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

Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang,
Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat

Presented by Frank Austin Nothaft

Network management at scale

- O(10-100K) machines per datacenter:
    - O(40) machines per rack
        - O(20) racks per row
- Network is typically organized in a Fat-tree like topology —> deeply hierarchical
- Machines are virtualized

Network management at scale: What is expensive?

- Routing tables explode
- Lots of hardware —> config-by-human = bad
- Updates that need to be broadcast

Network management at scale: What do we want?

1. Easy VM migration
2. No admin involvement
3. Efficient communication between any two nodes
4. No loops
5. Efficient failure recovery

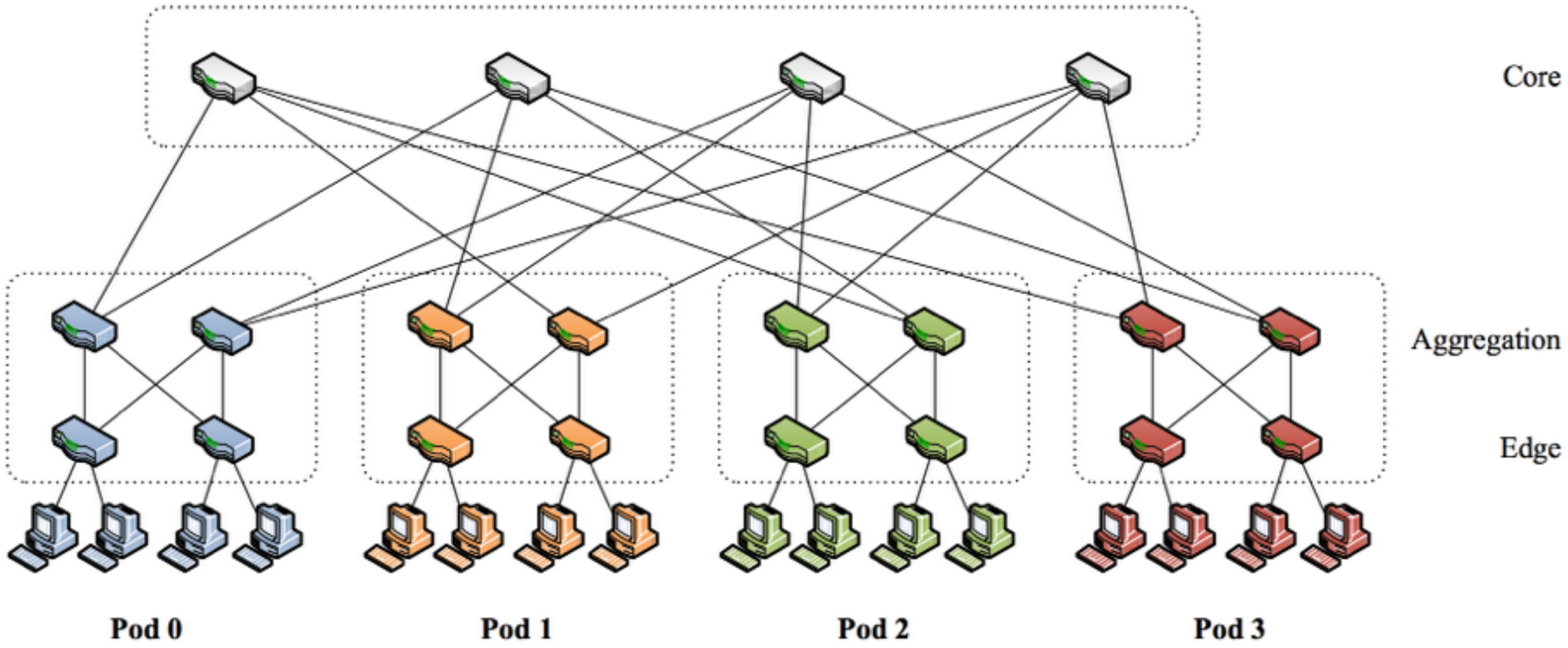
What are the implications of these aims?

R1: VM Migration	Cannot do at layer 3 —> breaks existing TCP connections
R2: No admin	Need single L2 fabric for whole DC Not compatible with R5 with current rout. protocols
R3: Any-to-any	Requires huge routing tables
R4: No loops	Can occur during routing protocol convergence Can avoid at L2, but either inefficient or incompat.
R5: Failure Recov.	Need to quickly update routing info Difficult with present protocols —> require bcast

PortLand's realization:

If topology is known,
problems are much simpler!

Datacenter Topology



- In a Fat Tree, bandwidth increases towards the core
- Variety of topologies, generally a multi-rooted tree

How can we exploit topology?

- We can solve some problems by minimizing distributed state, while solving others by centralizing state

- PortLand does the following two high-level optimizations:

1. Rewrite addresses:

    - If we can rewrite the MAC addresses of leaf nodes, we can greatly simplify routing tables

2. Offload management to centralized Fabric Controller:

    - Assists with address resolution, failover, etc.

Address Rewriting

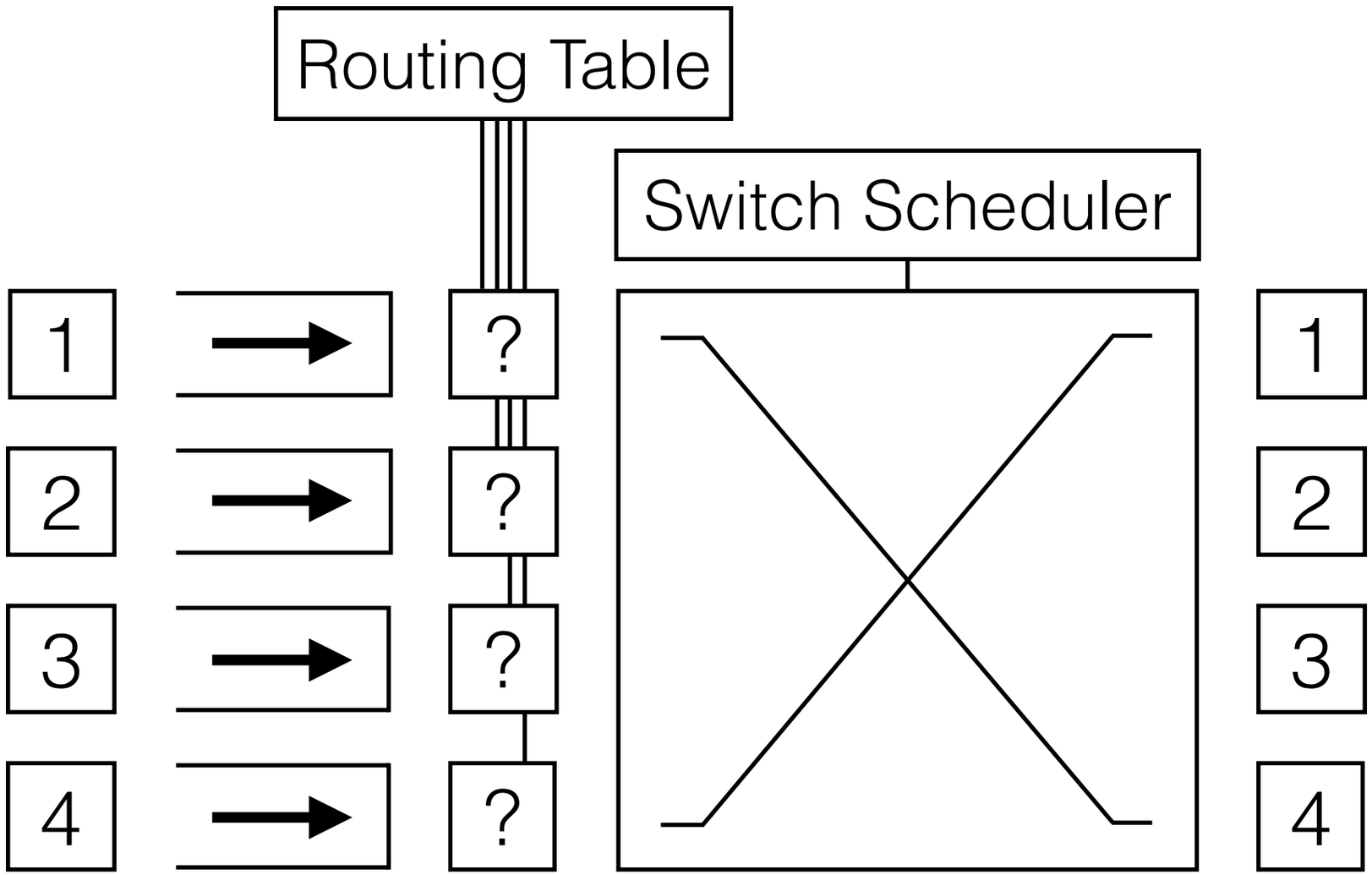
- MAC address —> 48 bit unique address for each endpoint, used for ethernet routing

- Issue: if MAC addresses are random, routing table of each switch grows O(n)

- PortLand uses knowledge of topology to rewrite MAC addresses at edge routers:

pod:position:port:vmid

Aside: Ethernet Switching



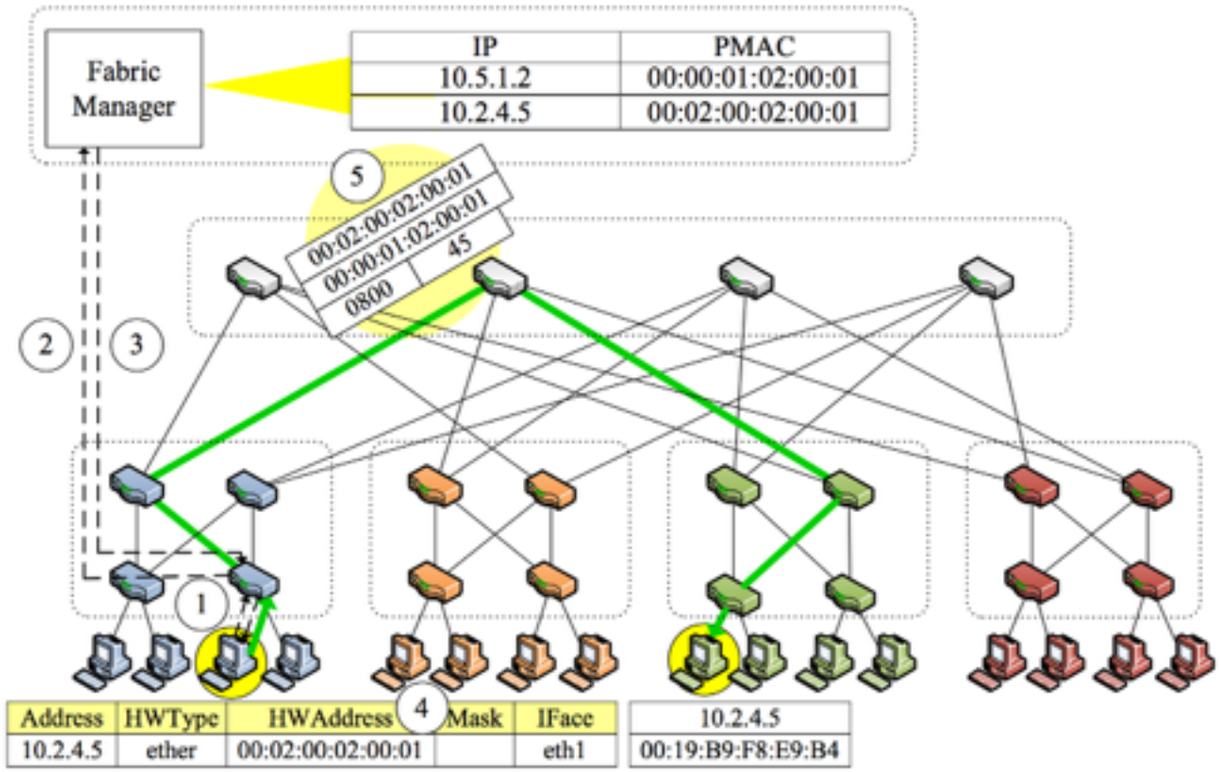
Routing table —> SRAM based CAM

Location Discovery

- Authors present distributed algo. for discovering switch location:
    - Each switch sends a message indicating port direction, nodes do not send messages
    - Insight: edge routers only receive messages from aggregation routers, aggregation routers will receive messages from edge routers on downwards facing ports
- Fabric manager assigns IDs to switches

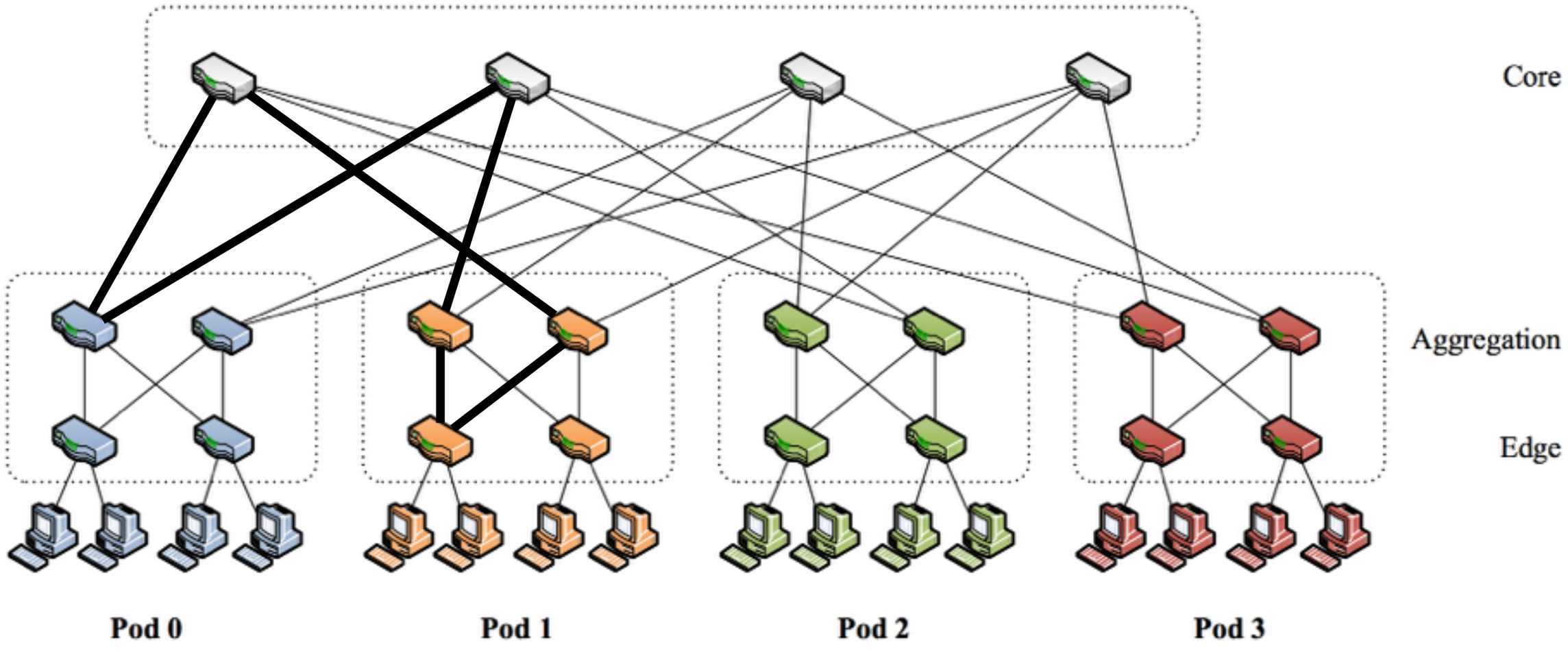
So, what do we do with this?

Proxy ARP



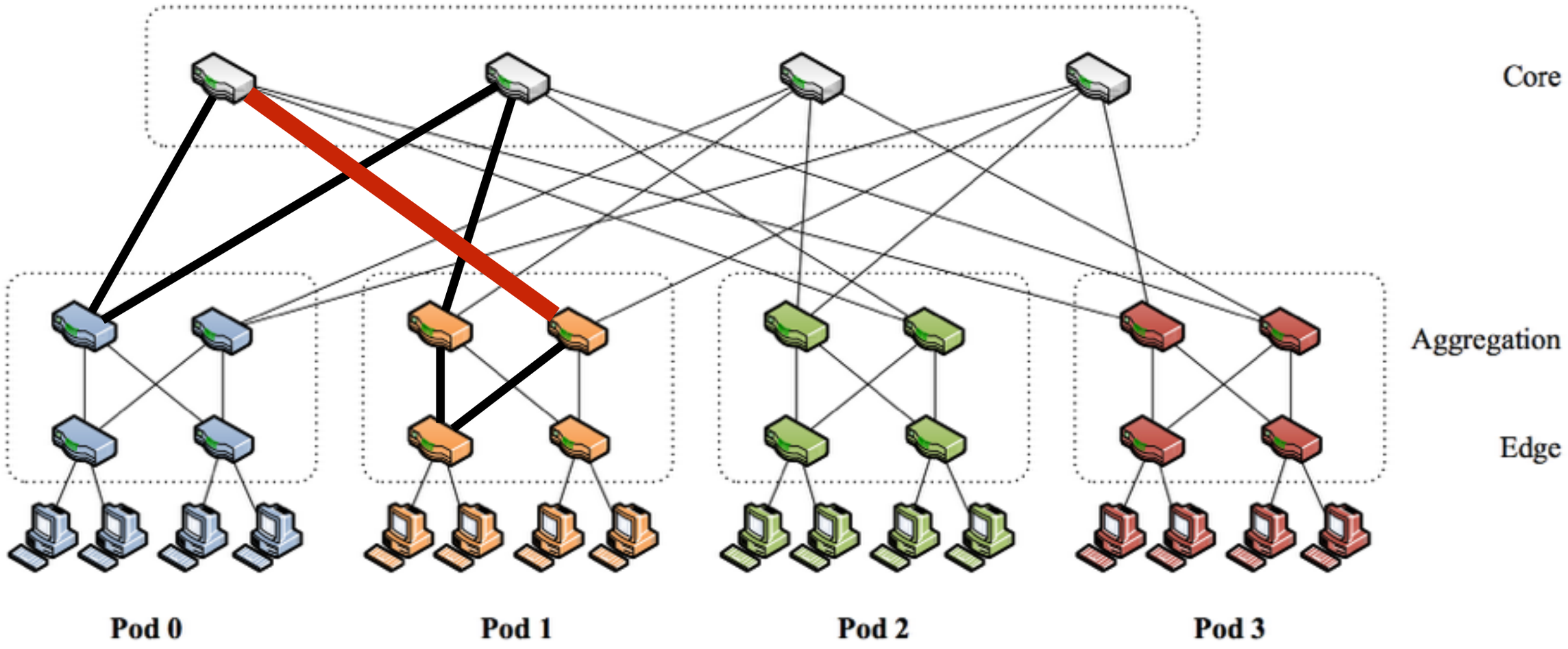
- Natively, resolve addresses by broadcasting
- However, if Fabric Manager knows an IP<->MAC mapping, we can eliminate broadcast traffic

Loop-free Forwarding



Fact: Fat trees have many physical loops!
Can you find them all?

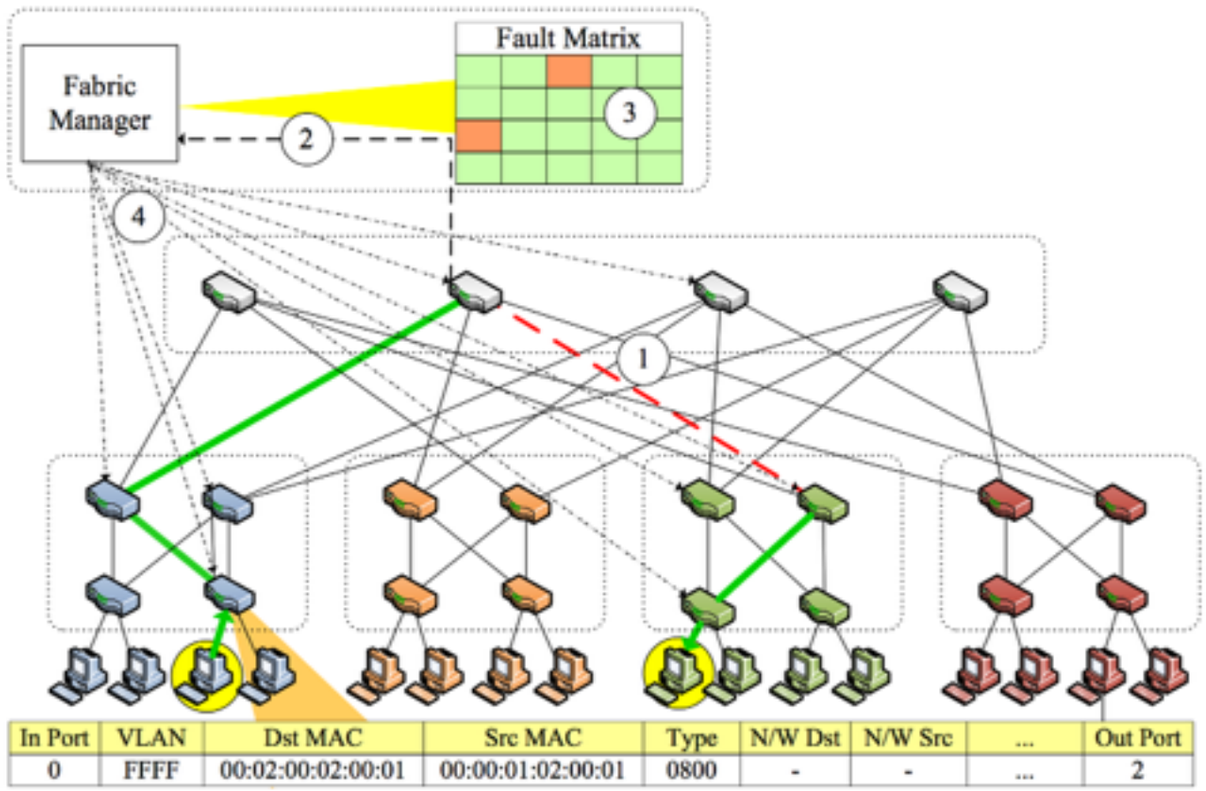
Provably Loop-free Forwarding



If we allow a packet to only go up the tree once, we cannot have a loop

As an aside: Fat Tree ~= folded Clos. Haven't we known this since the 50's?

Failover

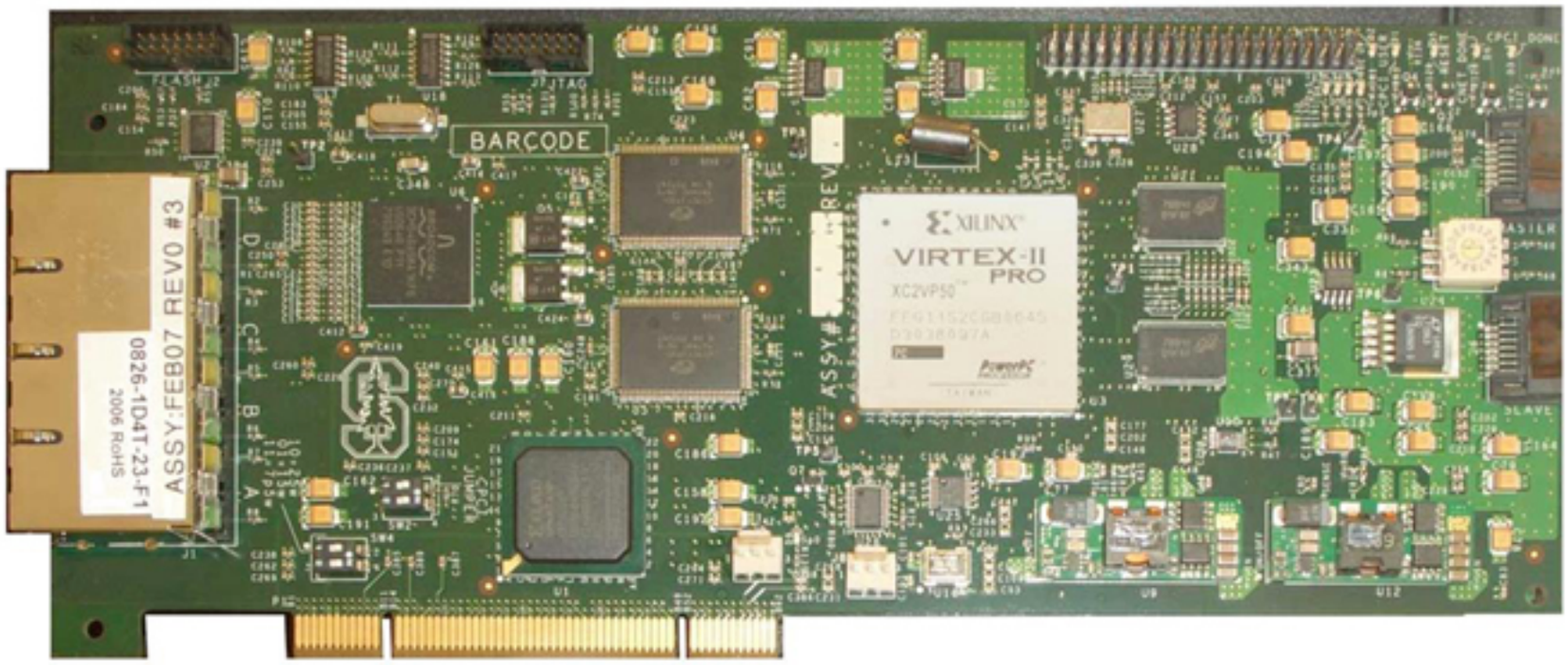


In Port	VLAN	Dst MAC	Src MAC	Type	N/W Dst	N/W Src	...	Out Port
0	FFFF	00:02:00:02:00:01	00:00:01:02:00:01	0800	-	-	...	2

- Fabric Manager maintains state of failed links
- Link failure is tracked using Location Discovery protocol —> missing message = failure

Validation

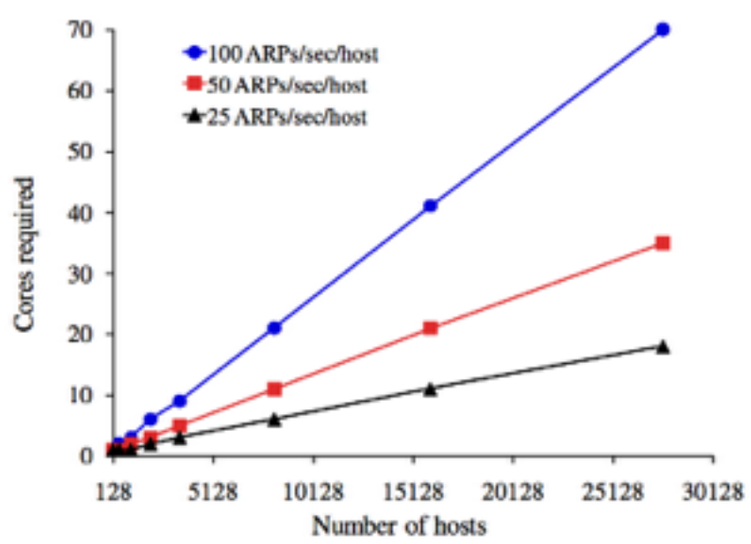
Experimental Testbed

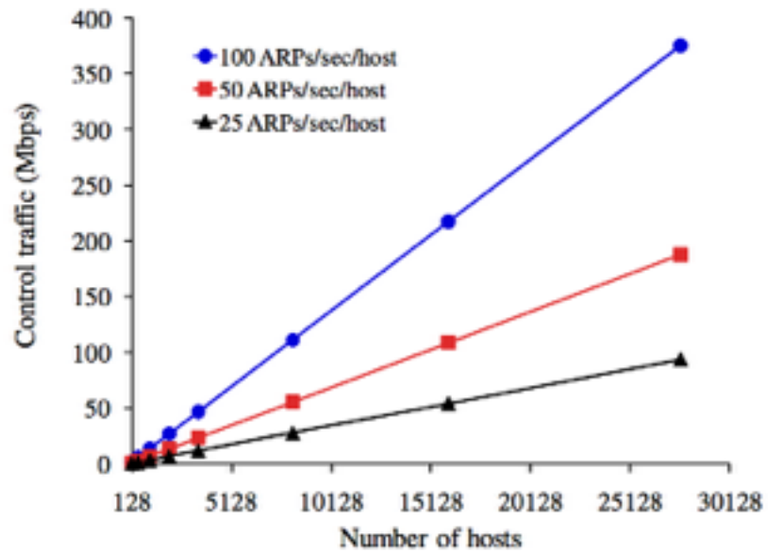


- 20x 4-port NetFPGAs arranged in a Fat Tree, 1 Gbps per port
- 16 compute nodes <— datacenter scale?

http://netfpga.org/1G_specs.html

Fabric Manager Capacity





- Authors cite 25 ARPs/sec/host as a high number
- 30,128 hosts <<< 100k machines, 32 VM/machine?
- Also, *deus ex machina* number: 25µs/request?

Conclusions

PortLand:
Did it achieve its goals?

R1: VM Migration	Migrate via ARP + unified L2
R2: No admin	Distributed process for learning network topology coupled with central Fabric Manager
R3: Any-to-any	Rewrite MAC addresses to reduce routing table size, makes problem tractable
R4: No loops	Provably loop free routing on Fat Trees
R5: Failure Recov.	Distributed topology learning algorithm rapidly learns failure status

Discussion

- If you have a practically static topology, you can eliminate most of the complexity in this paper…?

- Is the Fabric Manager actually viable as datacenters continue to increase in size?

- Wouldn't this be simpler if you pushed MAC address rewriting down to your VMM/hypervisor?