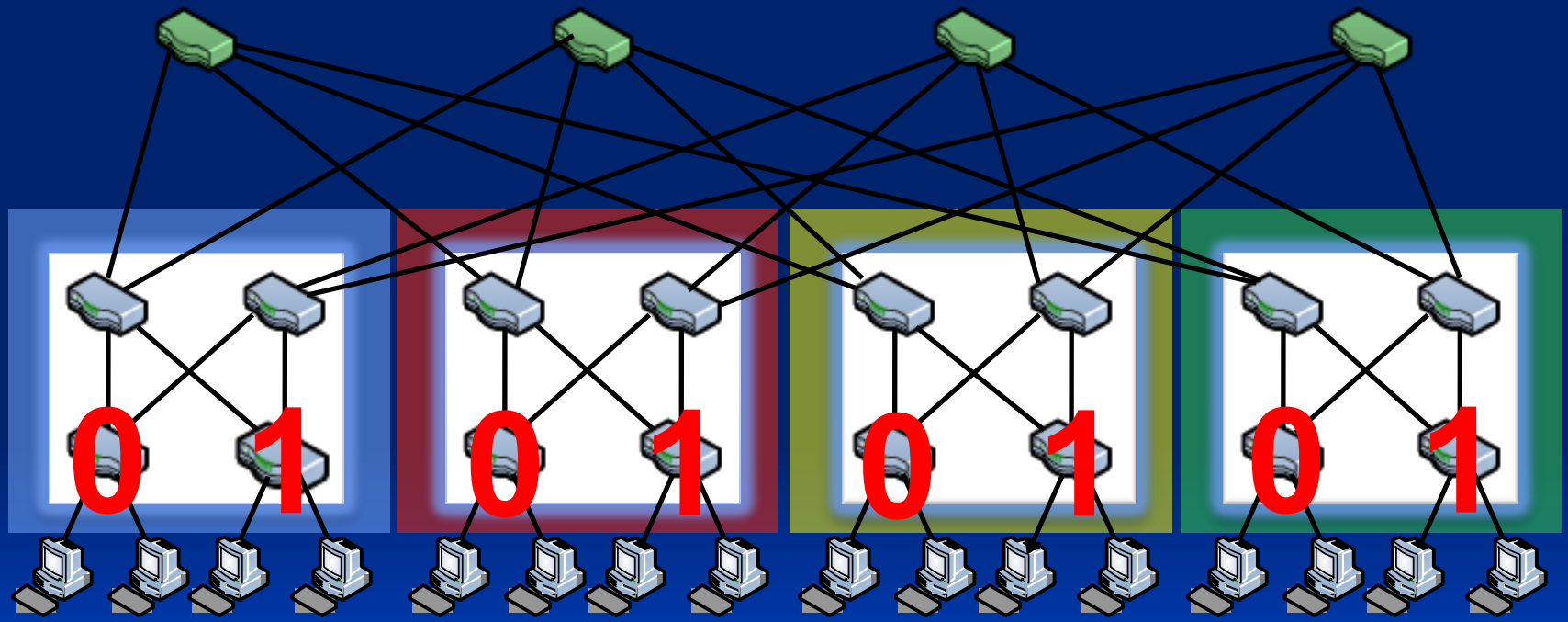
Pod Number

Layering Hierarchy On A Multi-Rooted Tree



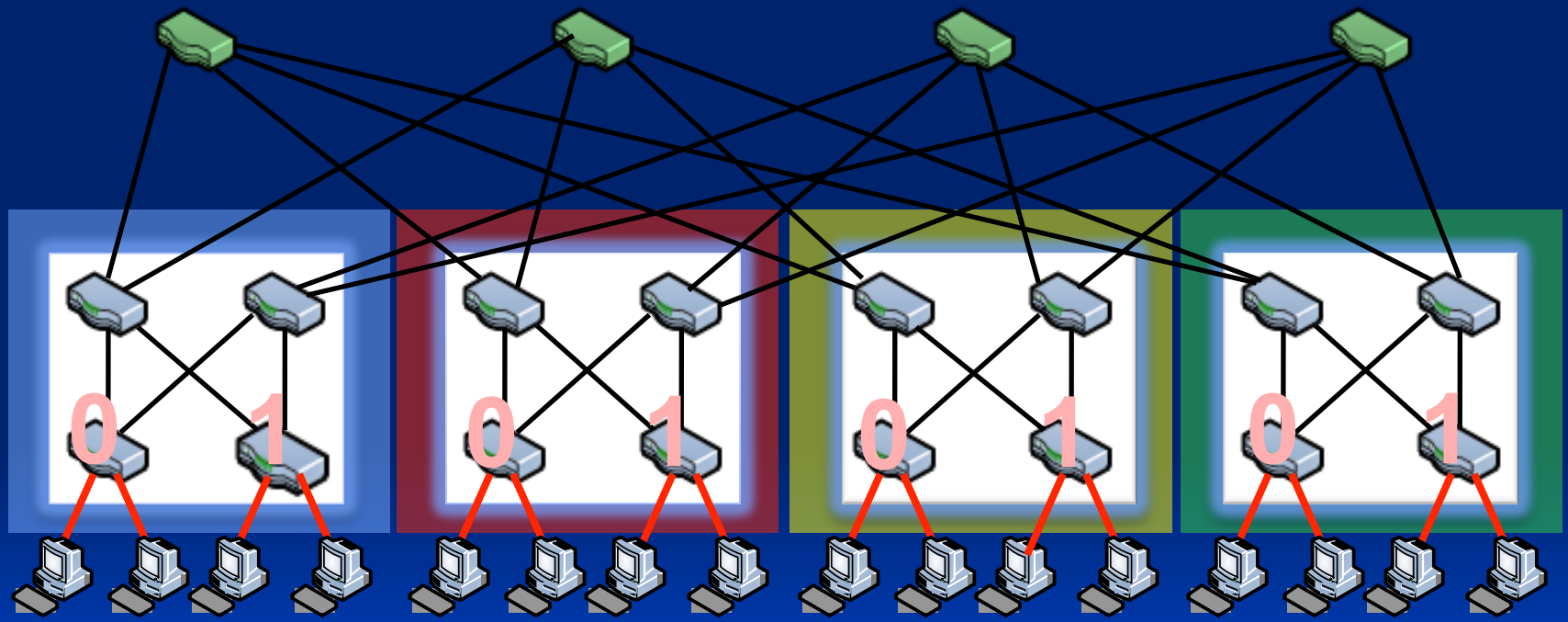
Position Number

Layering Hierarchy On A Multi-Rooted Tree



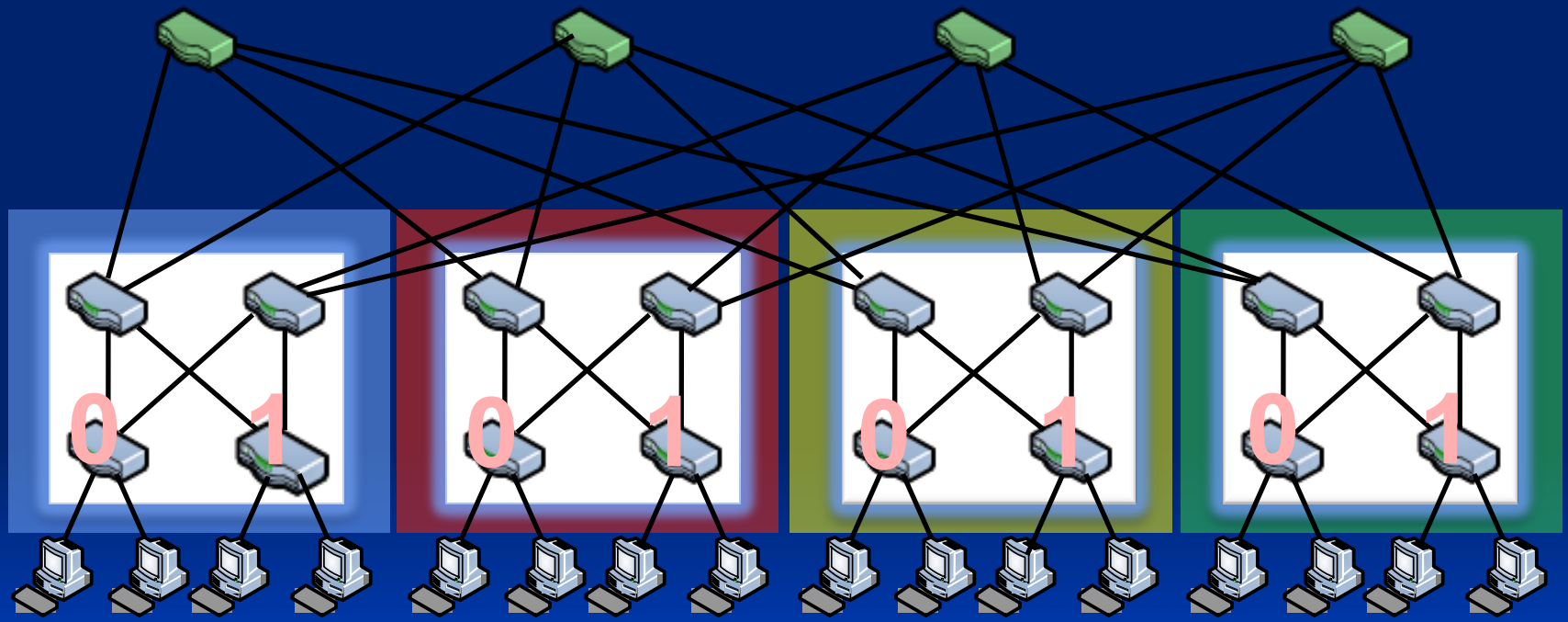
PMAC: pod.position.port.vmid

Layering Hierarchy On A Multi-Rooted Tree



PMAC: pod.position.port.vmid

Layering Hierarchy On A Multi-Rooted Tree



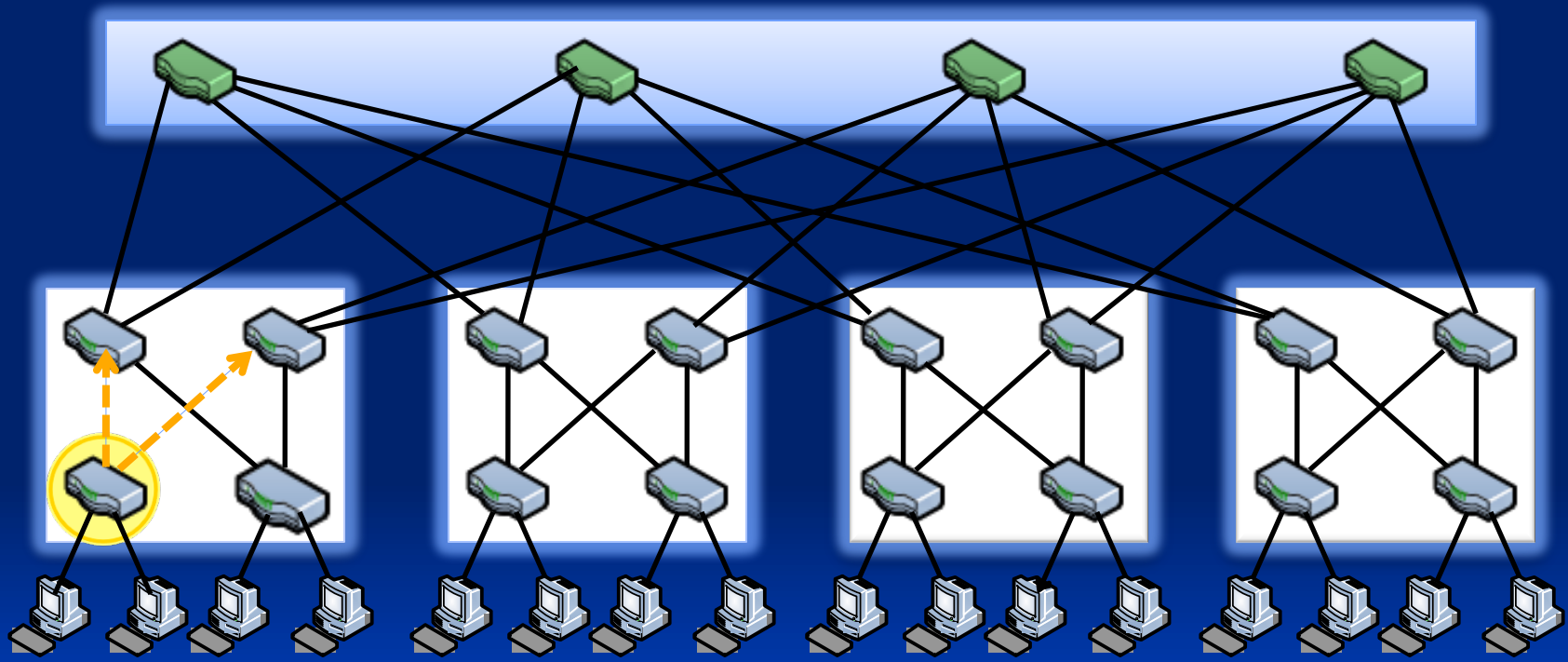
PMAC: pod.position.port.vmid

PortLand: Location Discovery Protocol

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

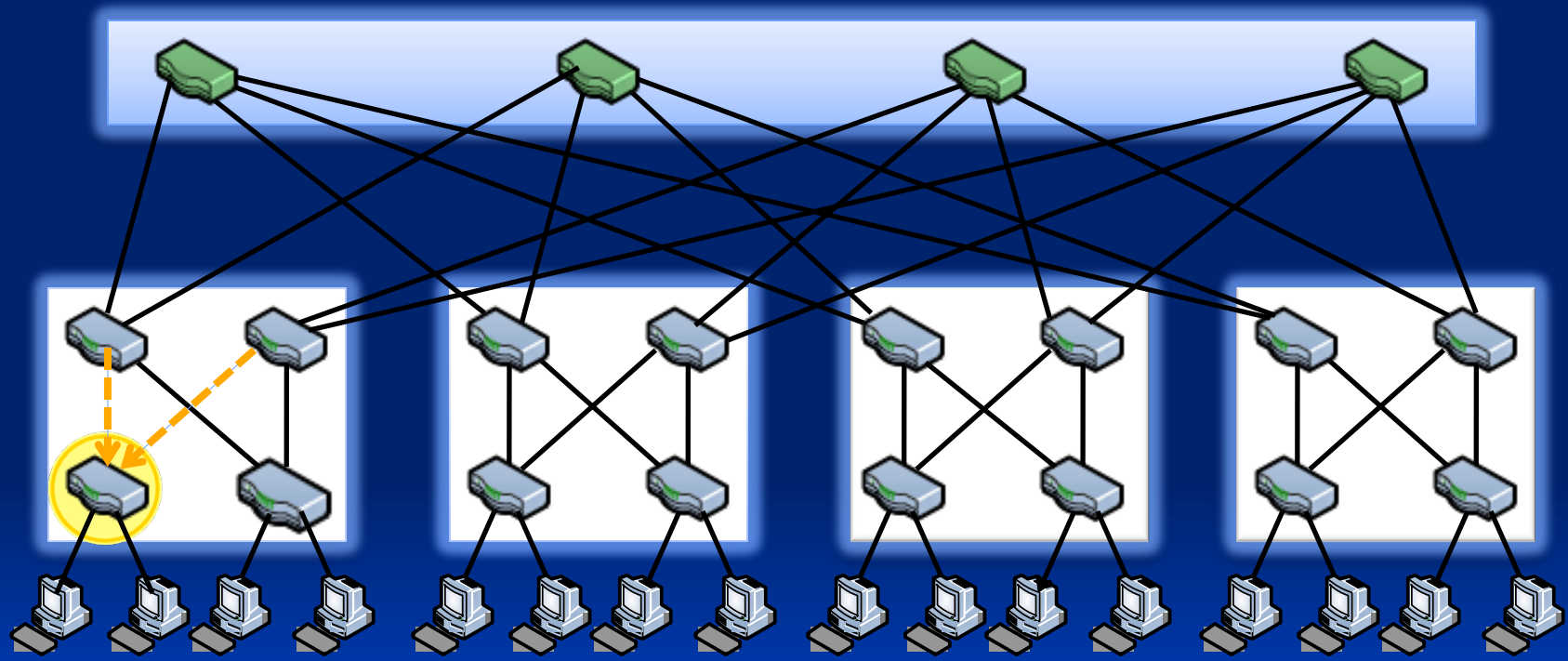
Location characteristic	Technique
1) Tree level / Role	Based on neighbor identity
2) Pod number	Aggregation and edge switches agree on pod number
3) Position number	Aggregation switches help edge switches choose unique position number

PortLand: Location Discovery Protocol



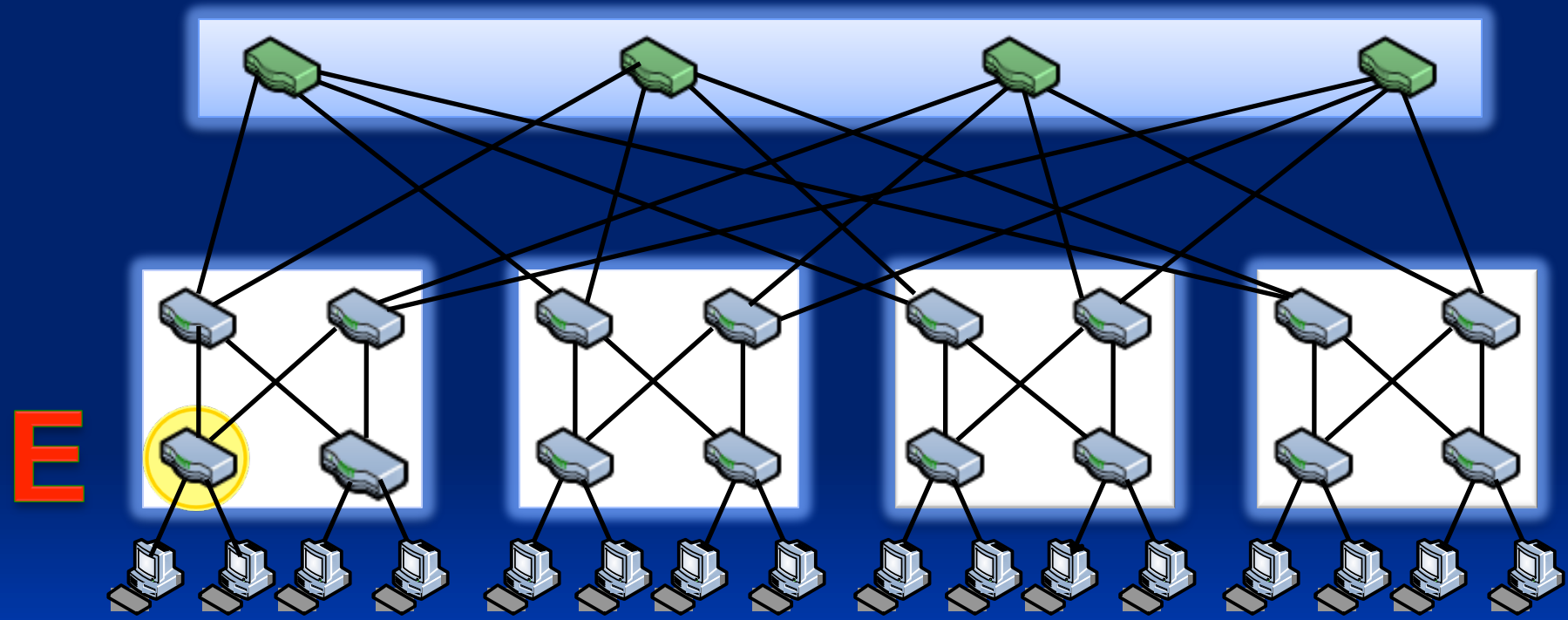
Switch Identifier	Pod Number	Position	Tree Level
A0:B1:FD:56:32:01	??	??	??

PortLand: Location Discovery Protocol



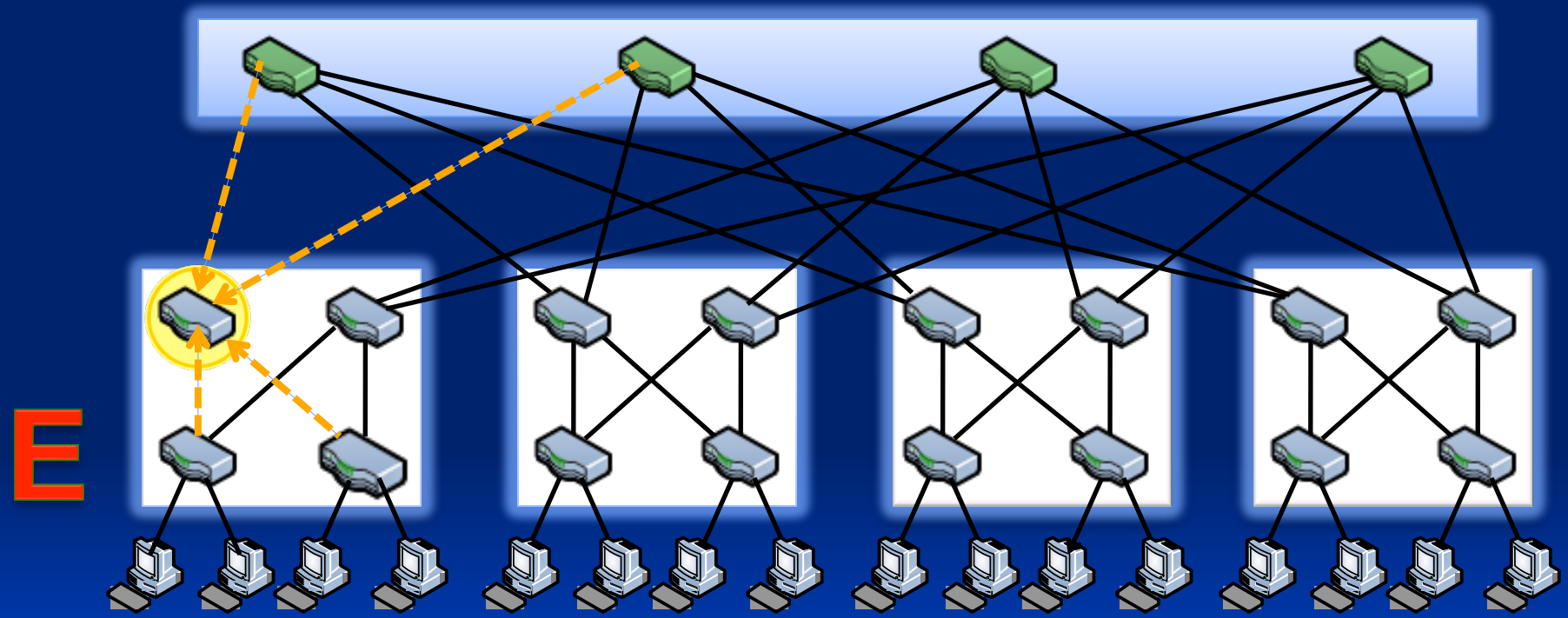
Switch Identifier	Pod Number	Position	Tree Level
A0:B1:FD:56:32:01	??	??	??

PortLand: Location Discovery Protocol



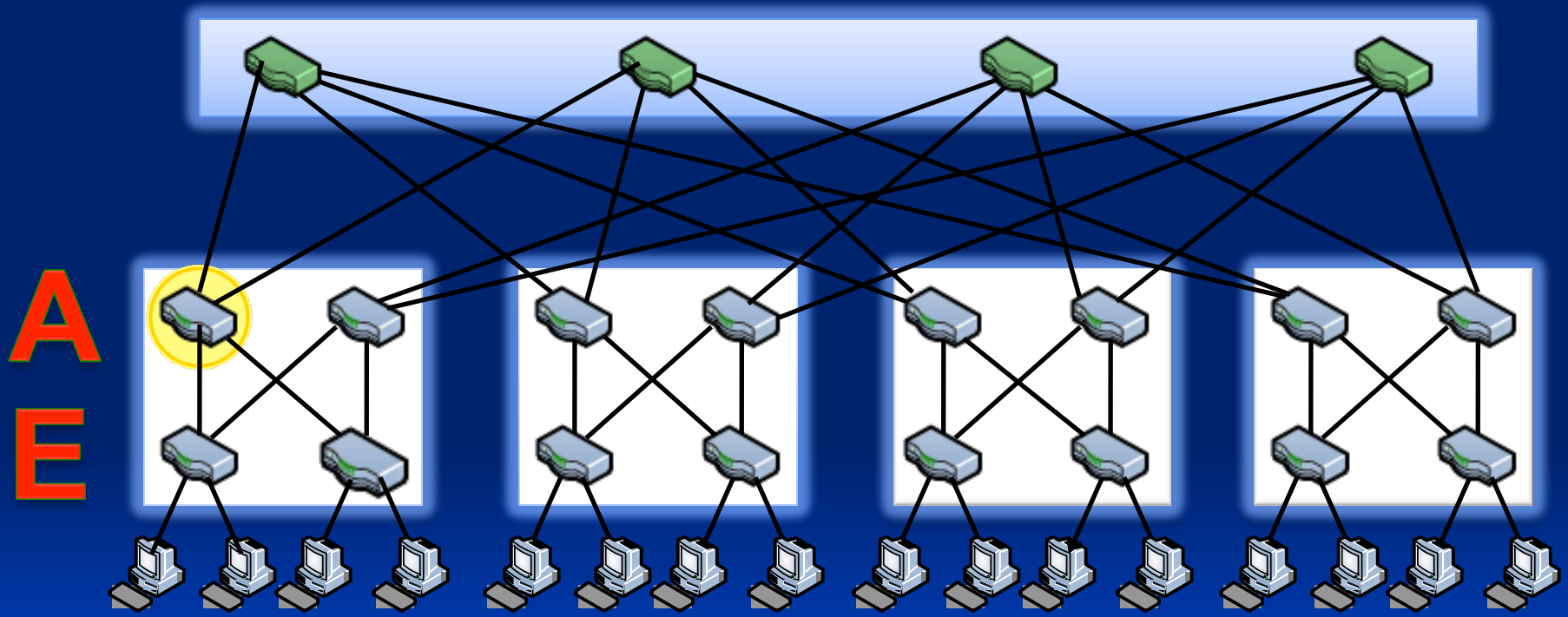
Switch Identifier	Pod Number	Position	Tree Level
A0:B1:FD:56:32:01	??	??	0

PortLand: Location Discovery Protocol



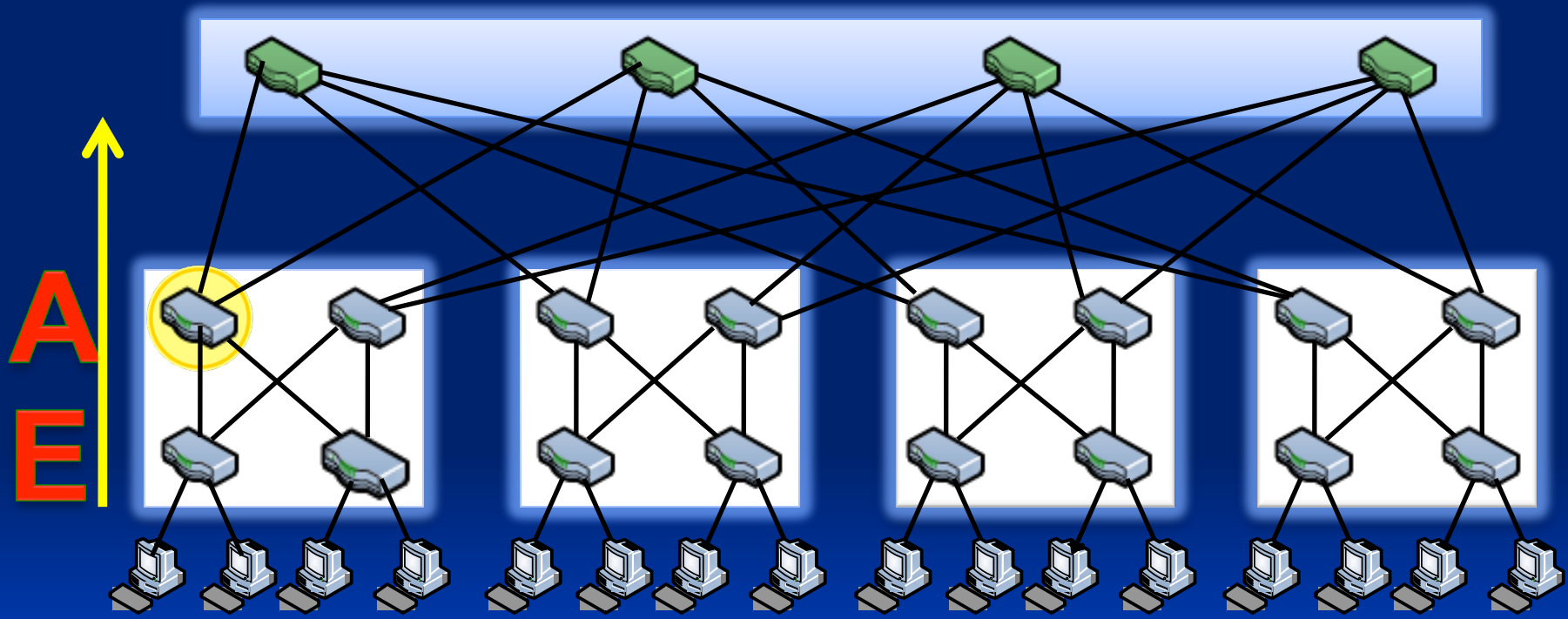
Switch Identifier	Pod Number	Position	Tree Level
B0:A1:FD:57:32:01	??	??	??

PortLand: Location Discovery Protocol



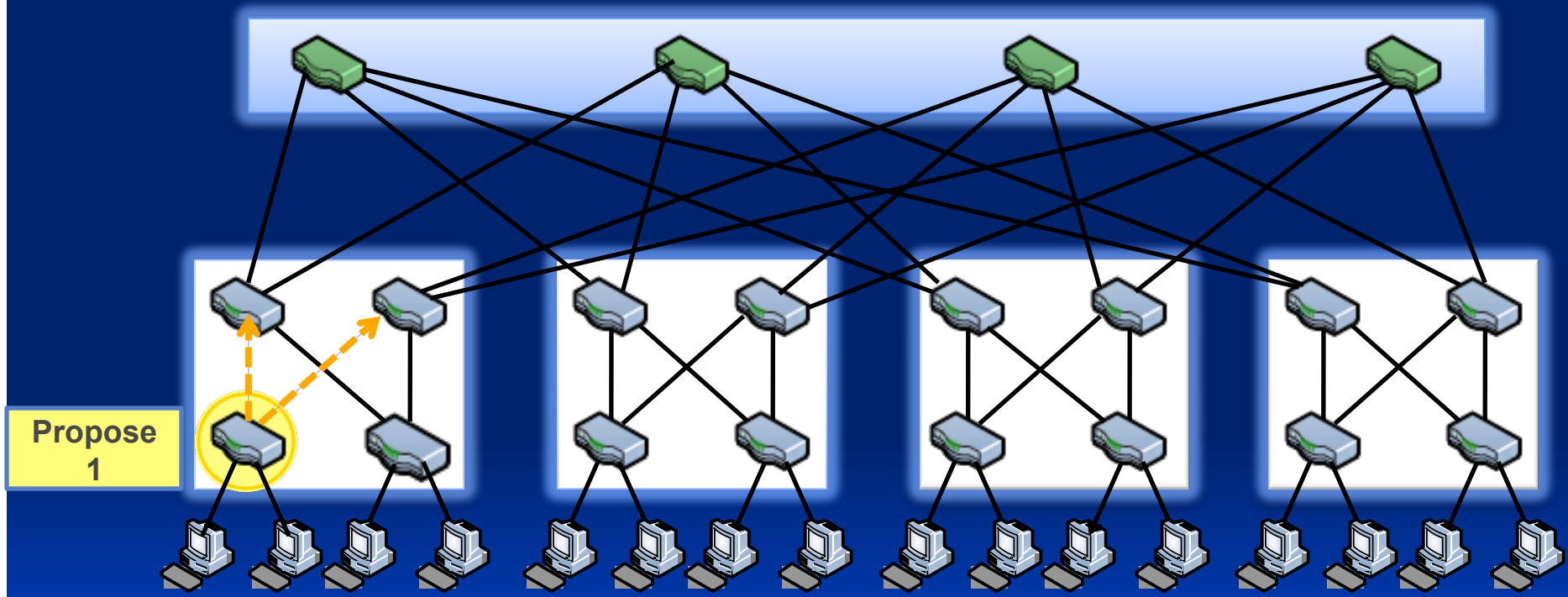
Switch Identifier	Pod Number	Position	Tree Level
B0:A1:FD:57:32:01	??	??	1

PortLand: Location Discovery Protocol



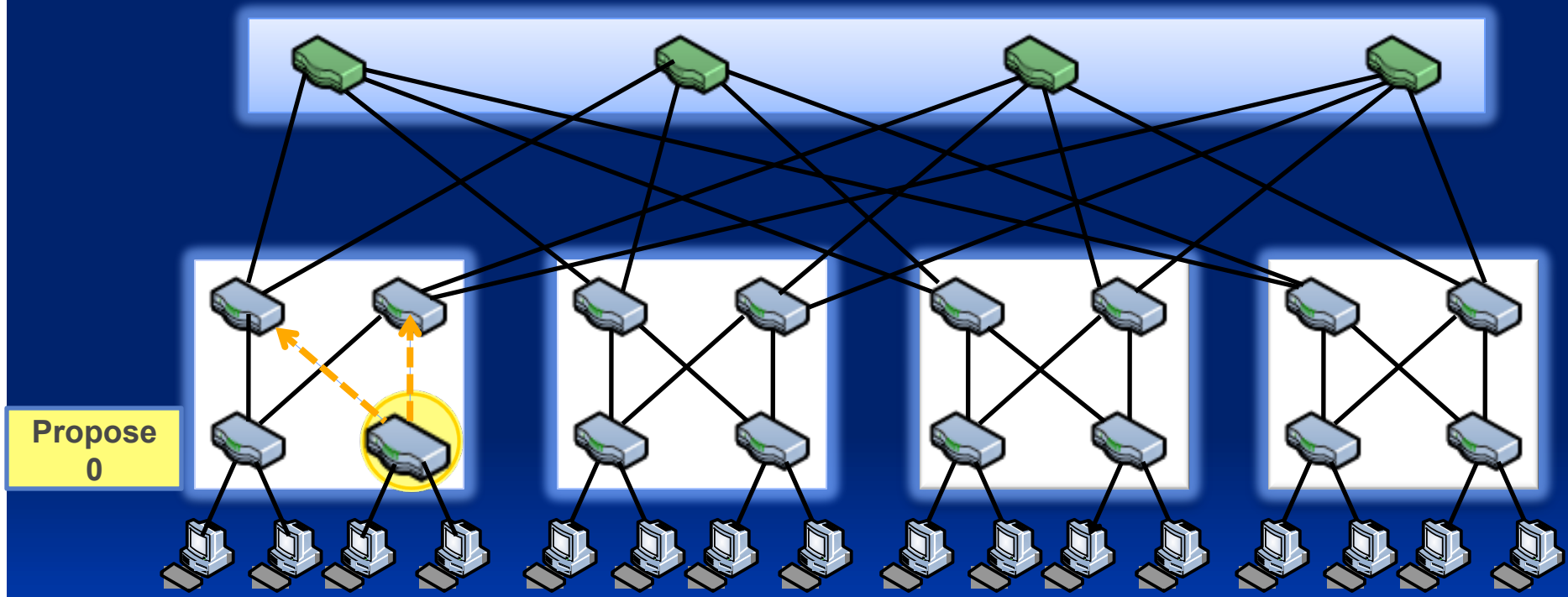
Switch Identifier	Pod Number	Position	Tree Level
B0:A1:FD:57:32:01	??	??	1

PortLand: Location Discovery Protocol



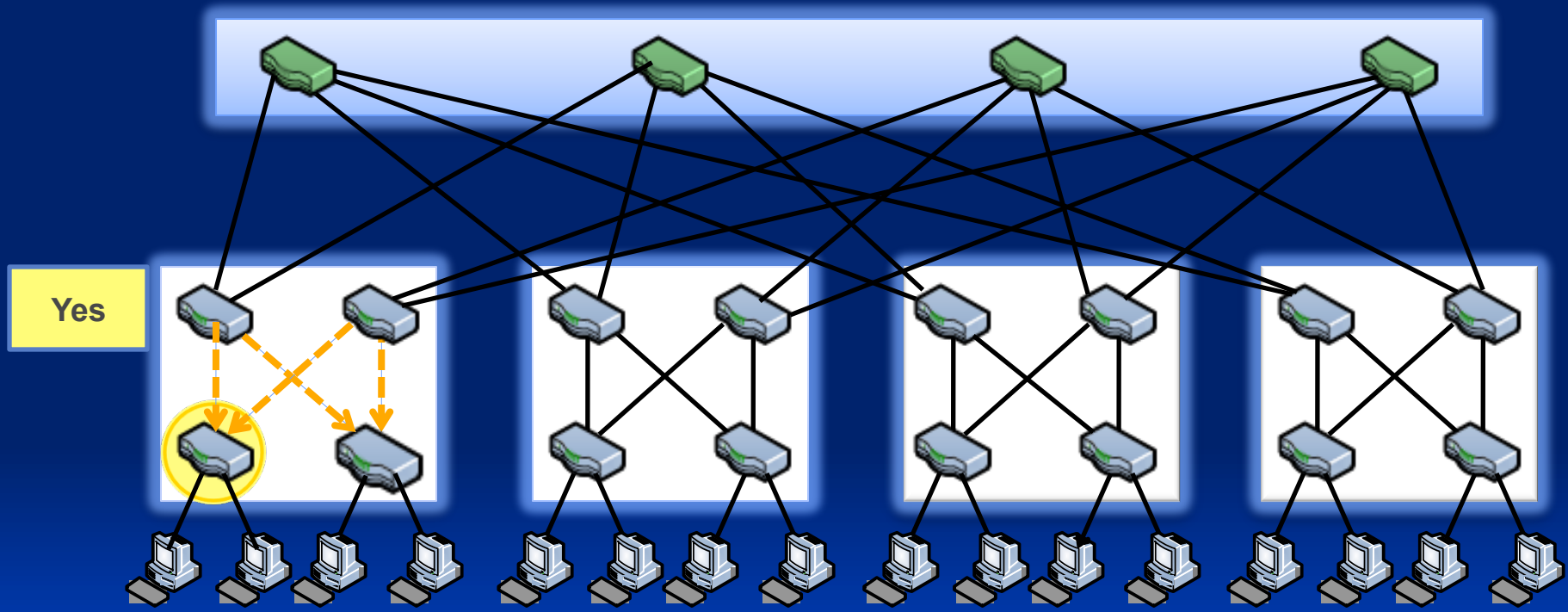
Switch Identifier	Pod Number	Position	Tree Level
A0:B1:FD:56:32:01	??	??	0

PortLand: Location Discovery Protocol



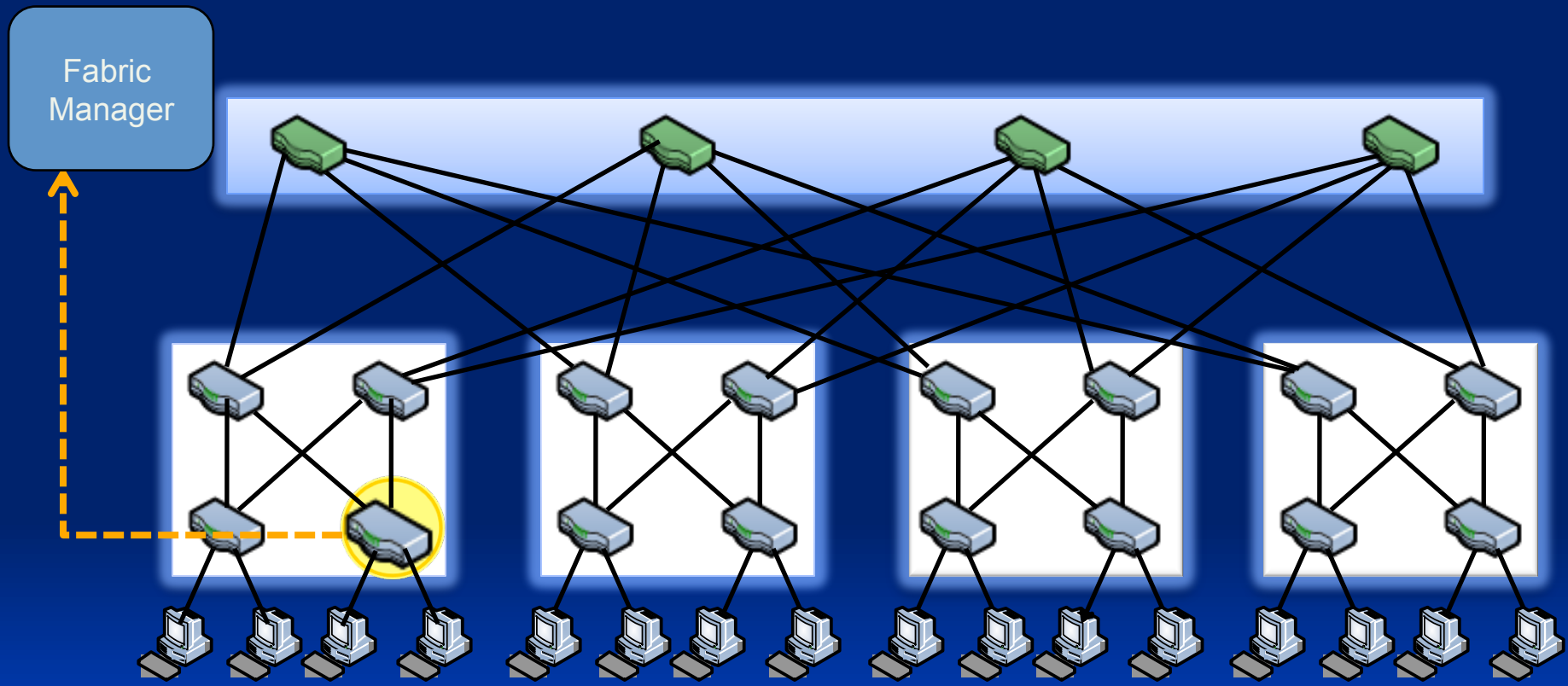
Switch Identifier	Pod Number	Position	Tree Level
D0:B1:AD:56:32:01	??	??	0

PortLand: Location Discovery Protocol



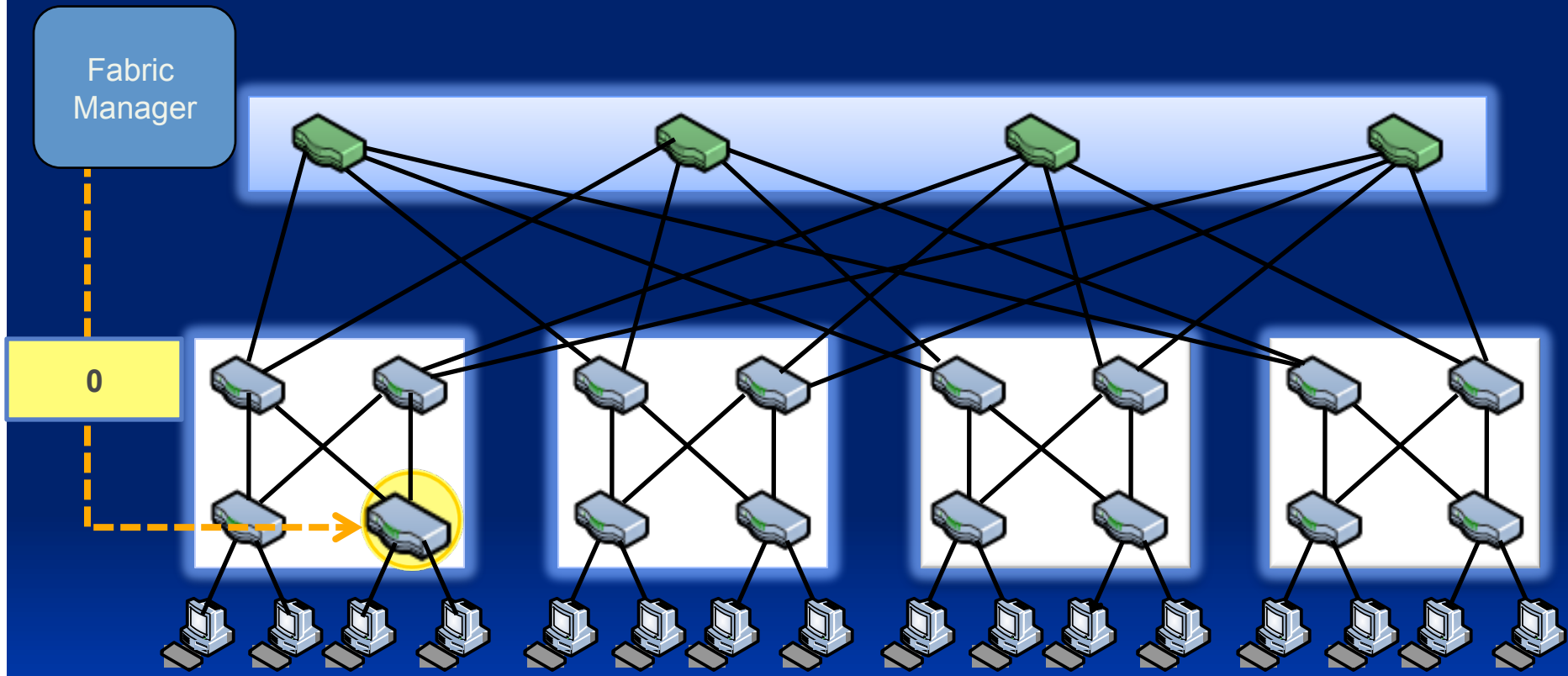
Switch Identifier	Pod Number	Position	Tree Level
A0:B1:FD:56:32:01	??	1	0

PortLand: Location Discovery Protocol



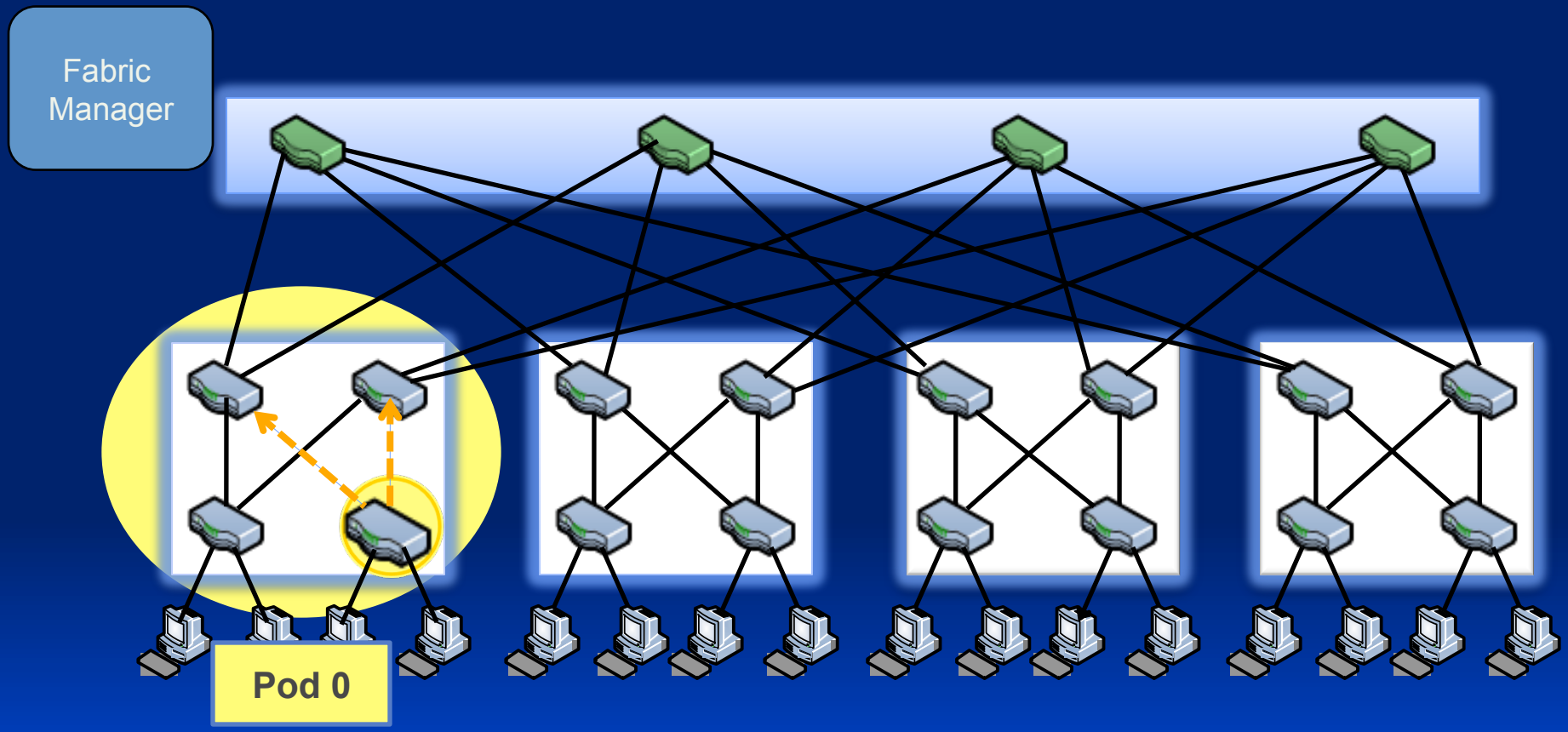
Switch Identifier	Pod Number	Position	Tree Level
D0:B1:AD:56:32:01	??	0	0

PortLand: Location Discovery Protocol



Switch Identifier	Pod Number	Position	Tree Level
D0:B1:AD:56:32:01	??	0	0

PortLand: Location Discovery Protocol

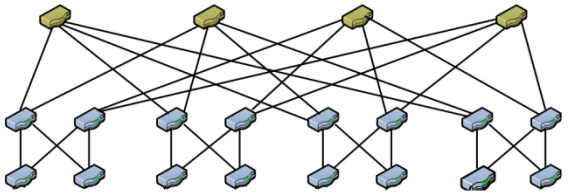


Switch Identifier	Pod Number	Position	Tree Level
D0:B1:AD:56:32:01	0	0	0

PortLand: Fabric Manager

Fabric Manager

IP	Pseudo MAC
10.5.1.2	00:00:01:02:00:01
10.2.4.5	00:02:00:02:00:01



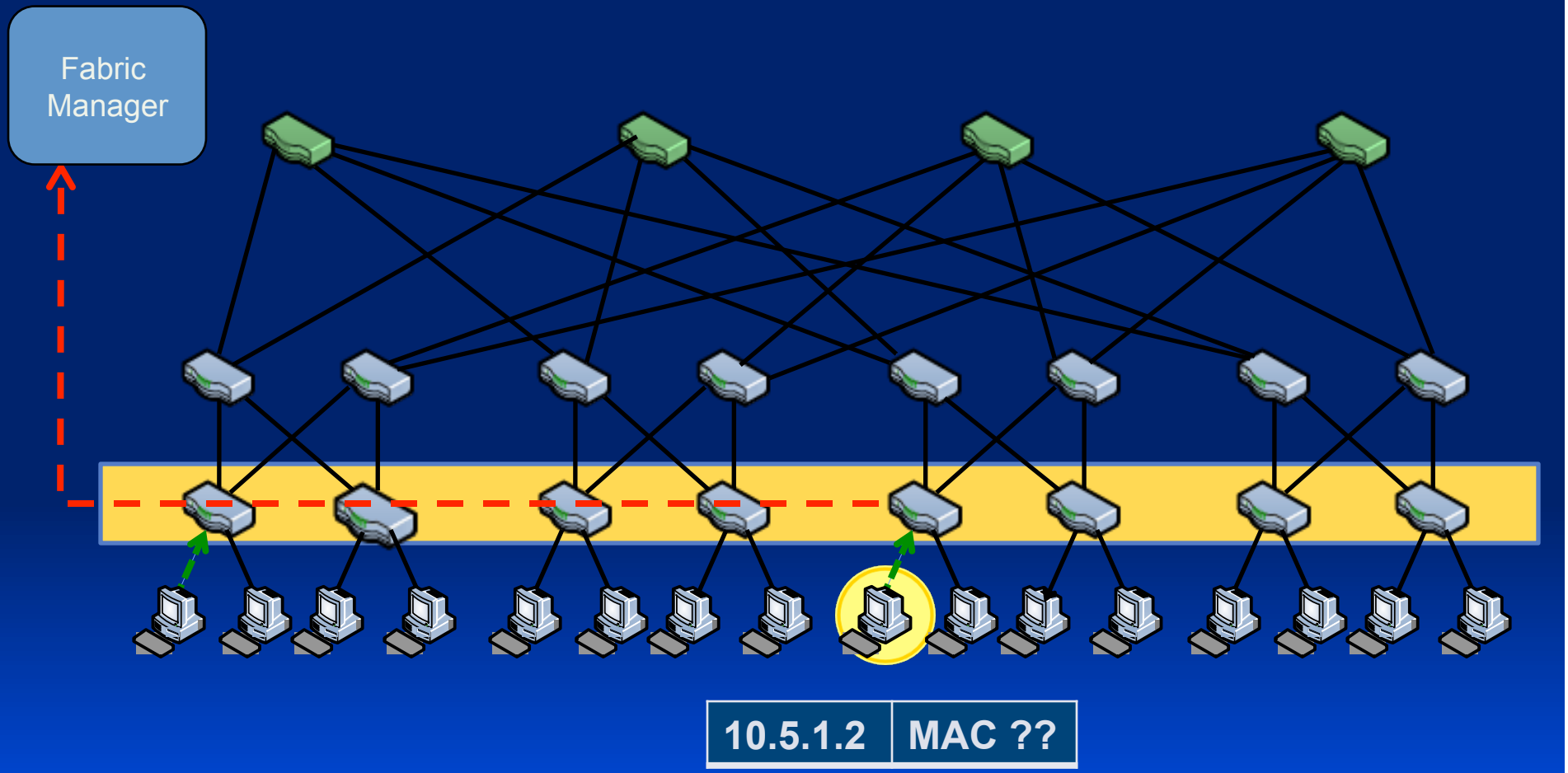
ARP mappings

Network map

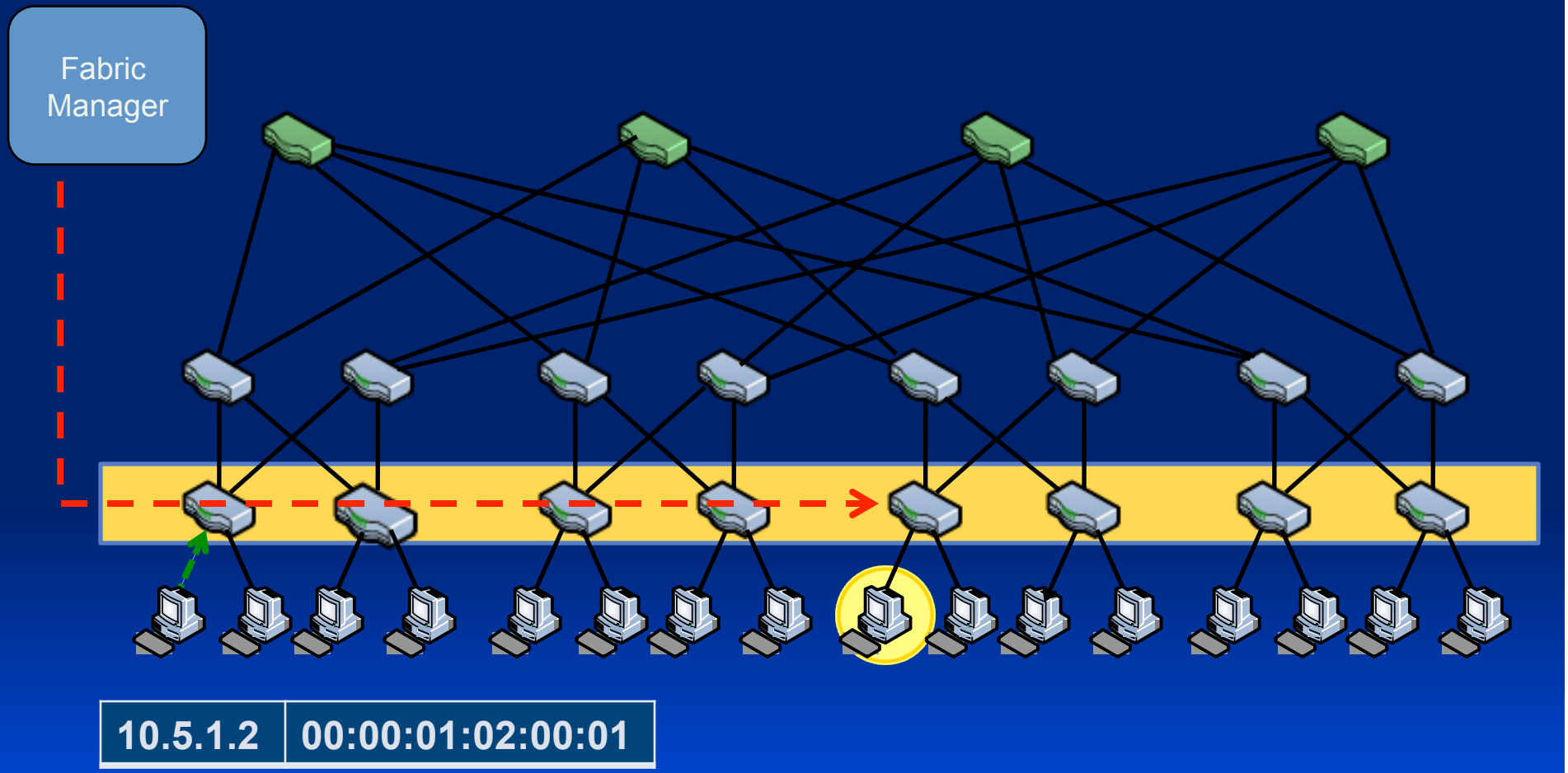
Soft state

~~Administrator configuration~~

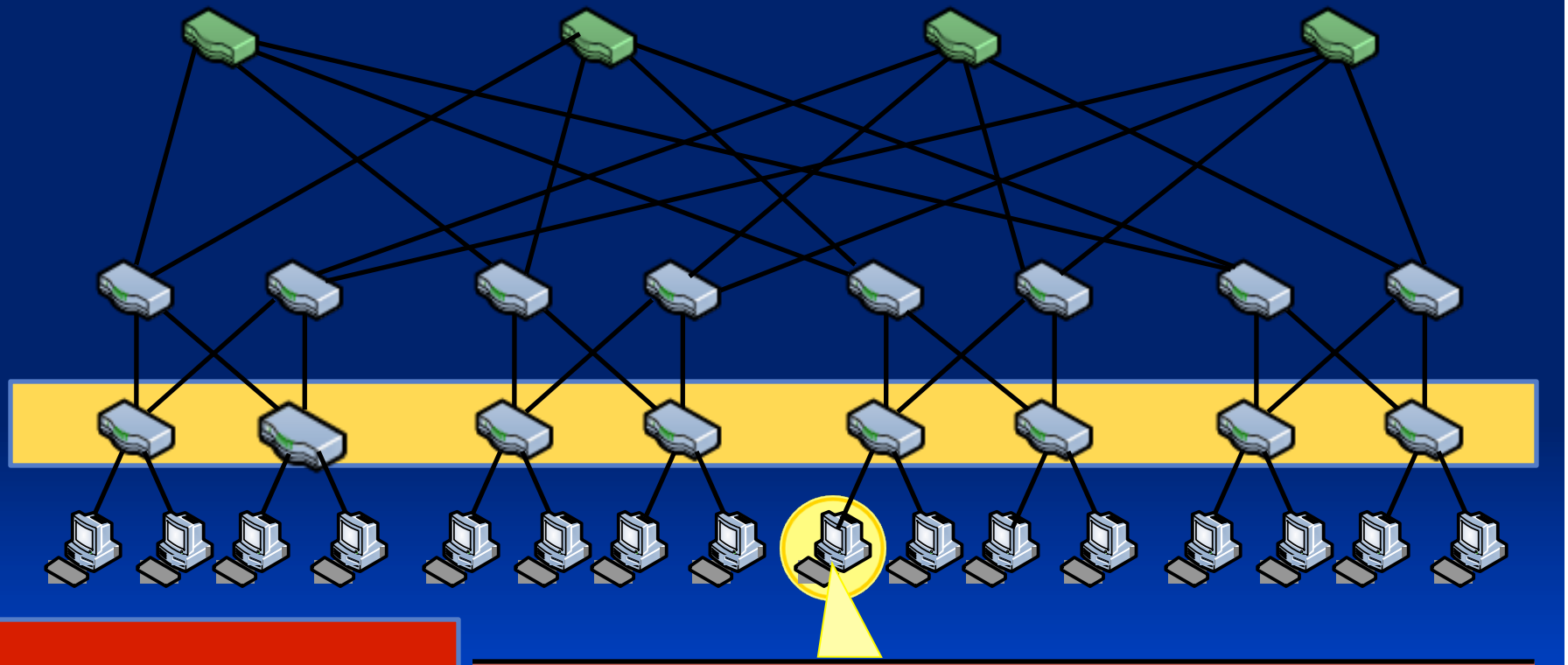
PortLand: Name Resolution



PortLand: Name Resolution



PortLand: Name Resolution



ARP replies contain only PMAC

Address	HWtype	HWAddress	Flags	Mask	Iface
10.5.1.2	ether	00:00:01:02:00:01	C		eth1

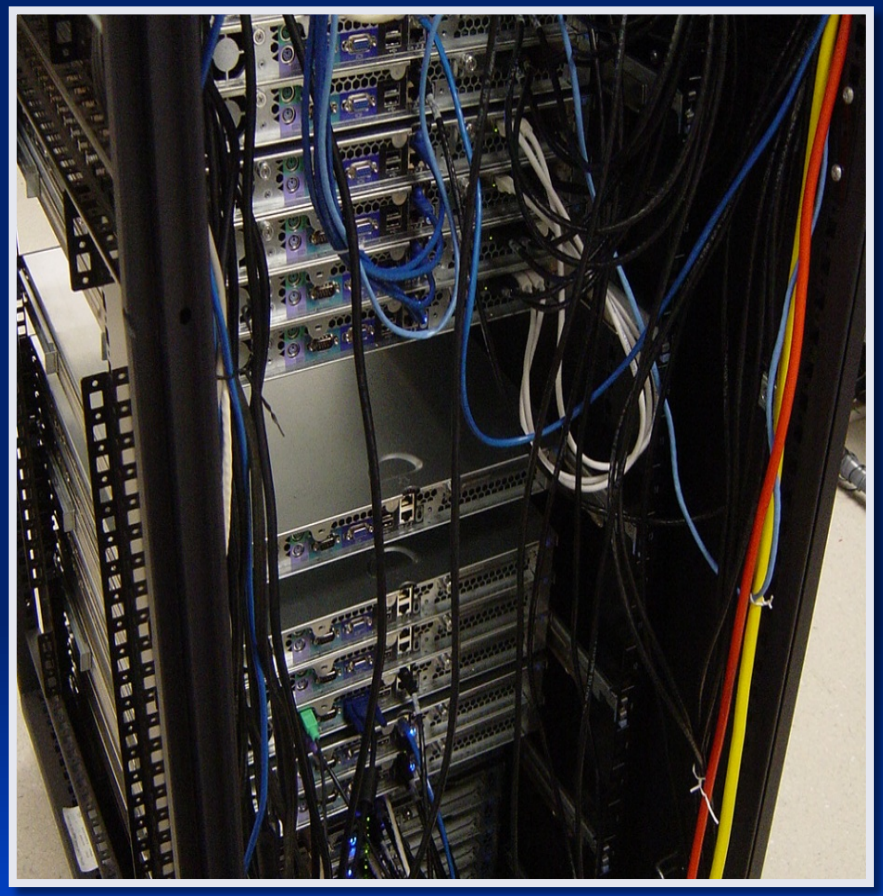
Design: Provably Loop Free Forwarding

- Up-Down semantics [AutoNet]
- Constraint: A switch must never forward a packet along an upward facing port when the ingress port for that packet is also an upward facing port
- Traffic can never change its direction more than once

Broadcast Free Routing Protocol

- No Link State Routing
- Liveness Monitoring by LDP exchanges
- Switches convey fault information to Fabric Manager
- Fabric Manager informs affected switches of failure

PortLand Prototype



- 20 OpenFlow NetFPGA switches
- TCAM + SRAM for flow entries
- Software MAC rewriting
- 3 tiered fat-tree
- 16 end hosts
- Implementations underway for HP, Broadcom, and Fulcrum switches

PortLand: Evaluation

Measurements	Configuration	Results
Network convergence time	Keepalive frequency = 10 ms Fault detection time = 50 ms	65 ms
TCP convergence time	RTO_{min} = 200ms	~200 ms
Multicast convergence time		110ms
TCP convergence with VM migration	RTO_{min} = 200ms	~200 ms – 600 ms
Control traffic to fabric manager	27,000+ hosts, 100 ARPs / sec per host	400 Mbps
CPU requirements of fabric manager	27,000+ hosts, 100 ARPs / sec per host	70 CPU cores

Related Work

- Floodless in SEATTLE
  - Lookup scheme to enable layer 2 routing based on DHT
- DCell
  - Alternative topology for data center interconnect
  - Fewer modifications to switch, but end host modifications required
  - Topology is not rearrangeably nonblocking
- Rbridges/TRILL
  - Layer 2 routing protocol with MAC in MAC encapsulation

Related Work

- Much work in interconnection networks comes from Massively Parallel Processing (MPP) world
  - Many supercomputers are organized as fat-trees
  - Thinking Machines CM-5, SGI Altix, etc.
- Specialized networks targeting HPC clusters deploy fat-tree topologies
  - InfiniBand, Myrinet, Quadrics
  - Most use source routing

Host Centric vs. Network Centric Approach

VL2

- Network architecture that scales to support huge data centers
- Layer 3 routing fabric used to implement a virtual layer 2
- Scale Layer 2 via end host modifications
  - Unmodified switch hardware and software
  - End hosts modified to perform enhanced resolution to assist routing and forwarding

PortLand

- Modify network fabric to
  - Work with arbitrary operating systems and virtual machine monitors
  - Maintain the boundary between network and end-host administration
- Scale Layer 2 via network modifications
  - Unmodified switch hardware and end hosts

Conclusions

- Data Center Network Requirements
  - Cost and absolute performance
  - Packaging
  - Energy/heat
  - Fault tolerance
- Fat-tree as basis for delivering on above challenges
  - Regularity of the structure allows simplifying assumptions to address challenges above
  - Fat-tree built from 48-port switches support 27k hosts
  - Holds promise for cost, performance, energy

Questions?

- For more information:
  - “A Scalable Commodity Data Center Network Architecture,” Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. *Proceedings of ACM SIGCOMM*, August 2008.
  - “Data Center Switch Architecture in the Age of Merchant Silicon,” Nathan Farrington and Amin Vahdat. *Proceedings of Hot Interconnects*, August 2009.
  - “PortLand: A Scalable Fault Tolerant Layer 2 Data Center Network Fabric,” Radhika Niranjan Mysore, Anreas Pamporis, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. *Proceedings of ACM SIGCOMM*, August 2009.
  - http://dcswitch.sysnet.ucsd.edu
  - http://idleprocess.wordpress.com