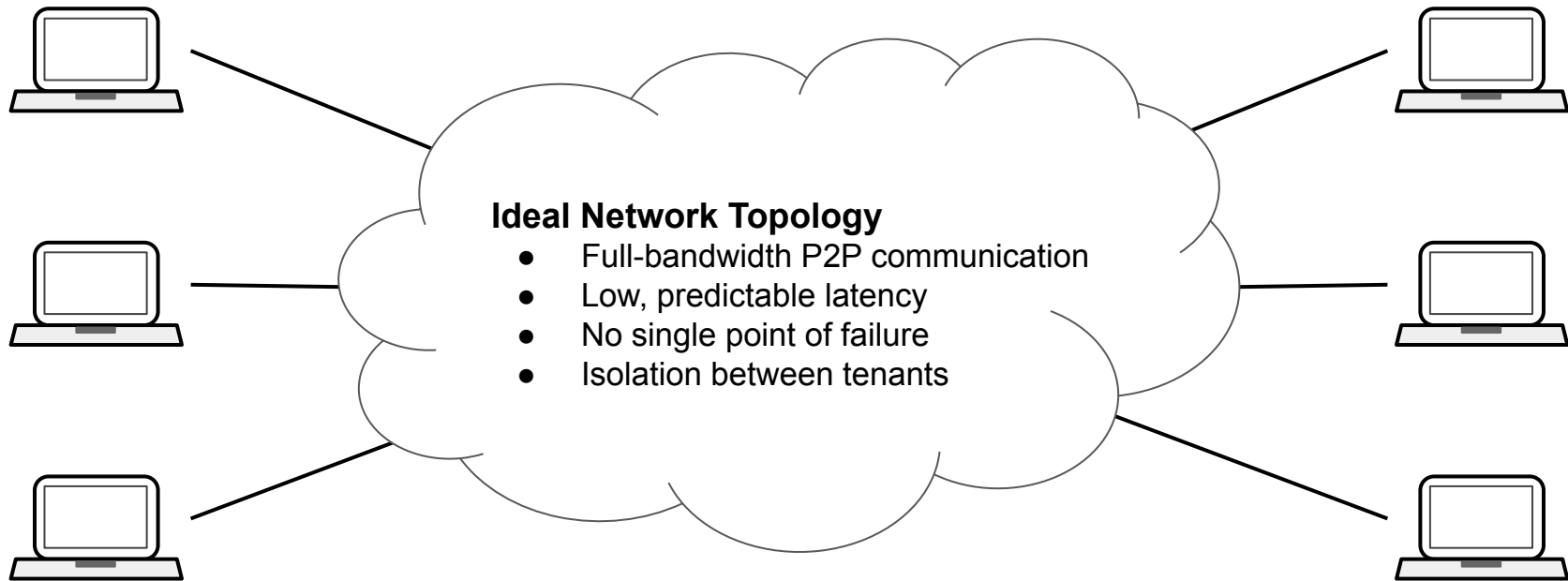


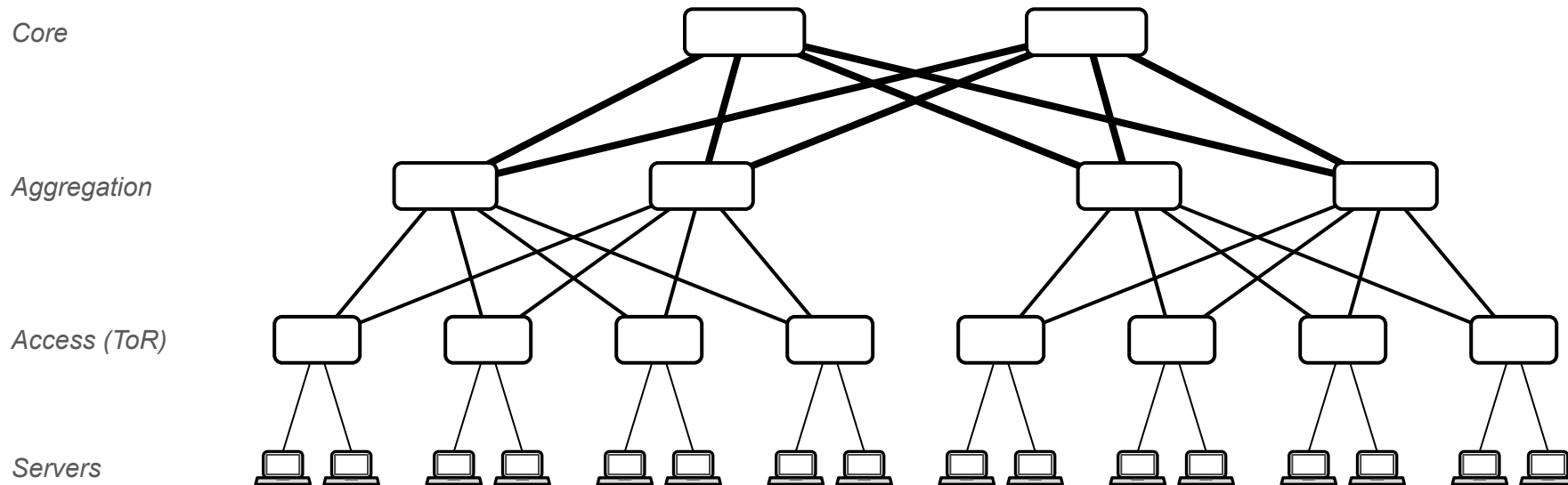
Datacenter Topology and Network Virtualization

Hongtao Zhang, Yuyao Wang

Topology Expectation

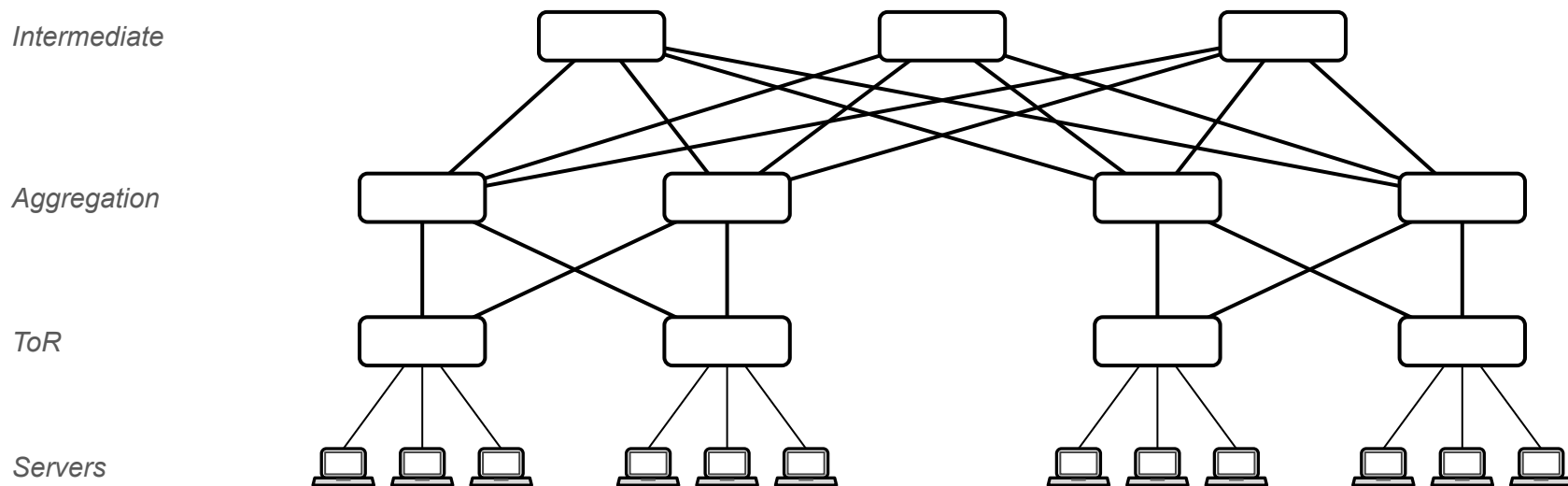


Traditional Tree Topology



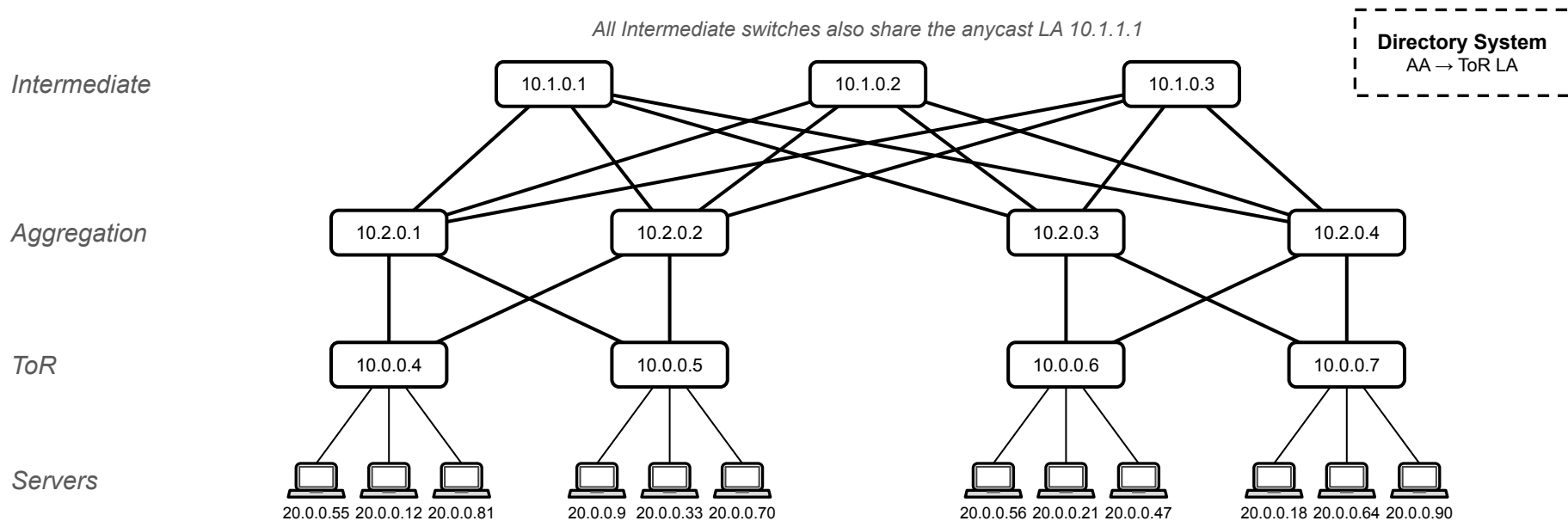
- Scale-up design: larger, more expensive switches as you move toward the core
- Heavy oversubscription at upper layers limits server-to-server bandwidth
- Few redundant paths: the core is both a bottleneck and a large failure domain

VL2



- Clos network: every Aggregation switch connects to every Intermediate switch (complete bipartite)
- Multiple upward/downward routes

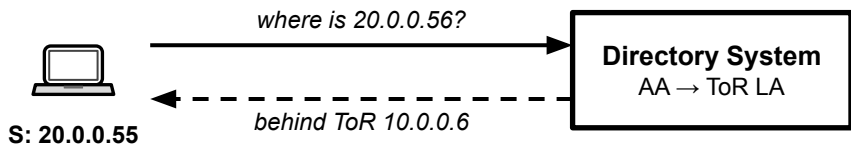
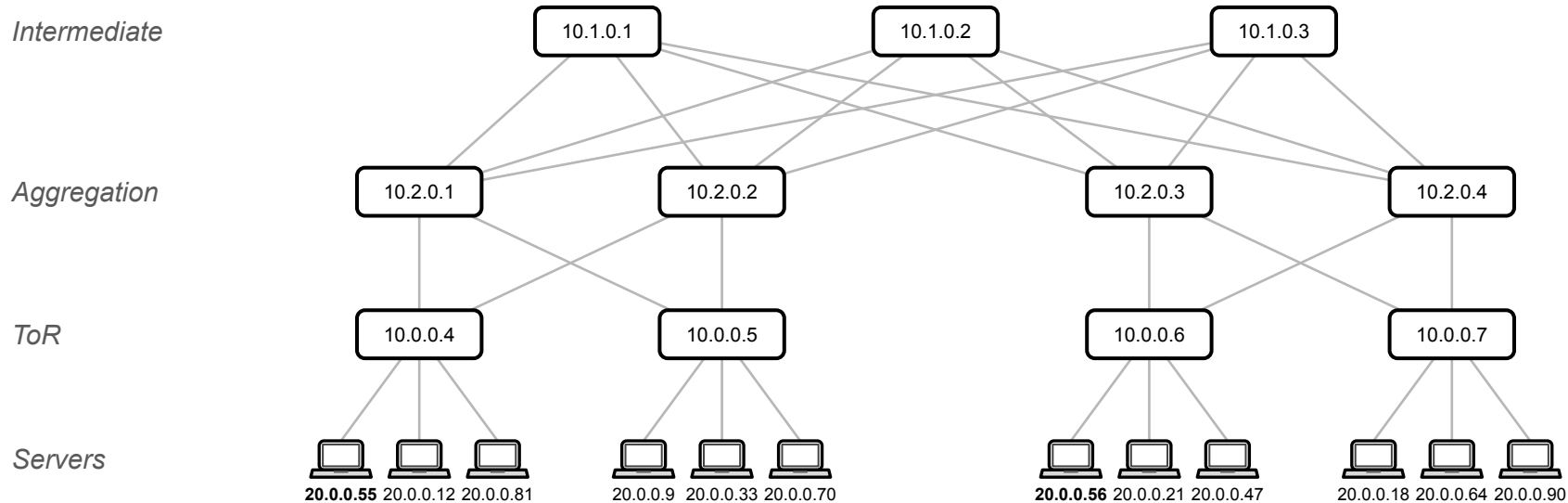
VL2 Addressing: AA vs. LA



- **LA (Location-specific Address):** on every switch; switches run link-state routing over LAs only
- **AA (Application-specific Address):** on servers; flat — 20.0.0.55 and 20.0.0.56 sit under different ToRs
- **Directory System** keeps AA → ToR-LA mappings; a VL2 agent on each server queries it

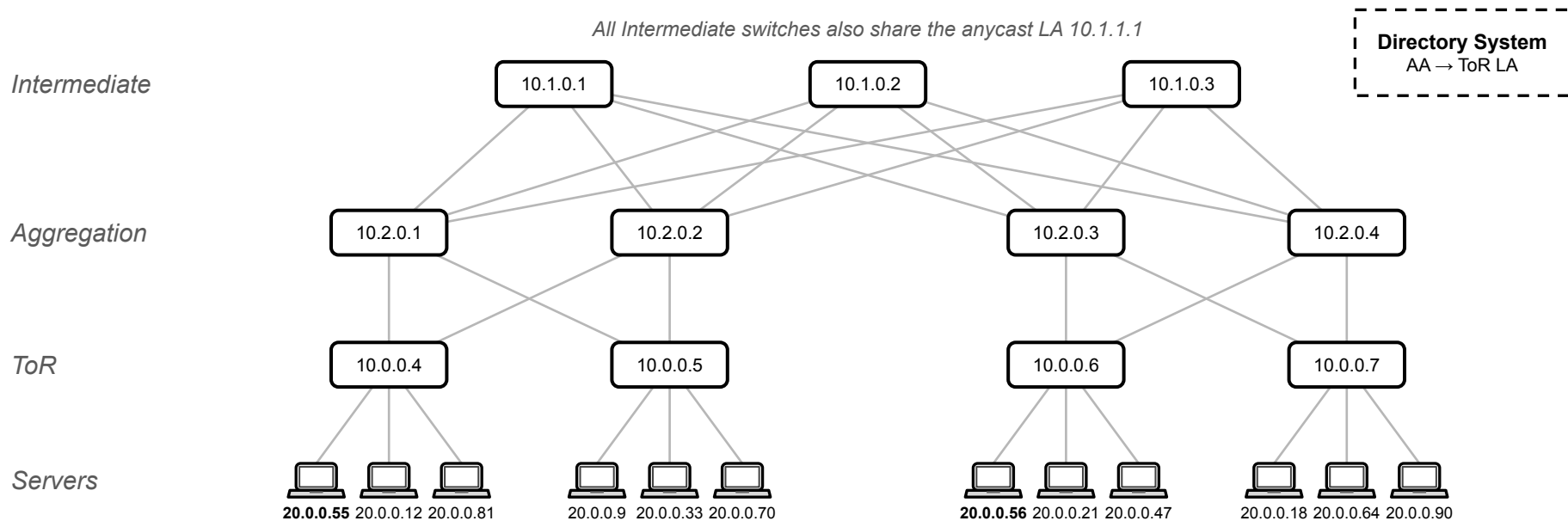
VL2 Routing (1/4): Resolve the destination

All Intermediate switches also share the anycast LA 10.1.1.1



Step 1/4 · S wants to reach AA 20.0.0.56. Its VL2 asks the Directory System, which maps the AA to the LA of the ToR it sits behind.

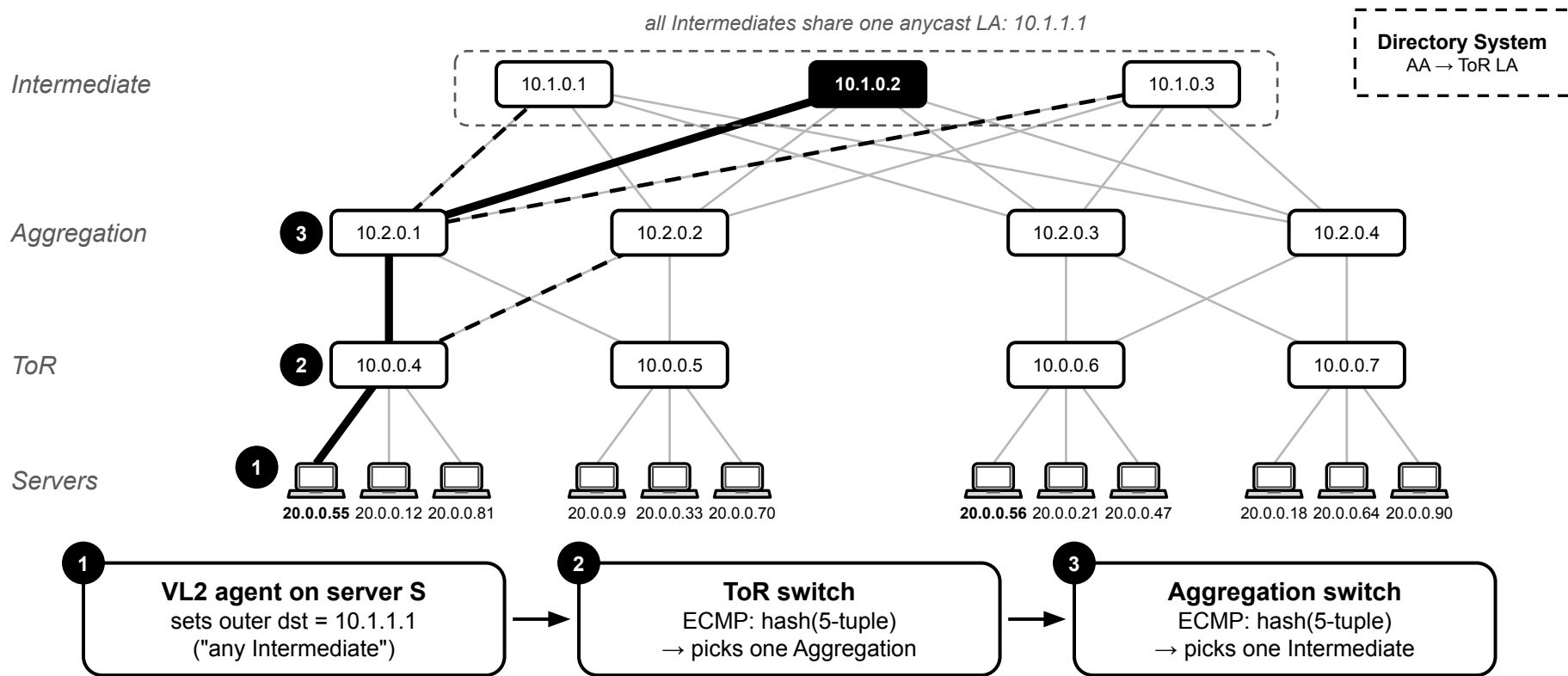
VL2 Routing (2/4): Double encapsulation



Step 2/4 · The agent wraps the AA packet twice: inner header → destination ToR's LA, outer header → anycast LA of the Intermediates.

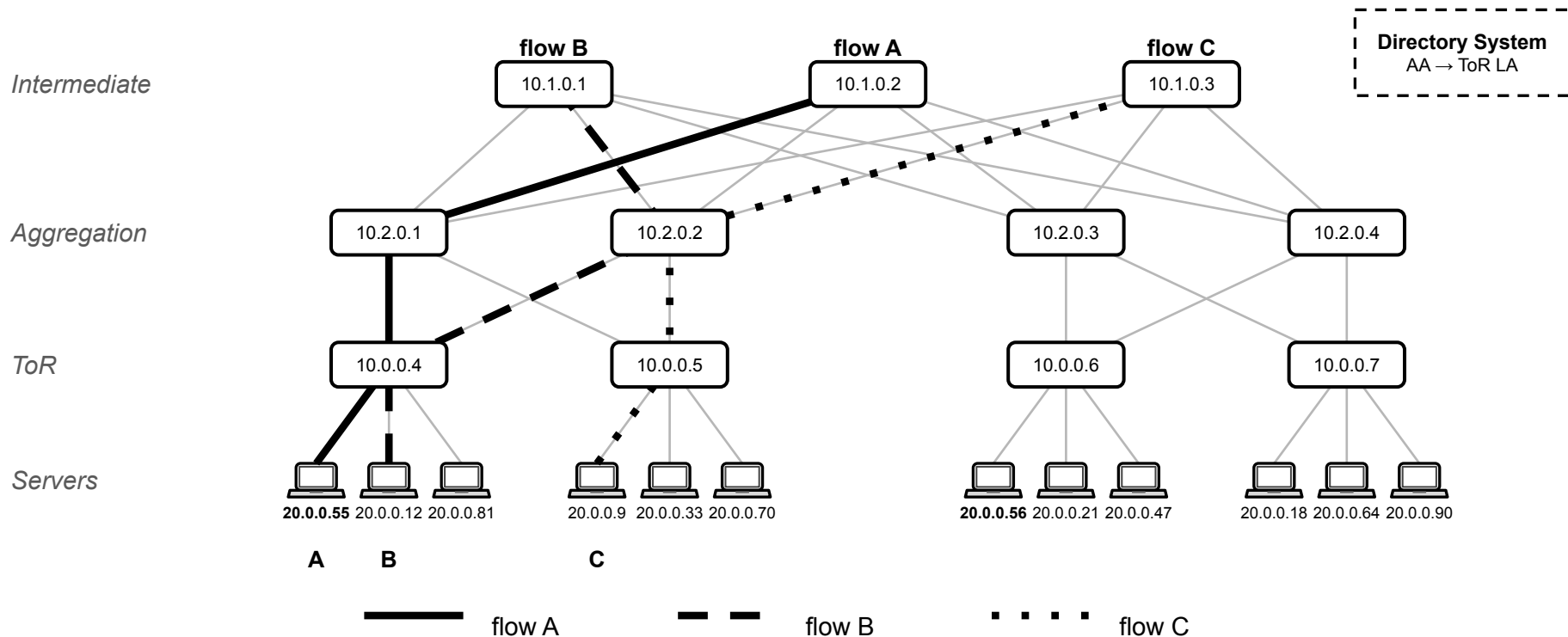
Outer: dst 10.1.1.1 <i>Intermediate anycast LA</i>	Inner: dst 10.0.0.6 <i>destination ToR LA</i>	20.0.0.55 → 20.0.0.56 <i>original AA packet</i>	Payload
--	---	---	----------------

VL2 Routing (3/4): Who picks the Intermediate?



The server only says "go to any Intermediate"; switch ECMP hashing makes the random choice, per flow. No central controller.

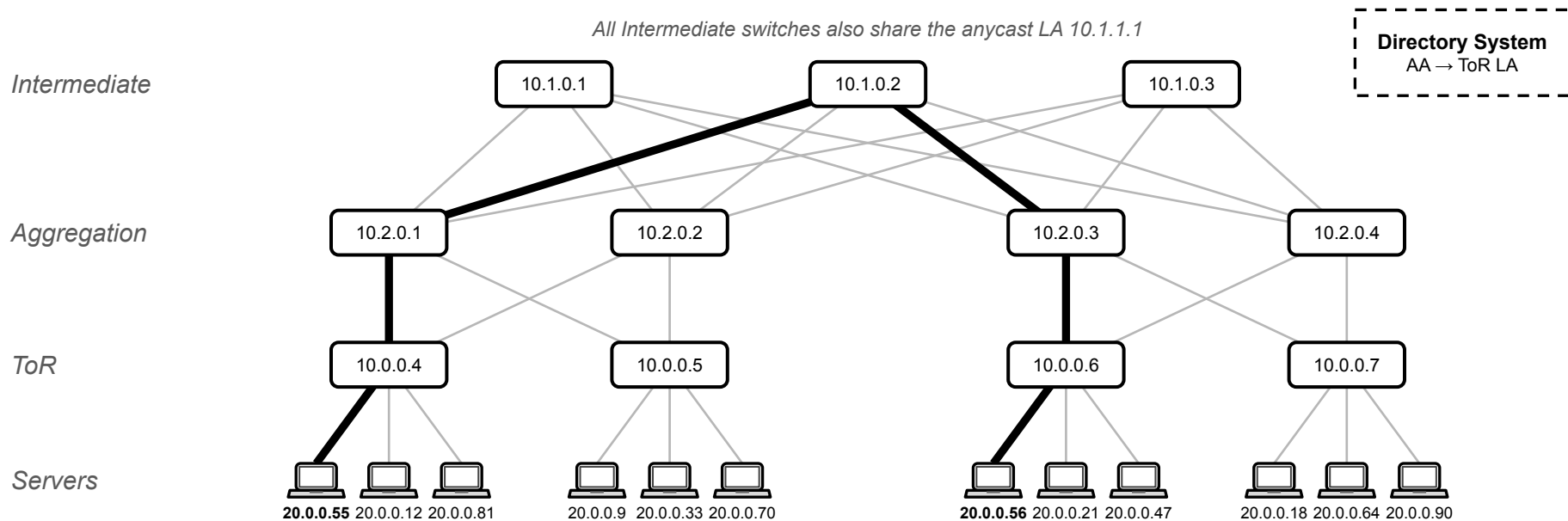
VL2 Routing (3/4): Many flows, all Intermediates



Each flow is hashed independently → load spreads evenly over all Intermediates, whatever the traffic matrix.

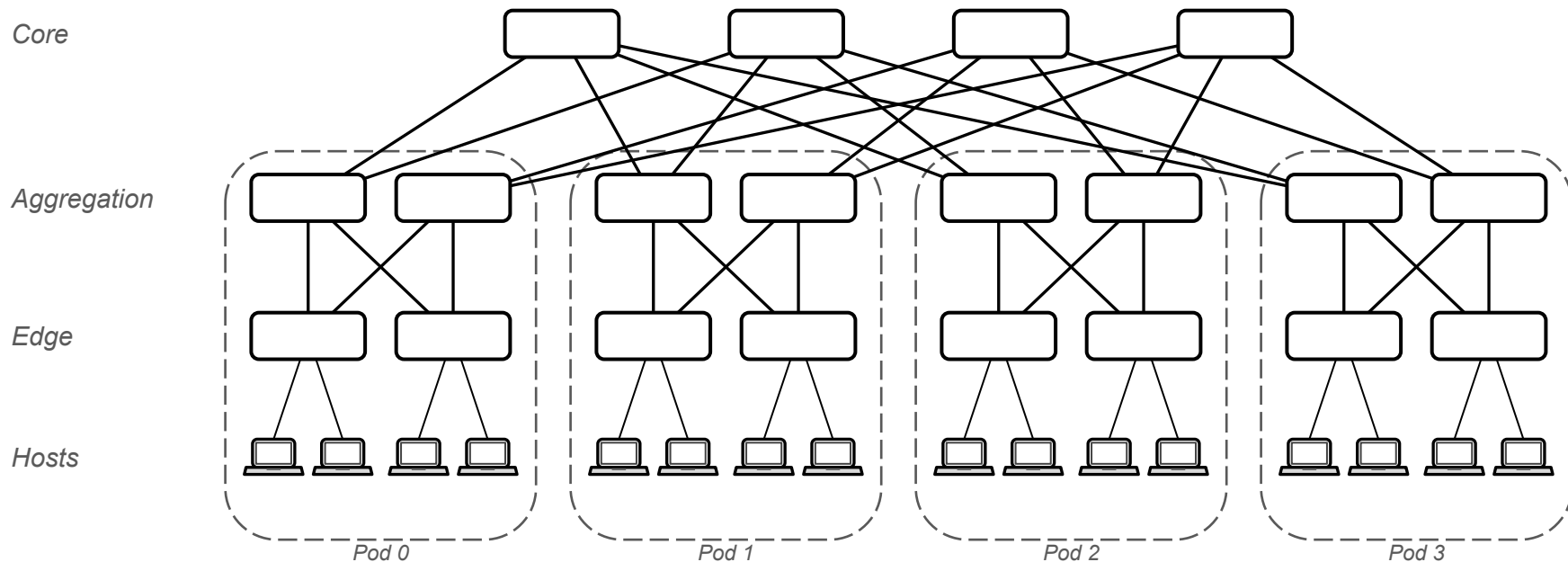
Cost: one extra hop up to the core; balancing is per flow, so two elephant flows can still collide.

VL2 Routing (4/4): Decapsulate and deliver



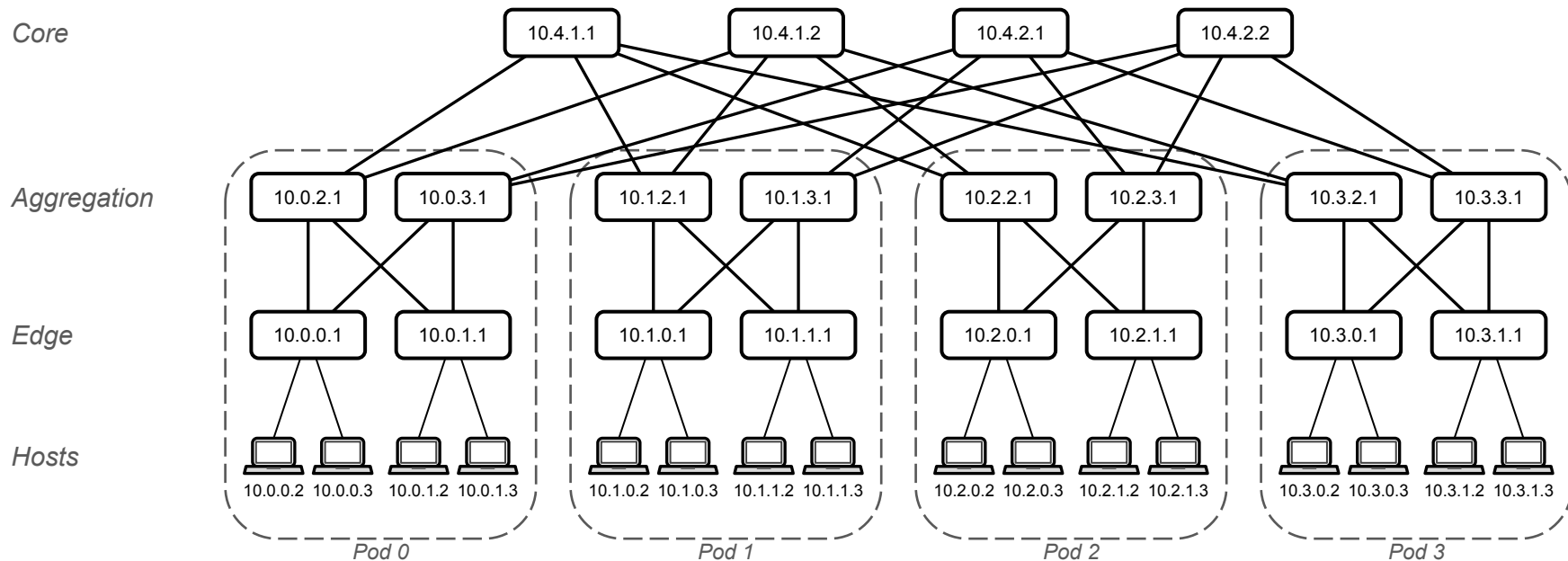
Step 4/4 • Intermediate 10.1.0.2 strips the outer header and routes to ToR 10.0.0.6; the ToR strips the inner header and delivers to AA 20.0.0.56. Different flows take different Intermediates.

Fat-Tree (k = 4)



- Built entirely from identical k -port commodity switches (here $k = 4$)
- k pods, $(k/2)^2$ core switches, supports $k^3/4$ hosts (16 here)
- Multiple paths from hosts to any core switches (upwards), one path from a core switch to a host (downwards)

Fat-Tree Addressing (10.0.0.0/8)



- **Pod switch:** 10.pod.switch.1 (switch 0–1 = edge, 2–3 = aggregation)
- **Core switch:** 10.k.j.i ($j, i \in [1, k/2]$; here 10.4.j.i)
- **Host:** 10.pod.switch.ID ($ID \in [2, k/2+1]$, i.e. .2 and .3)

Fat-Tree Two-Level Routing Tables

Edge 10.2.0.1

local hosts, else go up

Prefix / Suffix	Port
10.2.0.2/32	0
10.2.0.3/32	1
0.0.0.0/0 → suffix	
0.0.0.2/8	2
0.0.0.3/8	3

Aggregation 10.2.2.1

own pod: down by prefix, else up

Prefix / Suffix	Port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0 → suffix	
0.0.0.2/8	2
0.0.0.3/8	3

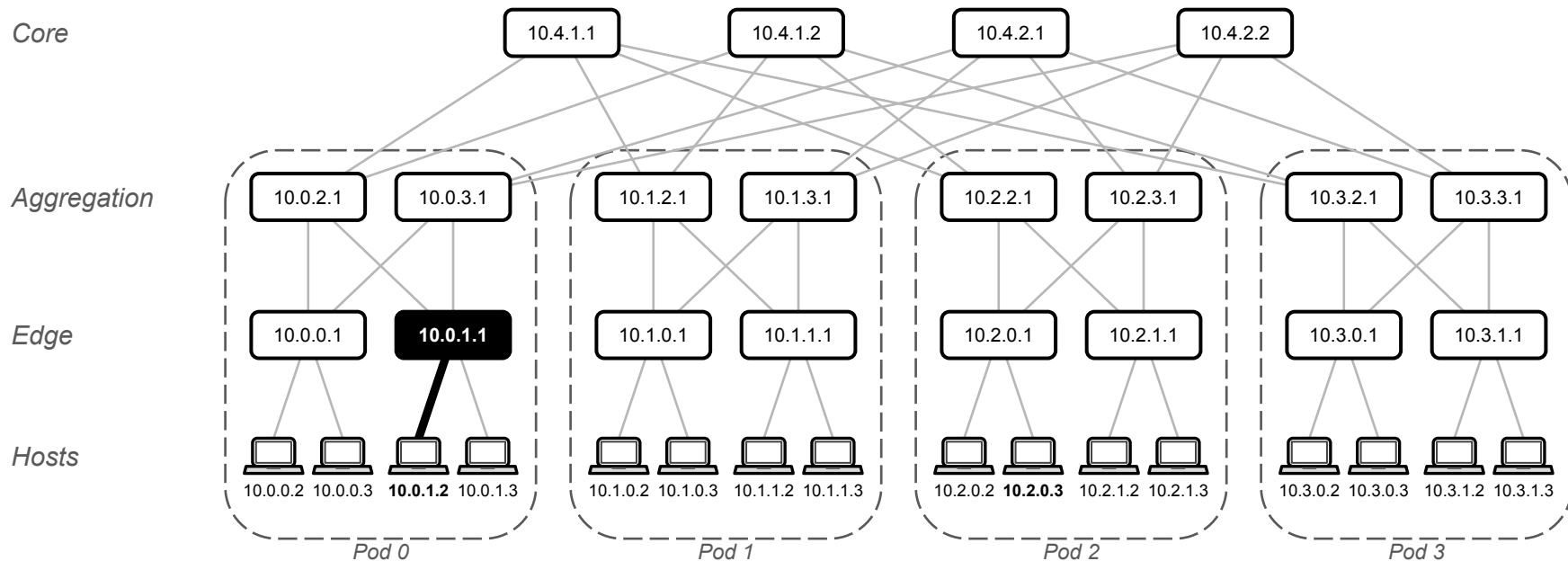
Core 10.4.1.1

one /16 per pod

Prefix / Suffix	Port
10.0.0.0/16	0
10.1.0.0/16	1
10.2.0.0/16	2
10.3.0.0/16	3

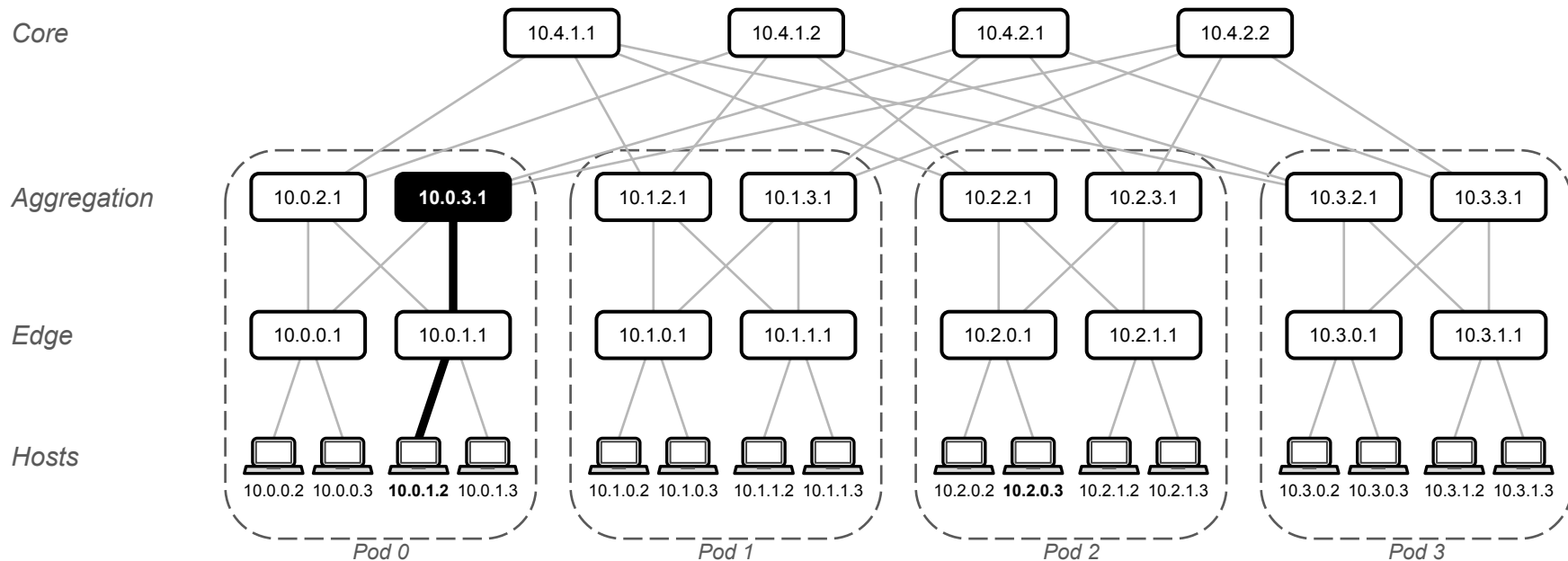
- **Prefix (left-handed) match** routes down toward the destination pod / subnet
- **Suffix (right-handed) match** on the host ID picks the uplink, spreading traffic over different cores
- **Determinism:** the same destination always takes the same path, so packets are not reordered

Fat-Tree Routing (1/5): 10.0.1.2 → 10.2.0.3



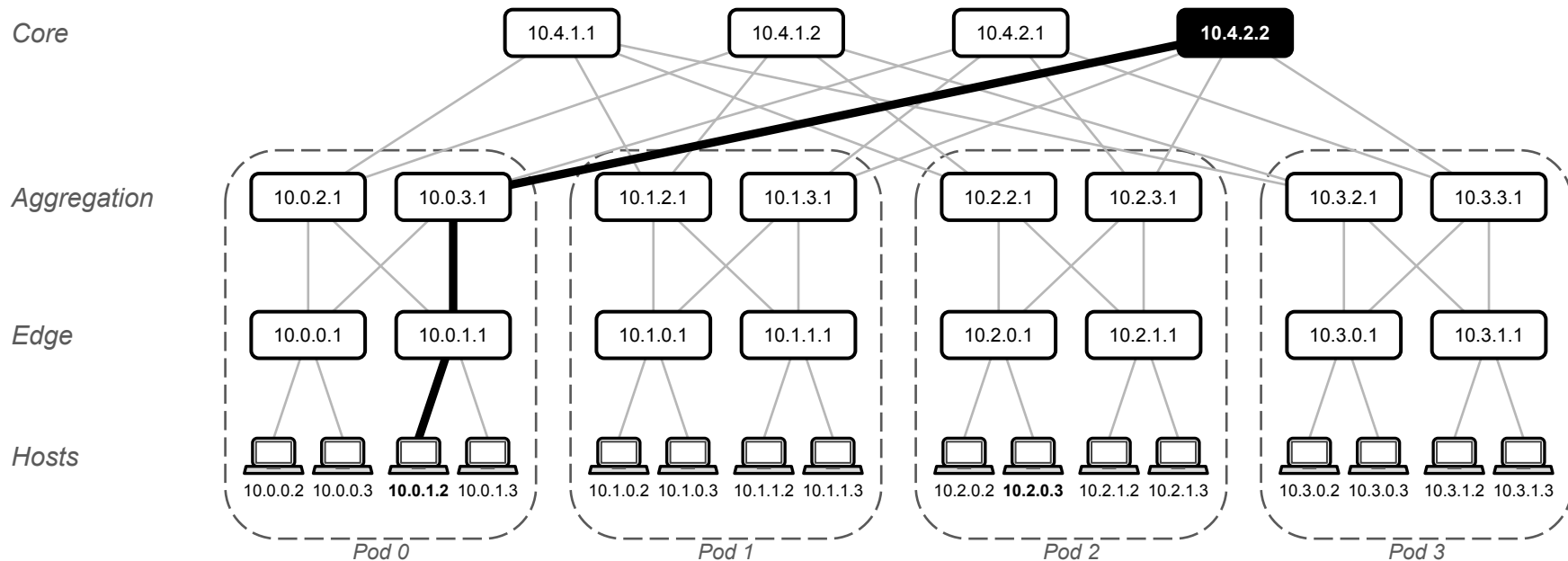
Step 1/5 · Edge 10.0.1.1: 10.2.0.3 is not a local host → 0.0.0.0/0 → suffix table: host ID .3 matches 0.0.0.3/8 → port 3
→ Aggregation 10.0.3.1

Fat-Tree Routing (2/5): 10.0.1.2 → 10.2.0.3



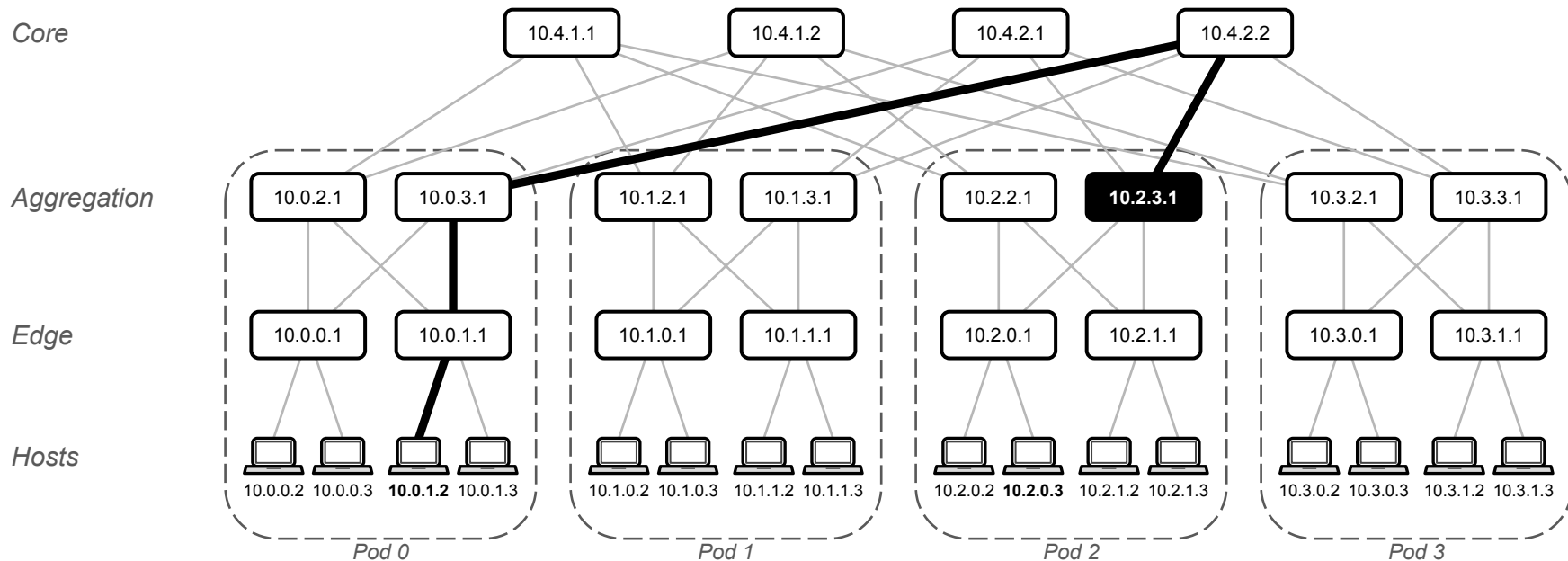
Step 2/5 · Aggregation 10.0.3.1: no pod-0 prefix matches 10.2.0.3 → suffix .3 matches 0.0.0.3/8 → port 3 → Core 10.4.2.2

Fat-Tree Routing (3/5): 10.0.1.2 → 10.2.0.3



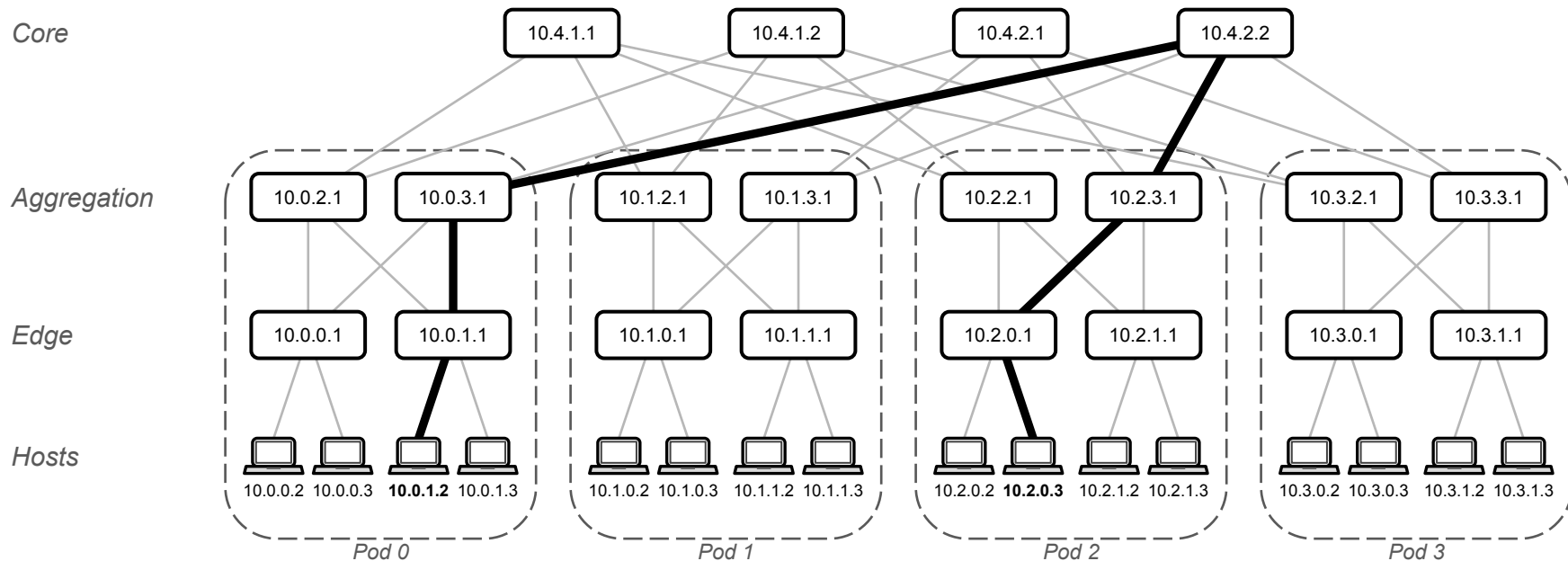
Step 3/5 · Core 10.4.2.2: prefix 10.2.0.0/16 → port 2 → Pod 2, Aggregation 10.2.3.1

Fat-Tree Routing (4/5): 10.0.1.2 → 10.2.0.3



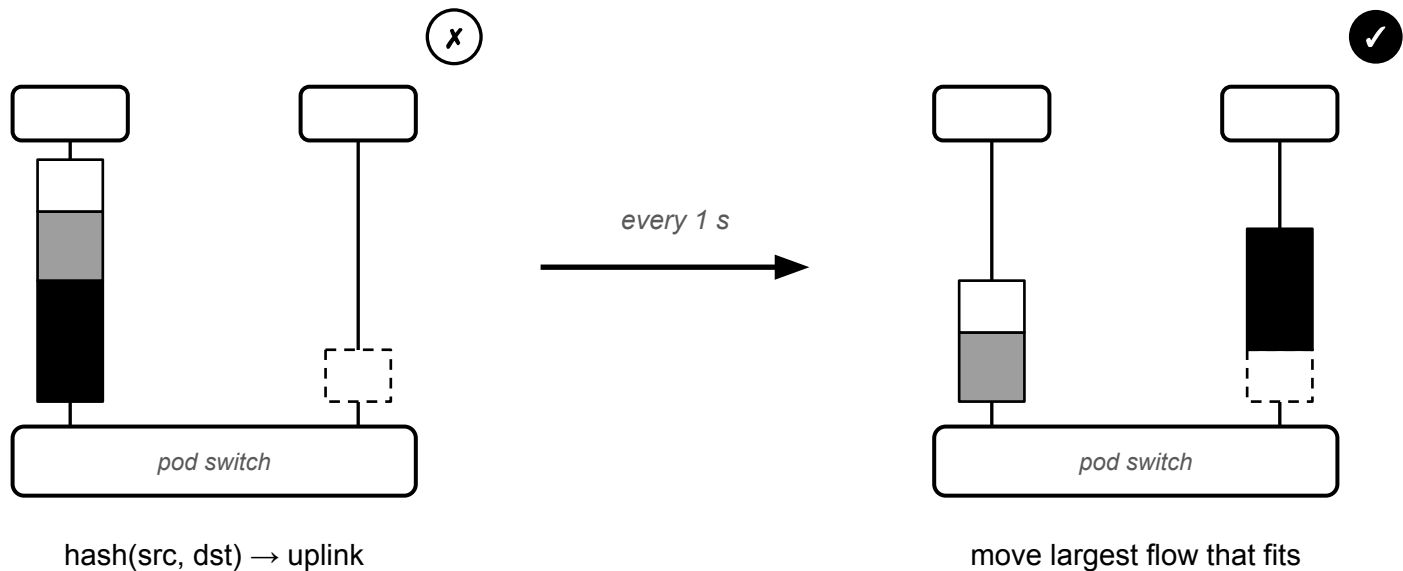
Step 4/5 · Aggregation 10.2.3.1: prefix 10.2.0.0/24 → port 0 → Edge 10.2.0.1

Fat-Tree Routing (5/5): 10.0.1.2 → 10.2.0.3



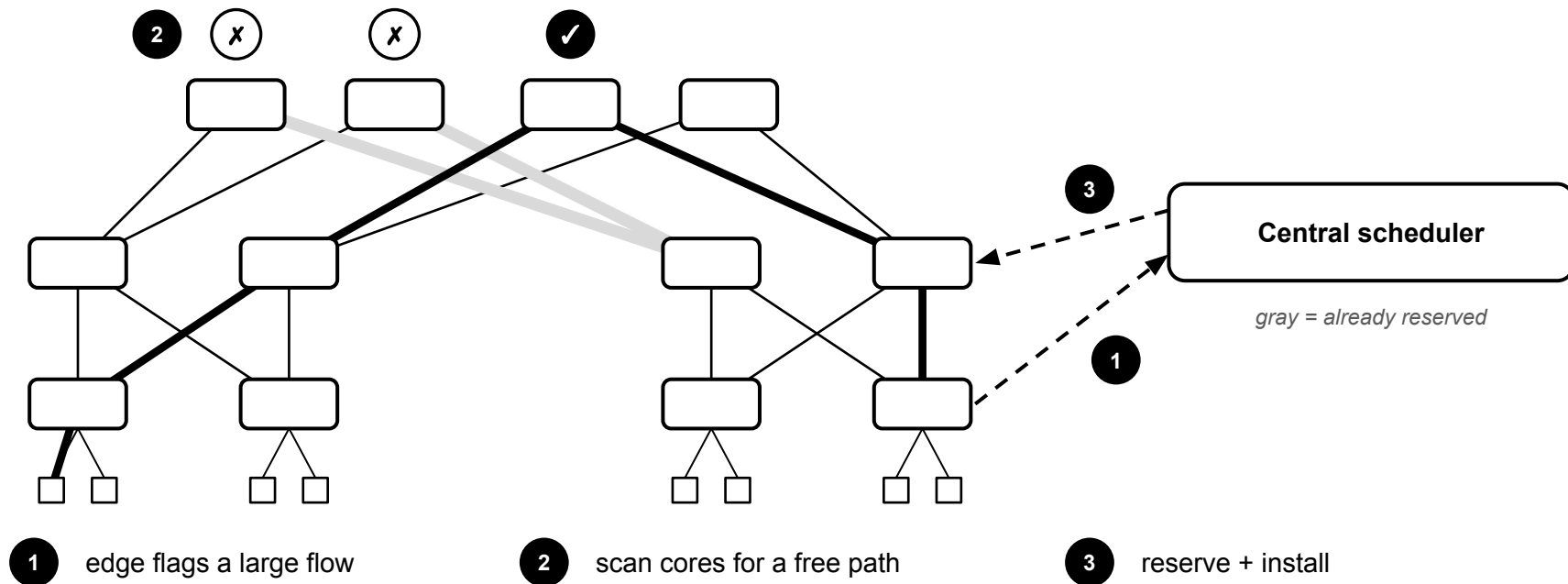
Step 5/5 · Edge 10.2.0.1: 10.2.0.3/32 → port 1 → host 10.2.0.3. Going up was chosen by the destination's suffix; going down was forced by prefixes.

Fat-Tree Dynamic Routing (1/3): Flow Classification

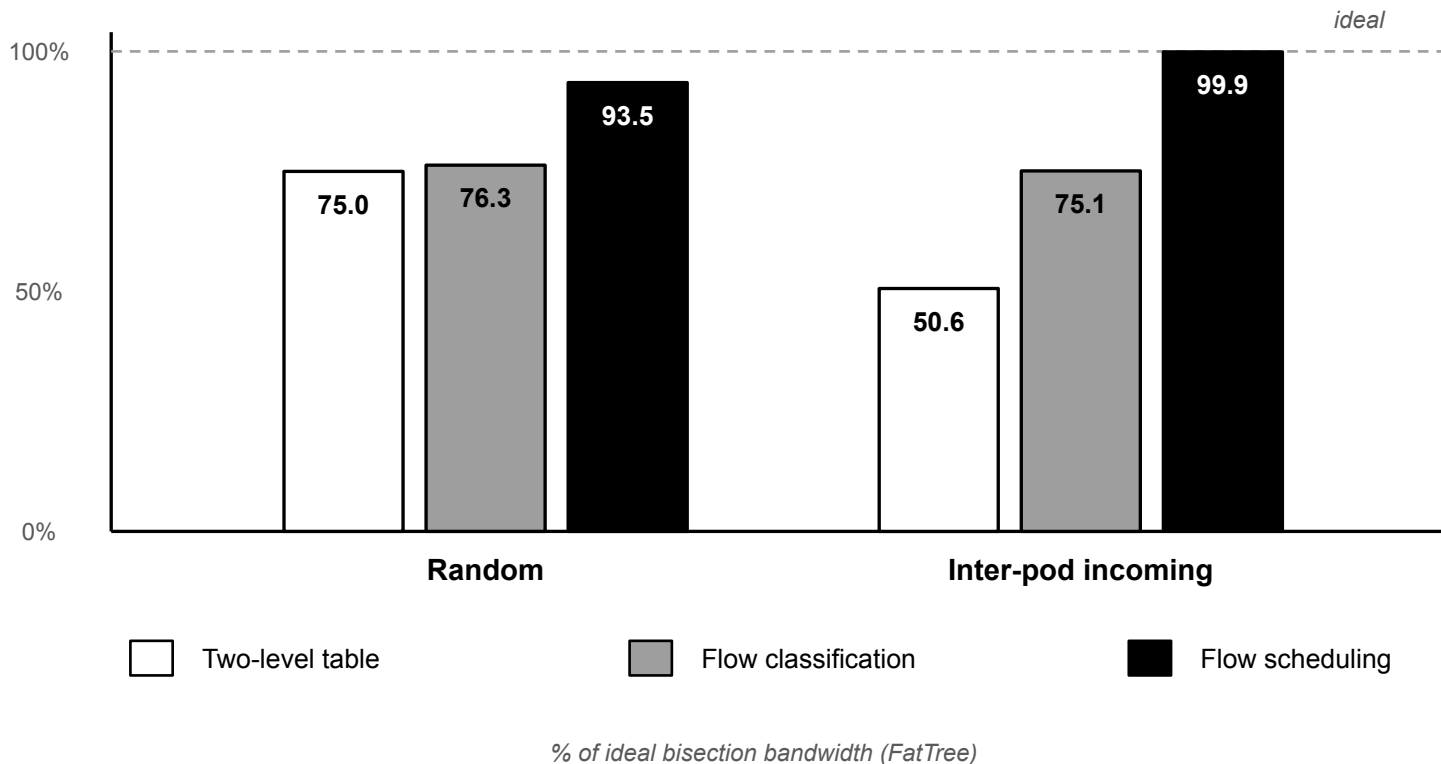


Local to each pod switch · keeps a flow on one port (no reordering) · ≤ 3 moves per round

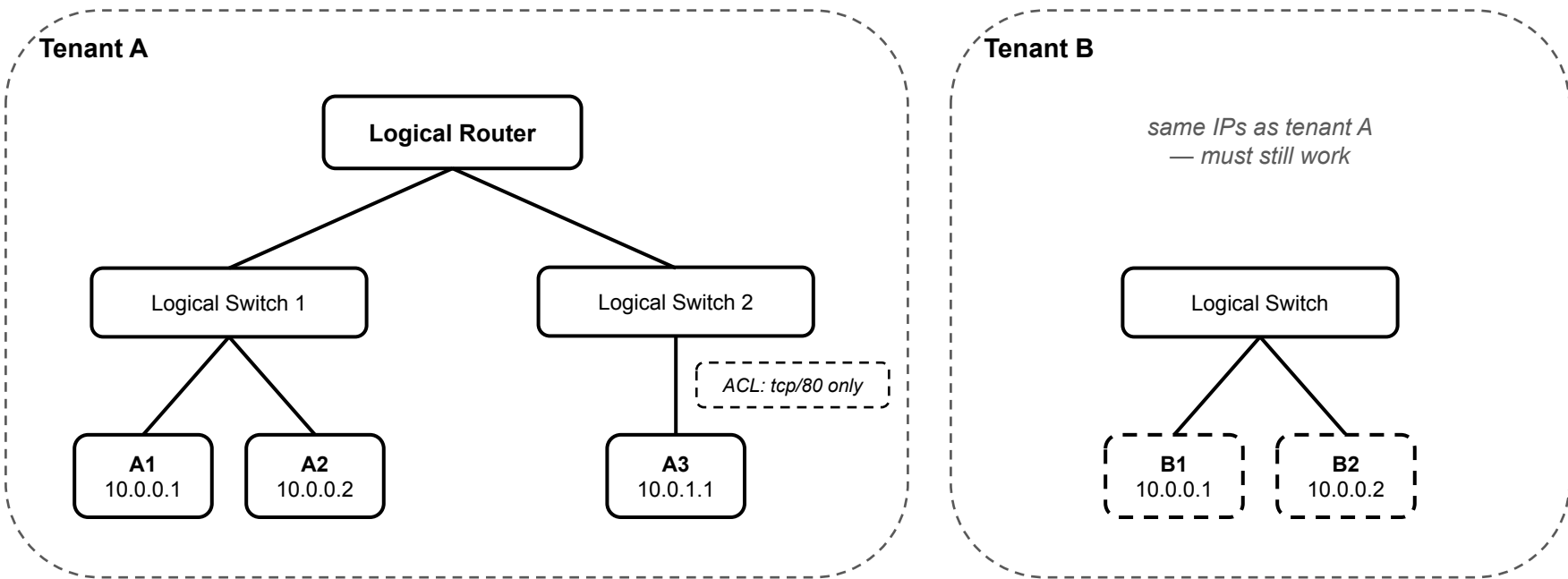
Fat-Tree Dynamic Routing (2/3): Flow Scheduling



Fat-Tree Dynamic Routing (3/3): How Much It Helps

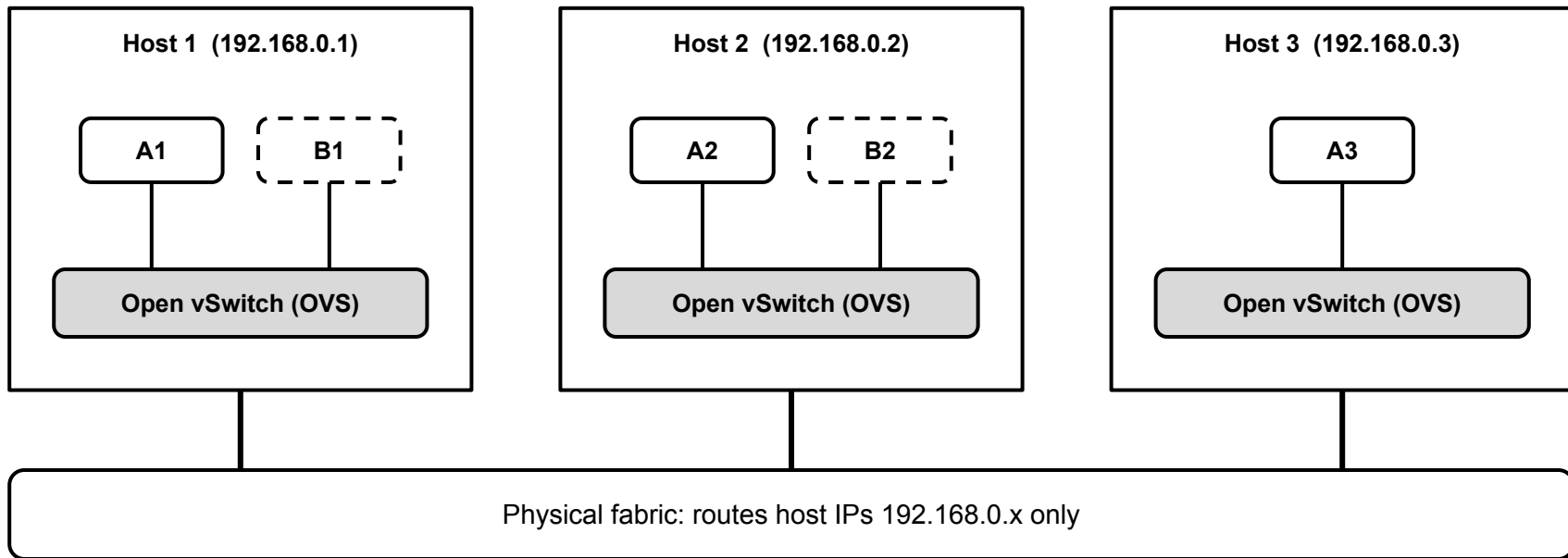


Network Virtualization: What Tenants Want



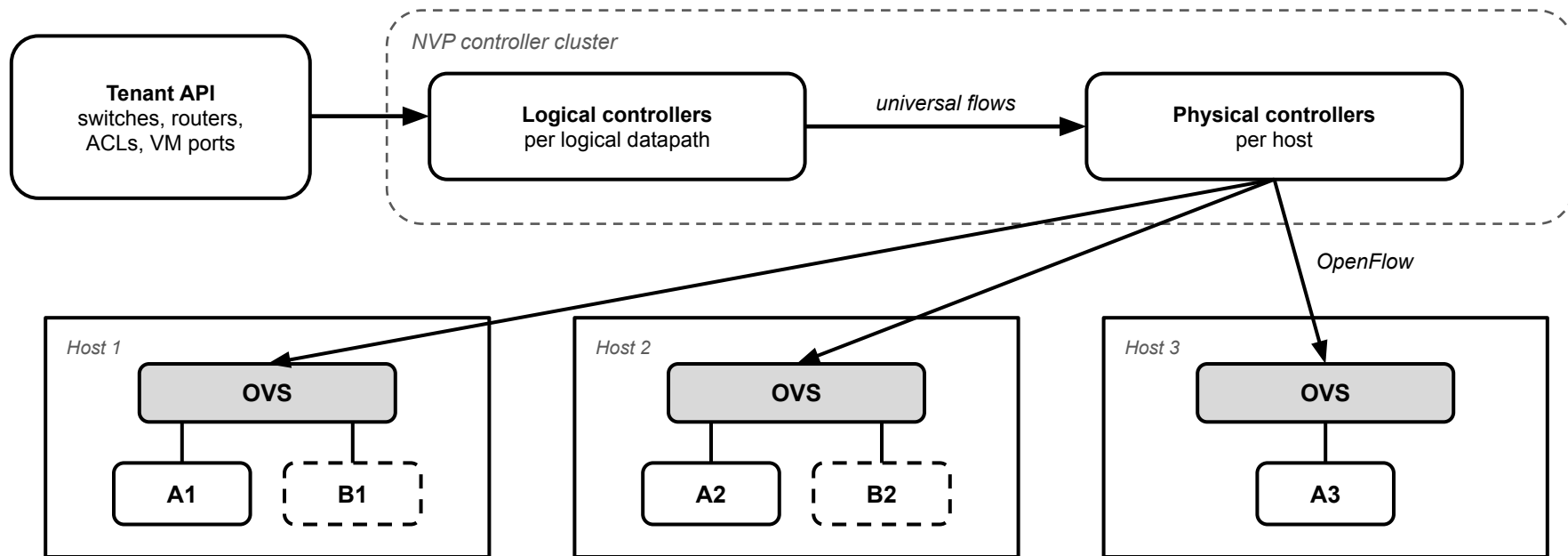
Each tenant: its own topology · its own addresses · its own policies

Reality: VMs Scattered Across Hosts



The fabric knows nothing about tenants → the logical networks must be built in the hosts' OVS.

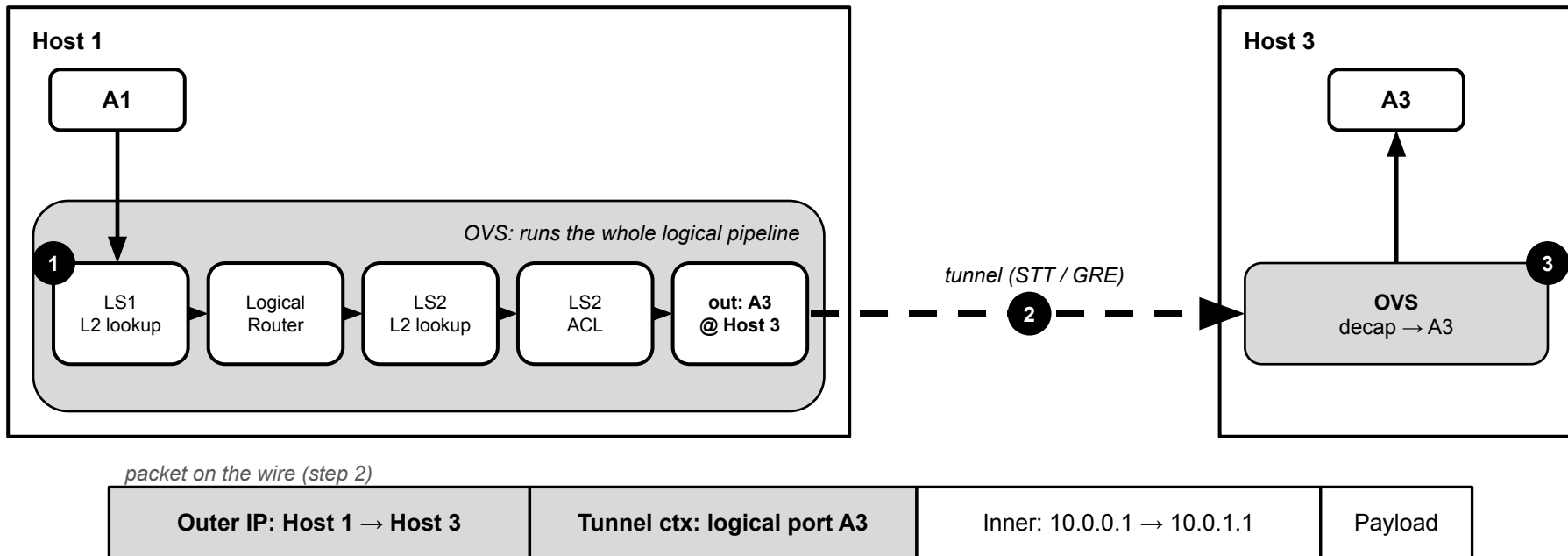
NVP: Compile Logical Networks to OVS



Tenant intent → universal flows → per-host OVS flows

Declarative (nlog) computation; recomputed incrementally when config or VM placement changes.

NVP Packet Walk: A1 → A3



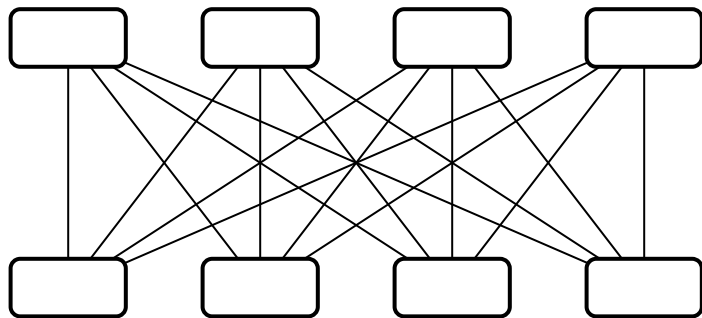
All logical hops run in the source OVS; the fabric just carries Host 1 → Host 3.

Broadcast/multicast go via service nodes; traffic to the physical world via gateways.

Discussions

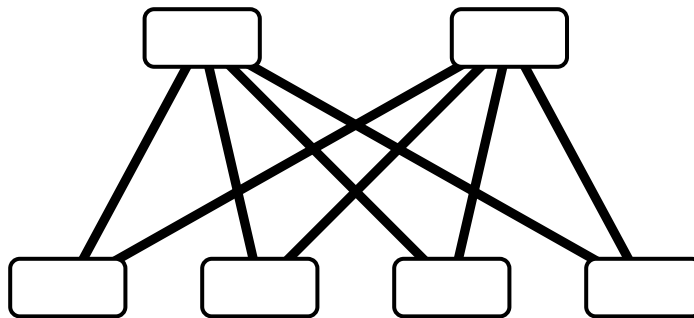
Topology :: Link Speeds

Fat-tree: 1G everywhere



many thin links → cable explosion

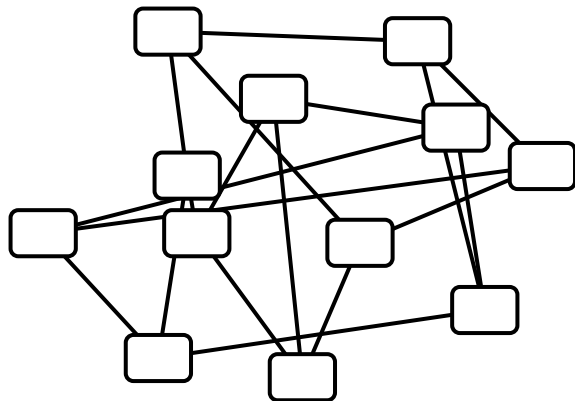
VL2: 10G up, 1G down



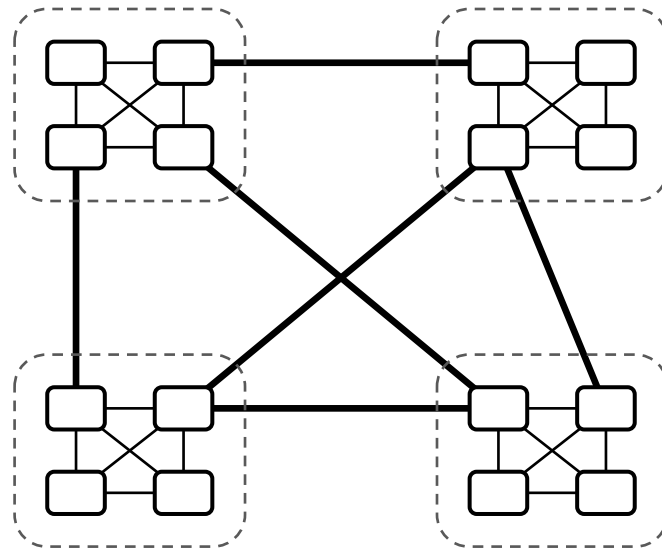
few fast links → fewer cables

Which 2008 constraints shaped each design, and which still hold?

Topology :: Alternate Structures (1/2)

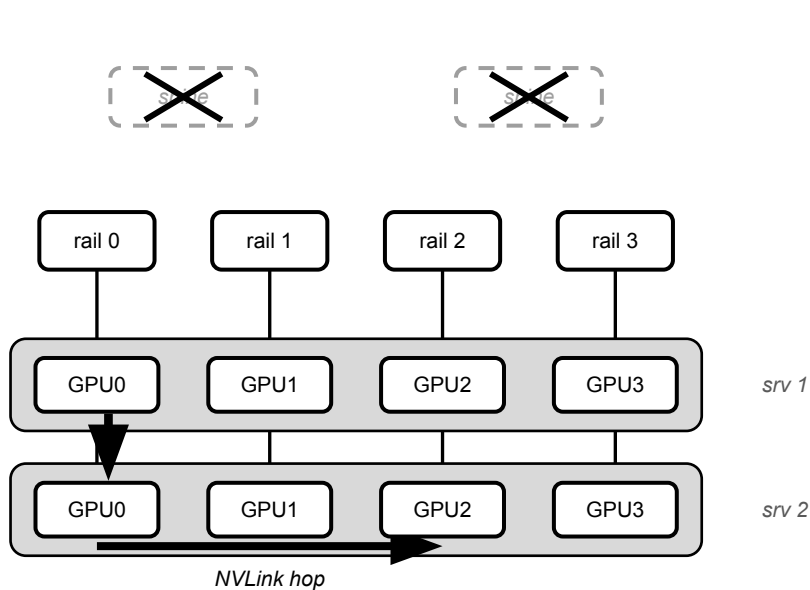


Random graph (Jellyfish, RNG)

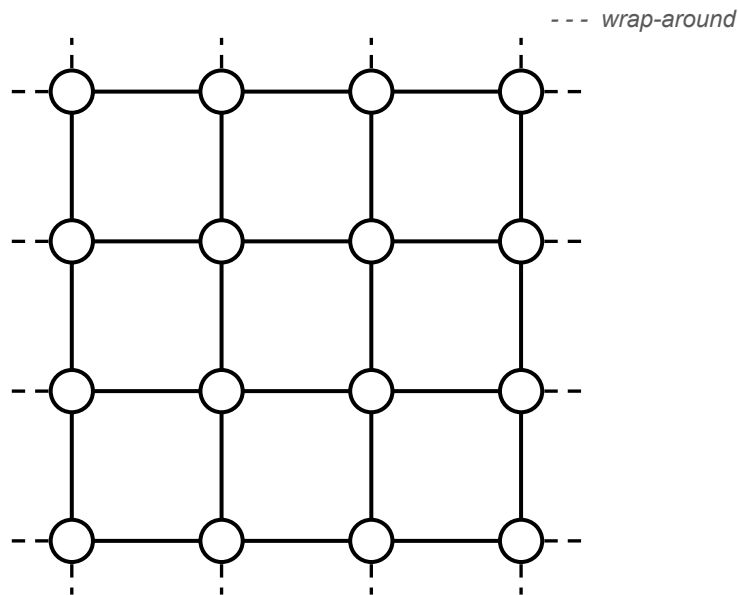


Dragonfly

Topology :: Alternate Structures (2/2)



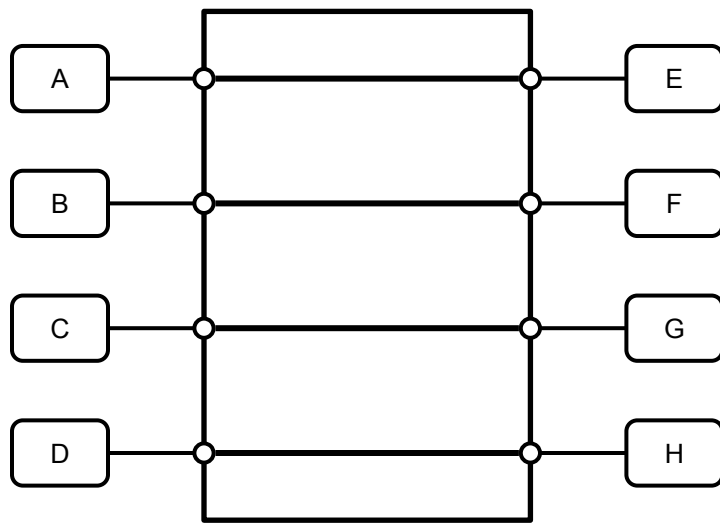
Rail-only



2D torus (TPU-style)

*When does matching the topology to the traffic beat a general-purpose topology?
Is there, and should there be a one-size-fits-all answer to DC topology?*

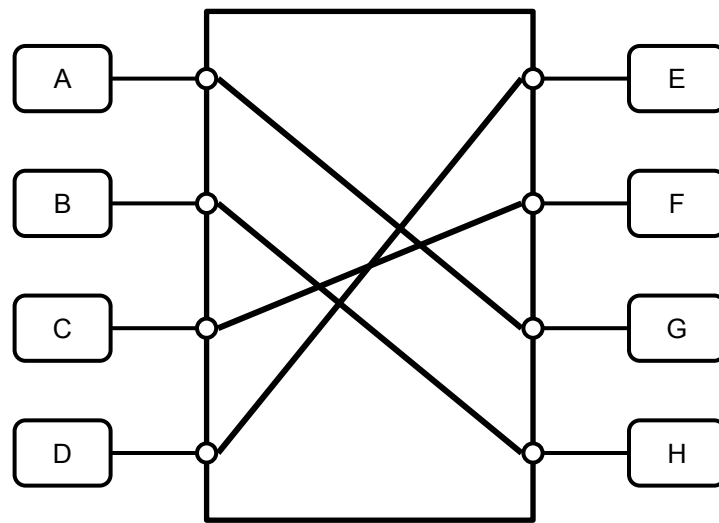
Topology :: Programmability



Optical circuit switch

Configuration at t_1

reconfigure



Optical circuit switch

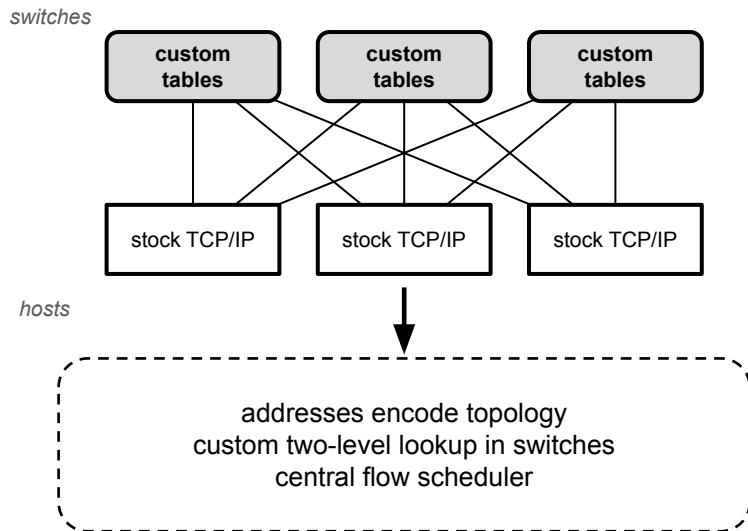
Configuration at t_2

If links can move, how often should they, and who decides?

Routing :: What Can Change?

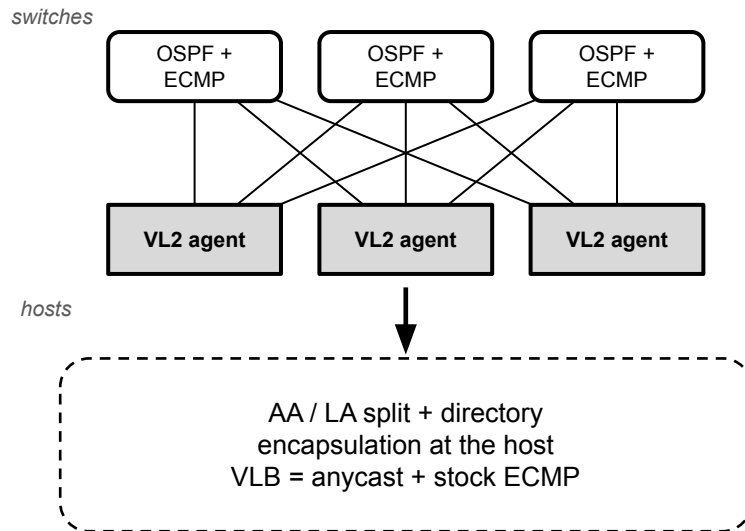
Fat-tree

hosts fixed, switches programmable



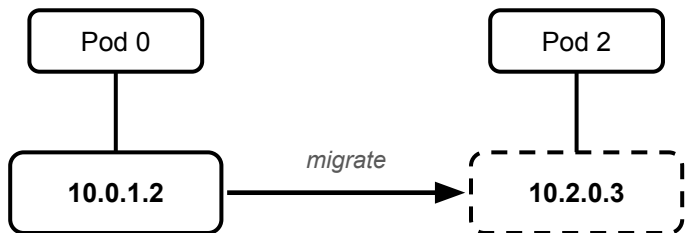
VL2

switches fixed, hosts programmable



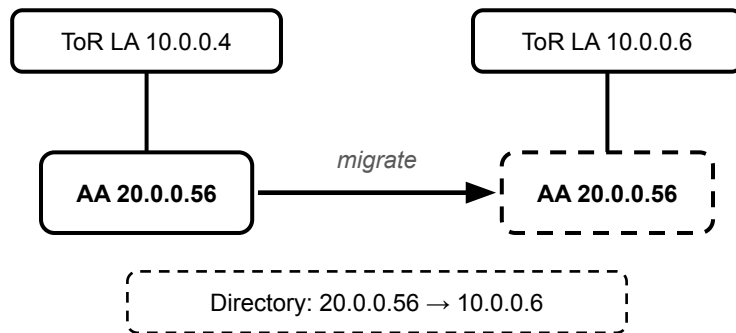
Routing :: Addressing

Fat-tree: address = location



VM moves → its IP must change

VL2: identity (AA) ≠ location (LA)

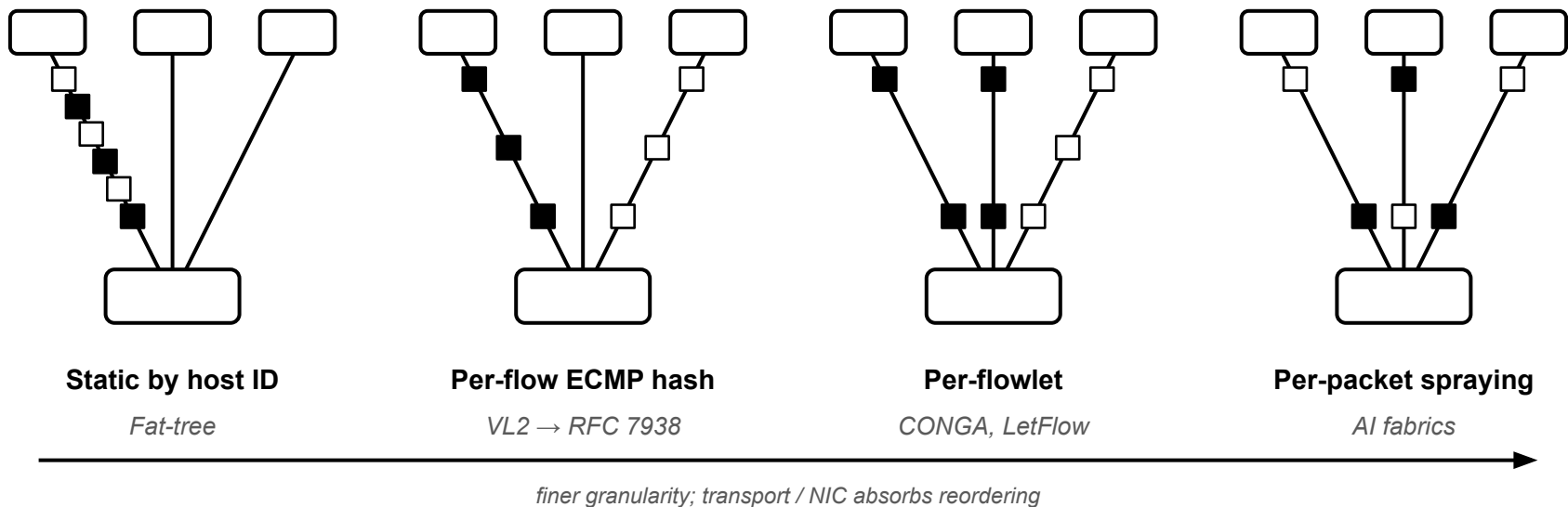


identity stays; only the mapping changes

Should flexibility be part of the topology concern, or be handled by virtualization?

Routing :: Load Balancing

■ flow 1 □ flow 2



What's the concern, pros/cons, host-side requirements for each strategy?