

Advanced Computer Networks

Addressing and Routing in Data Center Networks (II)

<https://pages.cs.wisc.edu/~mgliu/CS740/F26/index.html>

Ming Liu
mgliu@cs.wisc.edu

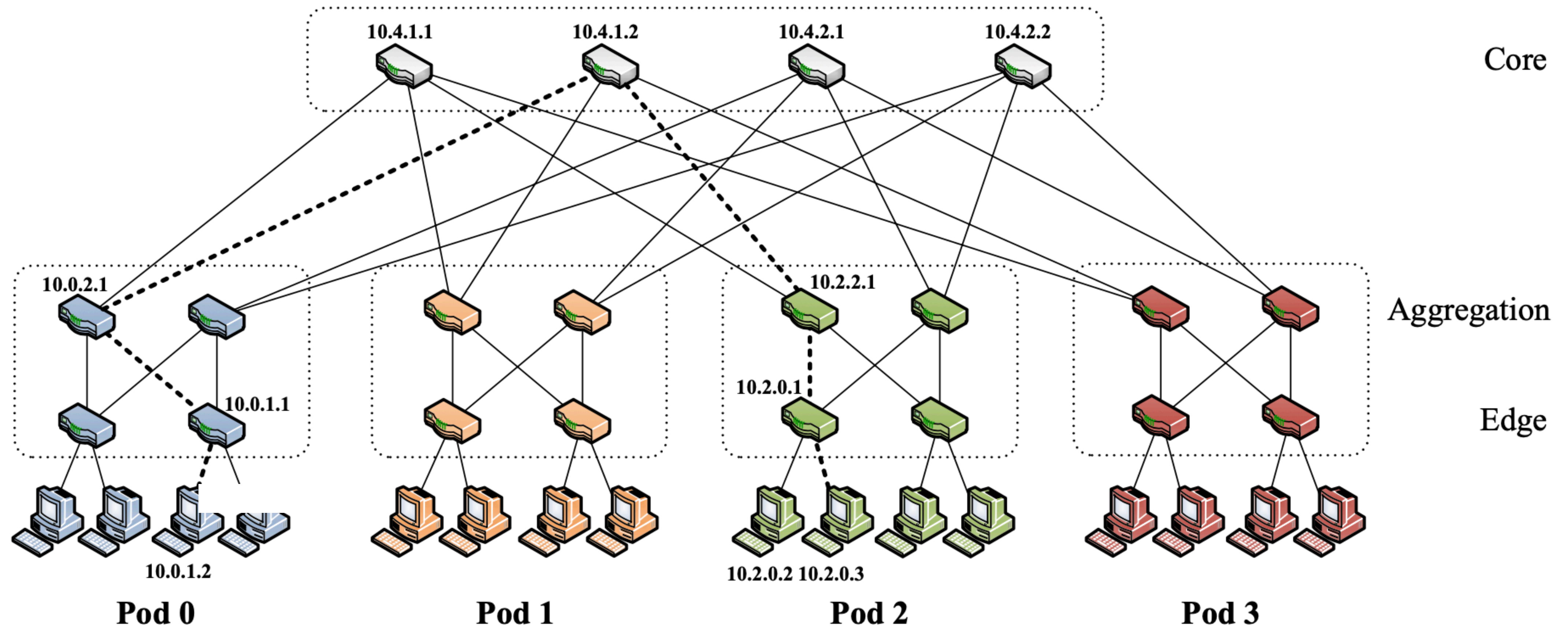
Outline

- Last lecture
 - Addressing and routing in data center networks (I)
- Today
 - Addressing and routing in data center networks (II)
- Announcements
 - Project proposal due 10/12/2026 11:59 PM
 - Lab1 due 10/09/2026 11:59 PM

An efficient addressing and routing mechanism should inherently encode the multiple-path routing capability!

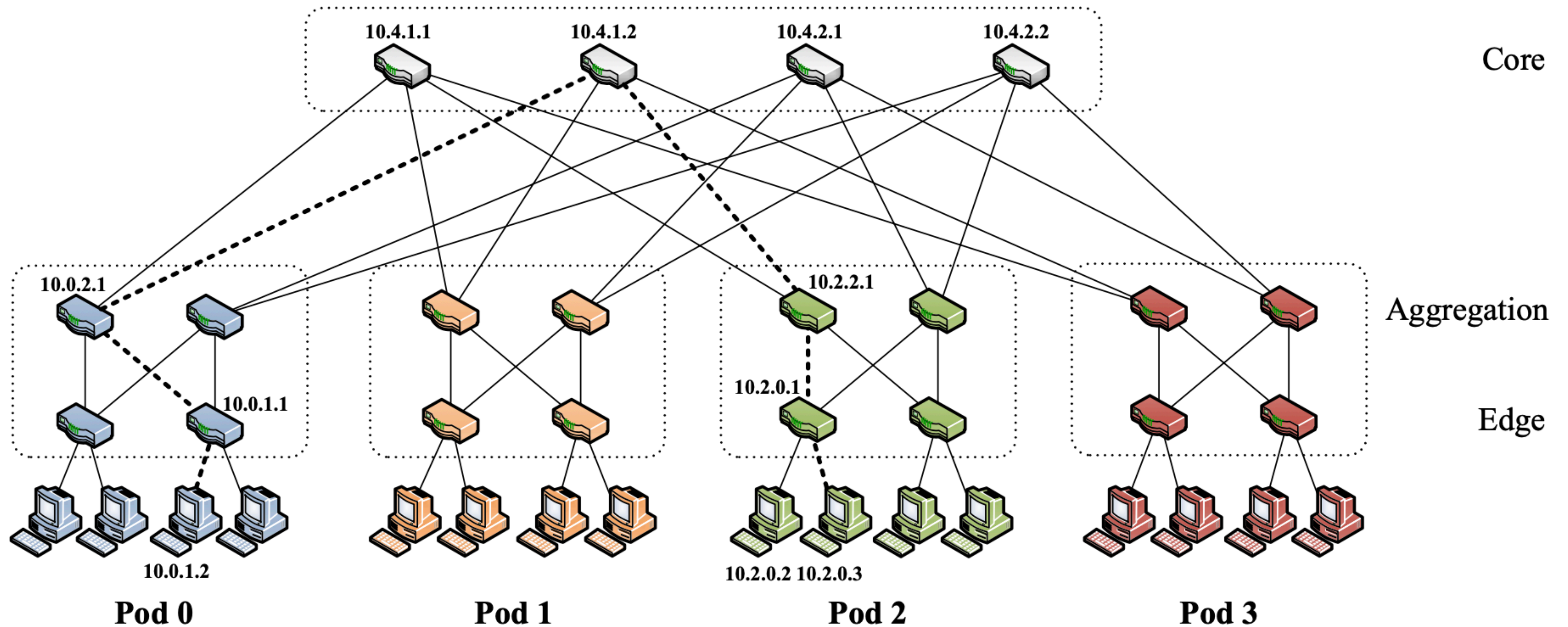
Proposal #3: Scalable DCNet (SIGCOMM'08)

- Key: co-design addressing and routing
- Target the k-array fat-tree topology, e.g., k=4



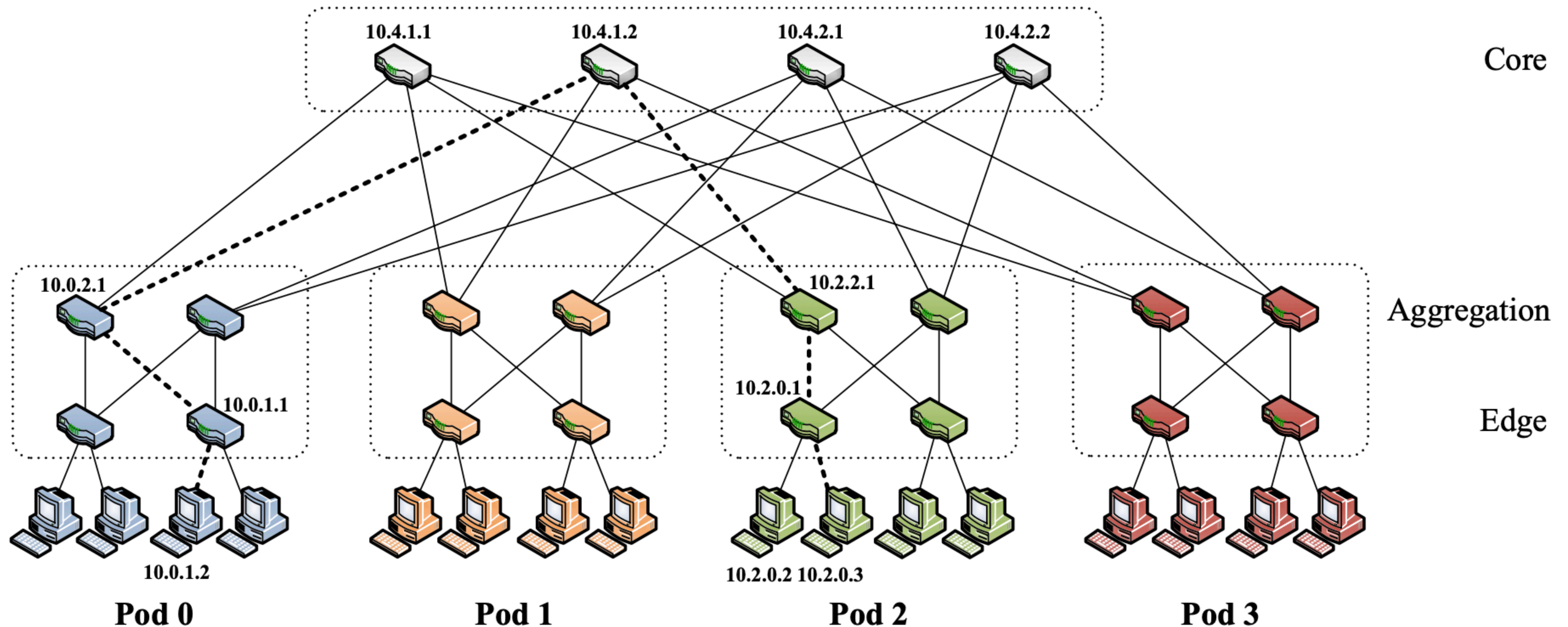
Addressing: pod switch

- 10.*pod*.*switch*.1
- pod:[0, k-1]



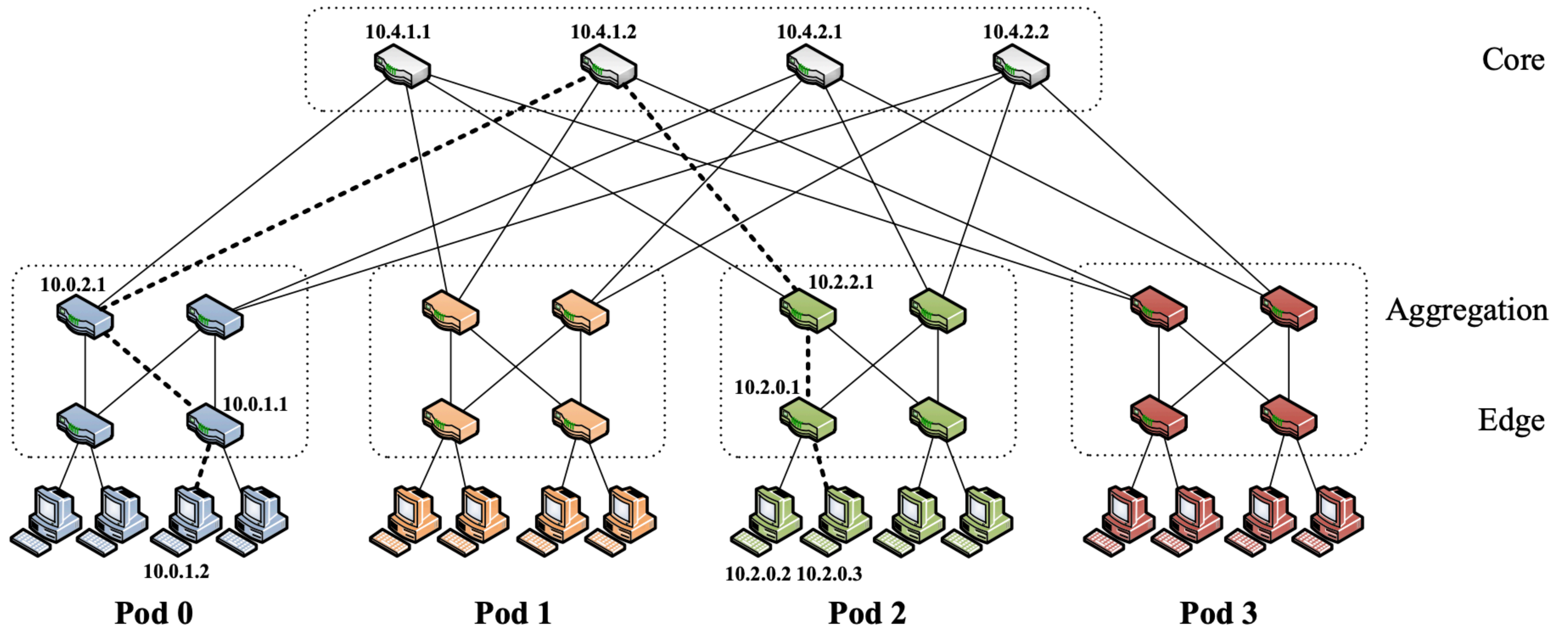
Addressing: pod switch

- 10.*pod*.*switch*.1
- switch:[0,k-1], starting from left to right, bottom to up



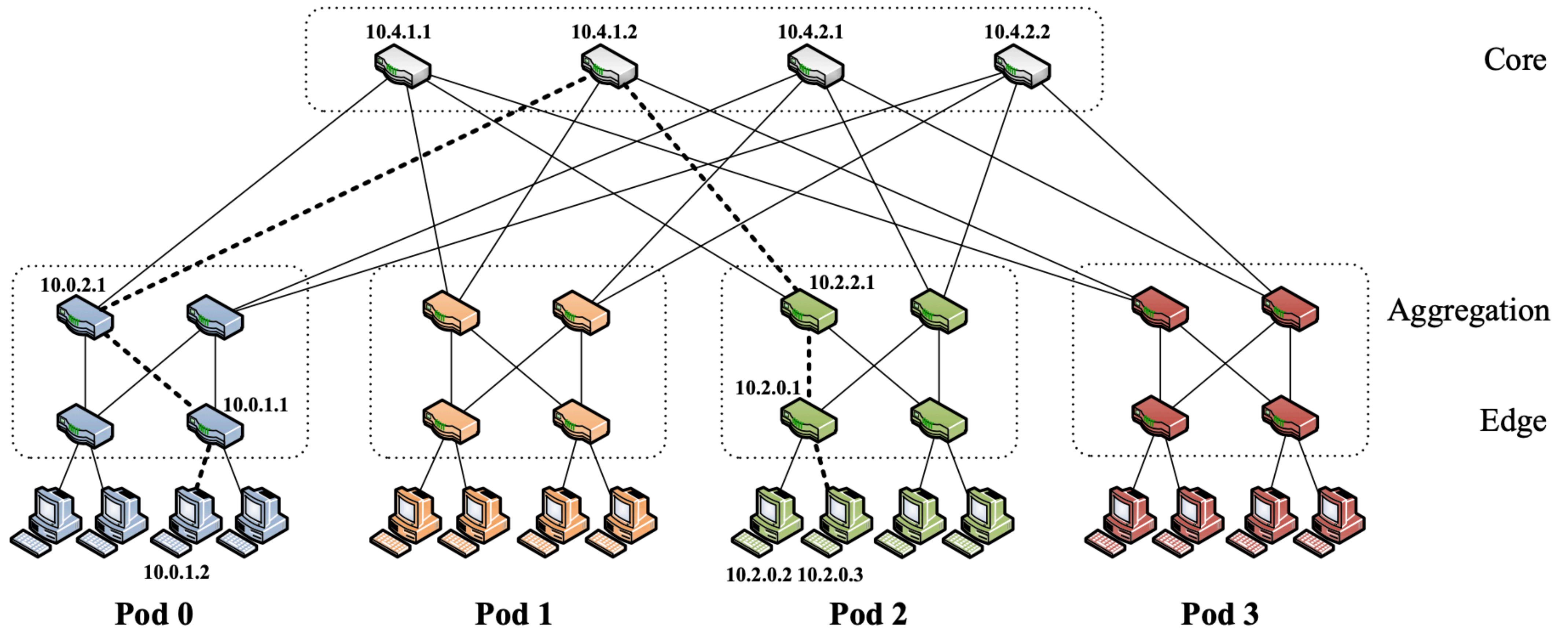
Addressing: core switch

- 10.*k*.*j*.*i*
- j and i is the switch's coordinates in the $(k/2)^2$ core switch grid



Addressing: host

- 10.*pod*.*switch*.*ID*
- ID:[2,k/2+1], starting from left to right



Routing Overview

- Two-Level Routing Table
 - Primary routing table: (prefix, port)
 - Secondary routing table: (suffix, port)
- Primary Table
 - Left-handed, i.e., /m prefix masks
- Secondary Table
 - Right-handed, i.e., /m suffix masks

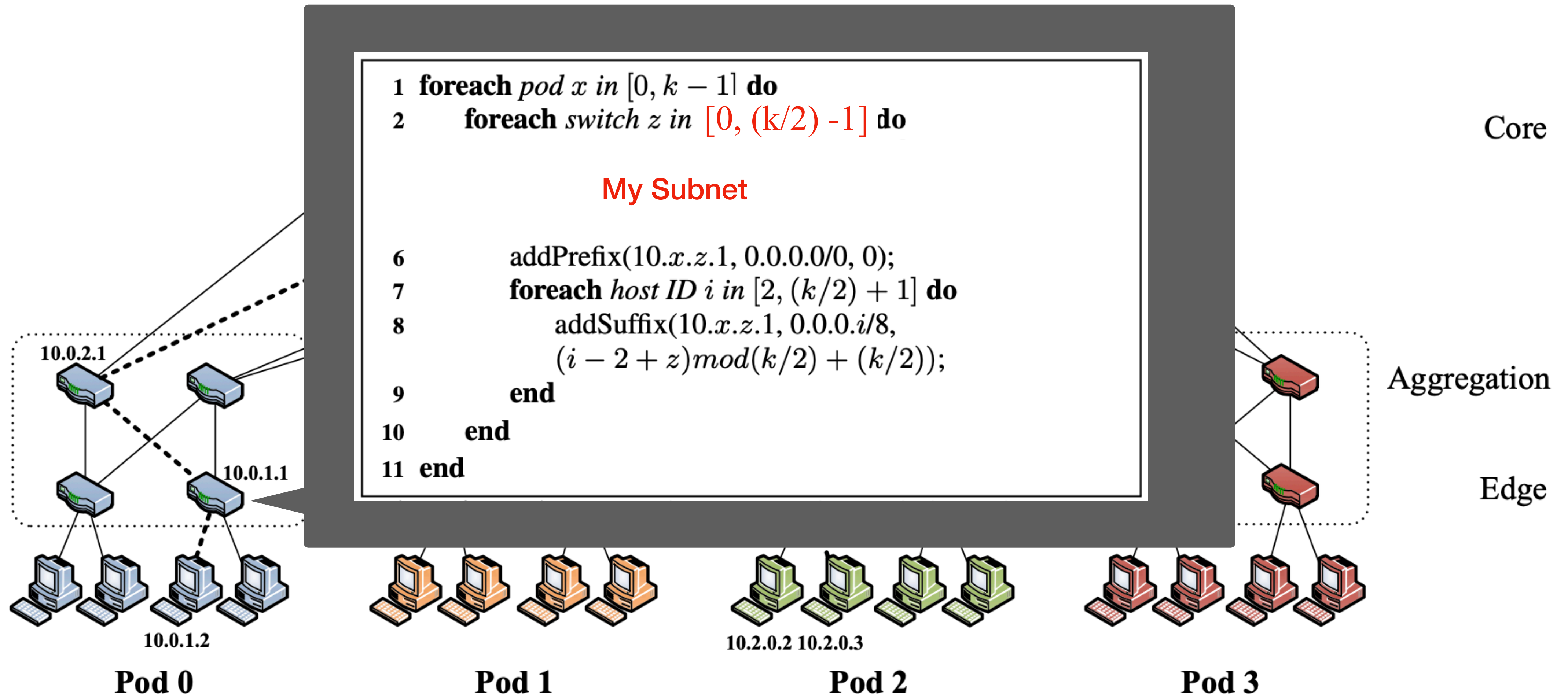
Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	



Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

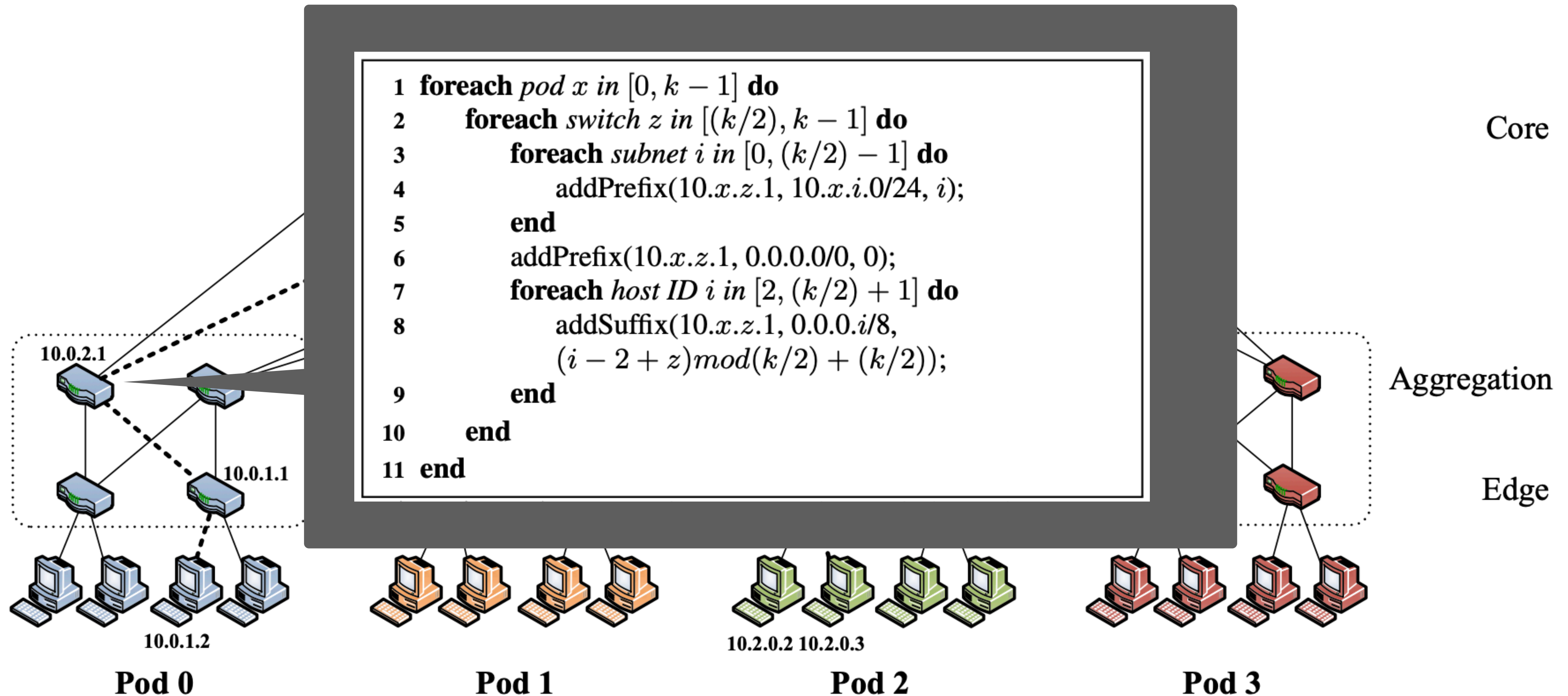
Routing @Edge Switch

- Two types of traffic: to hosts and to aggregation switches



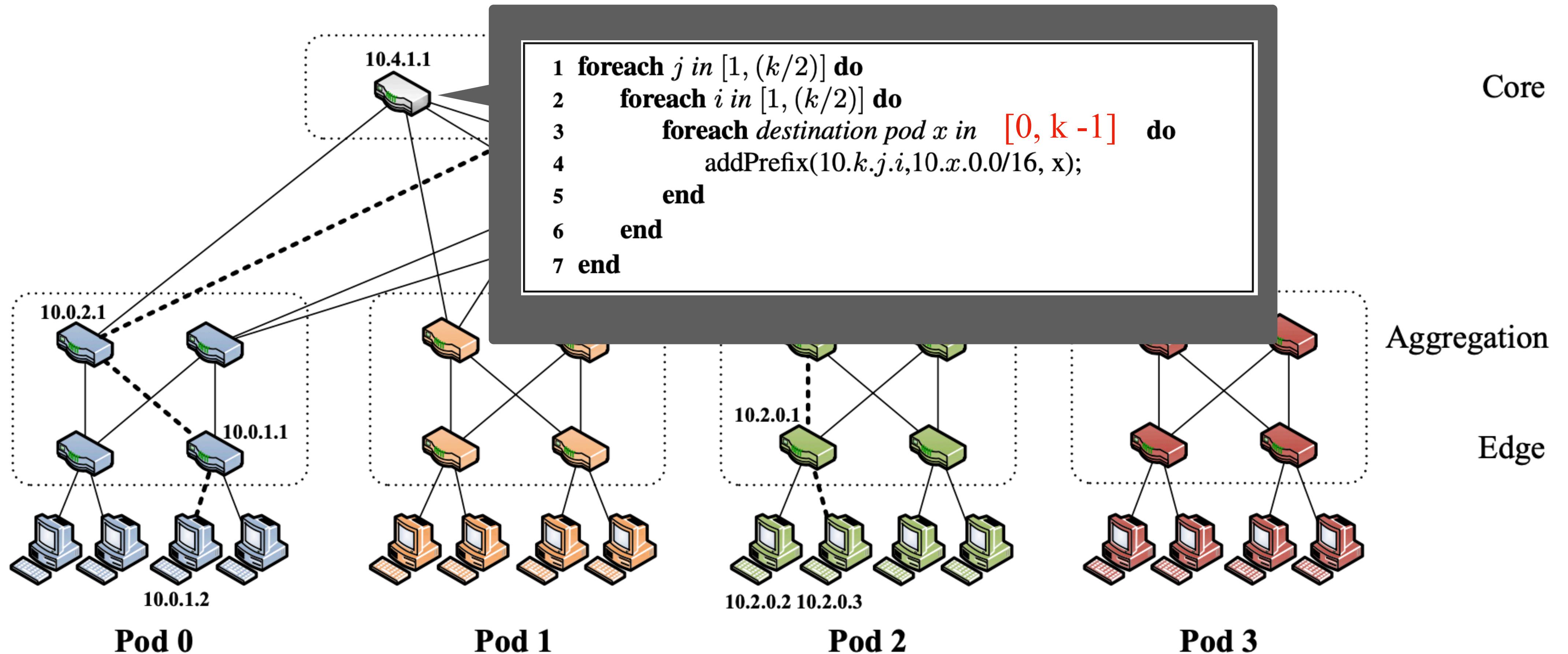
Routing @ Aggregation Switch

- Two types of traffic: to core switches and to edge switches



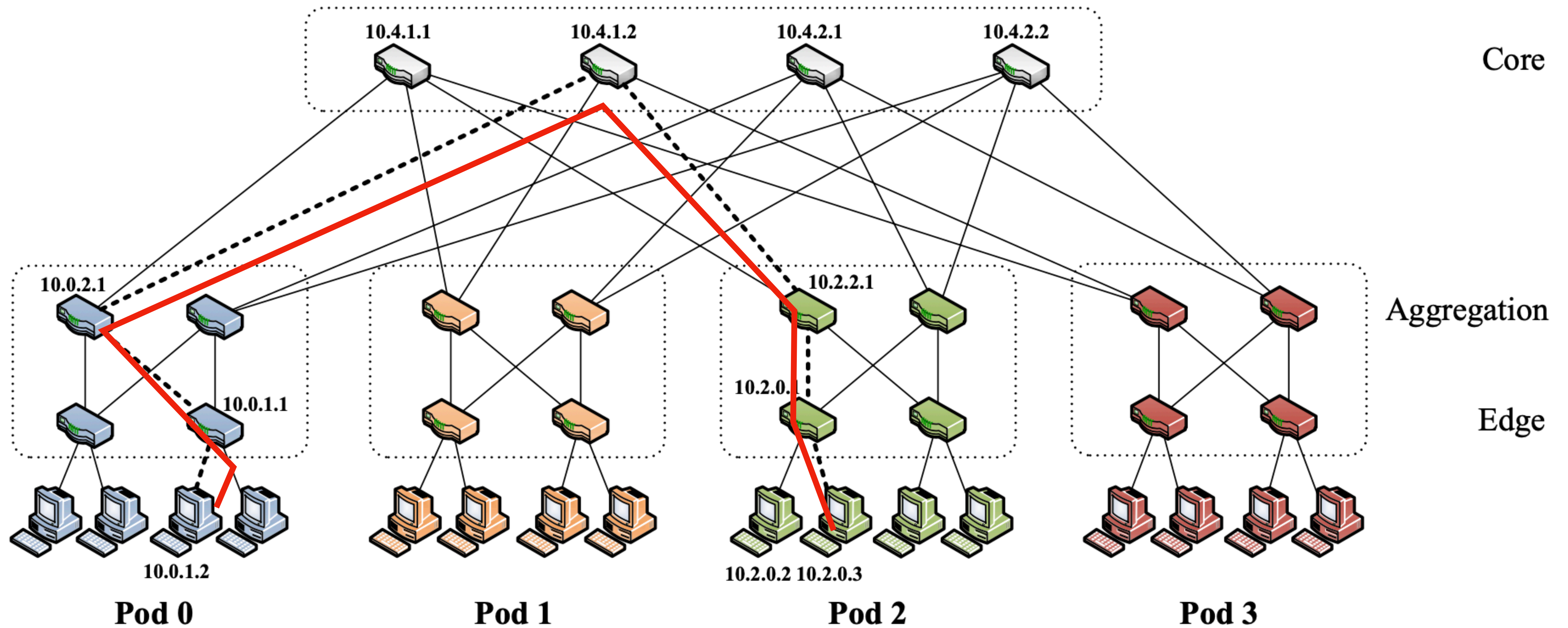
Routing @Core Switch

- One type of traffic: to aggregation switches



An Example

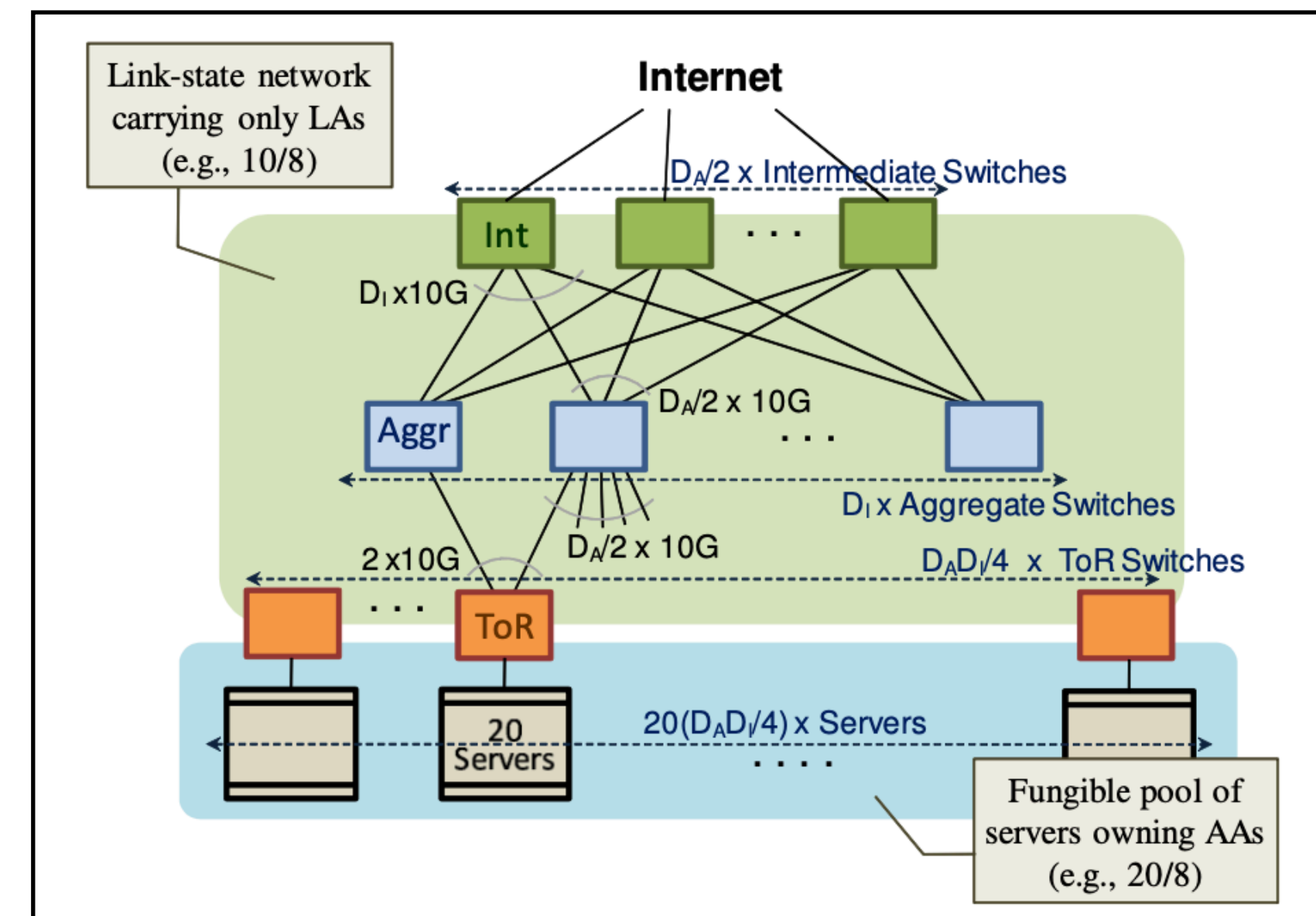
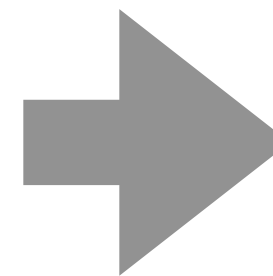
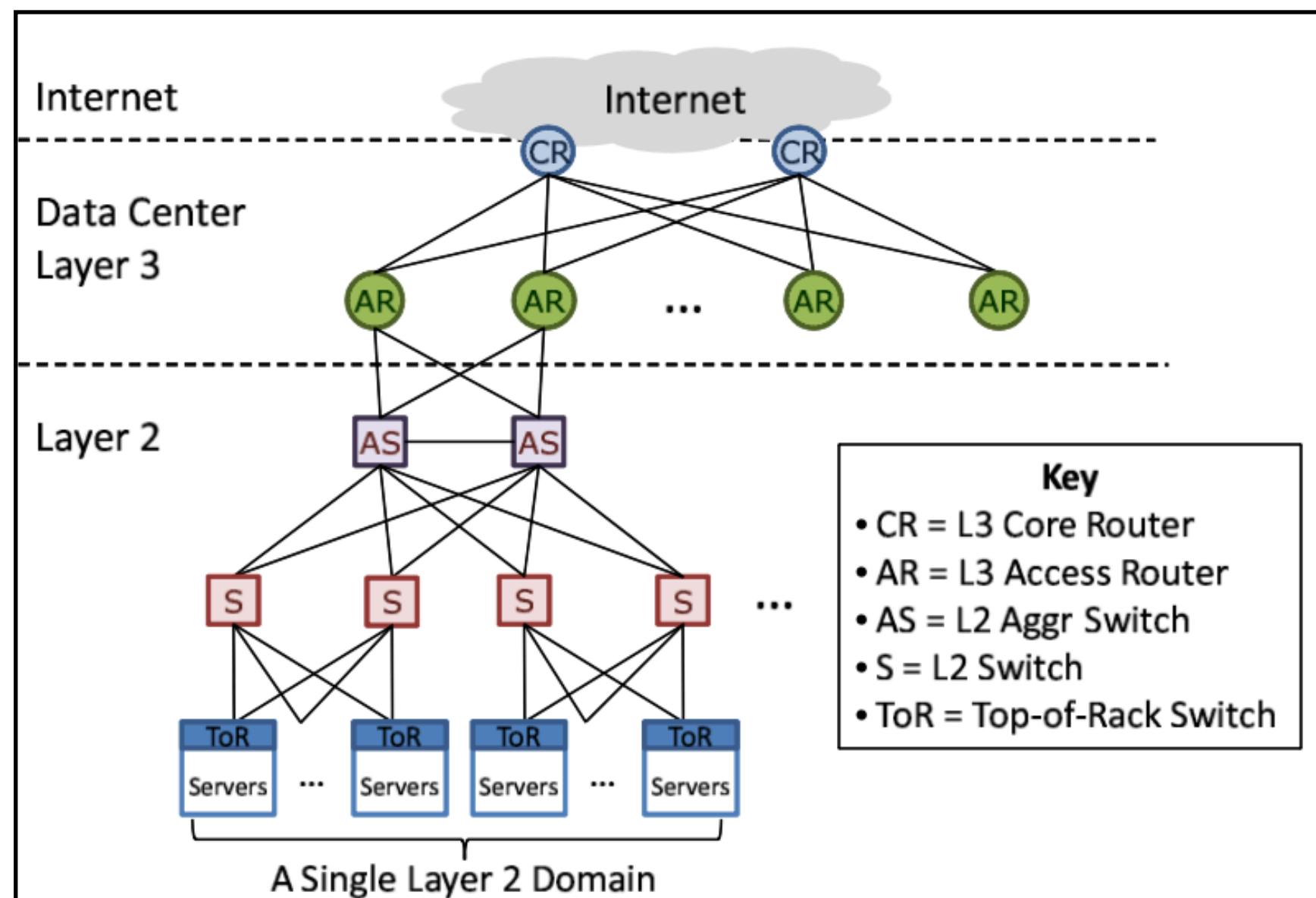
- 10.0.1.2 $\rightarrow$ 10.2.0.3



**Bring multi-path communication between
two edge switches!**

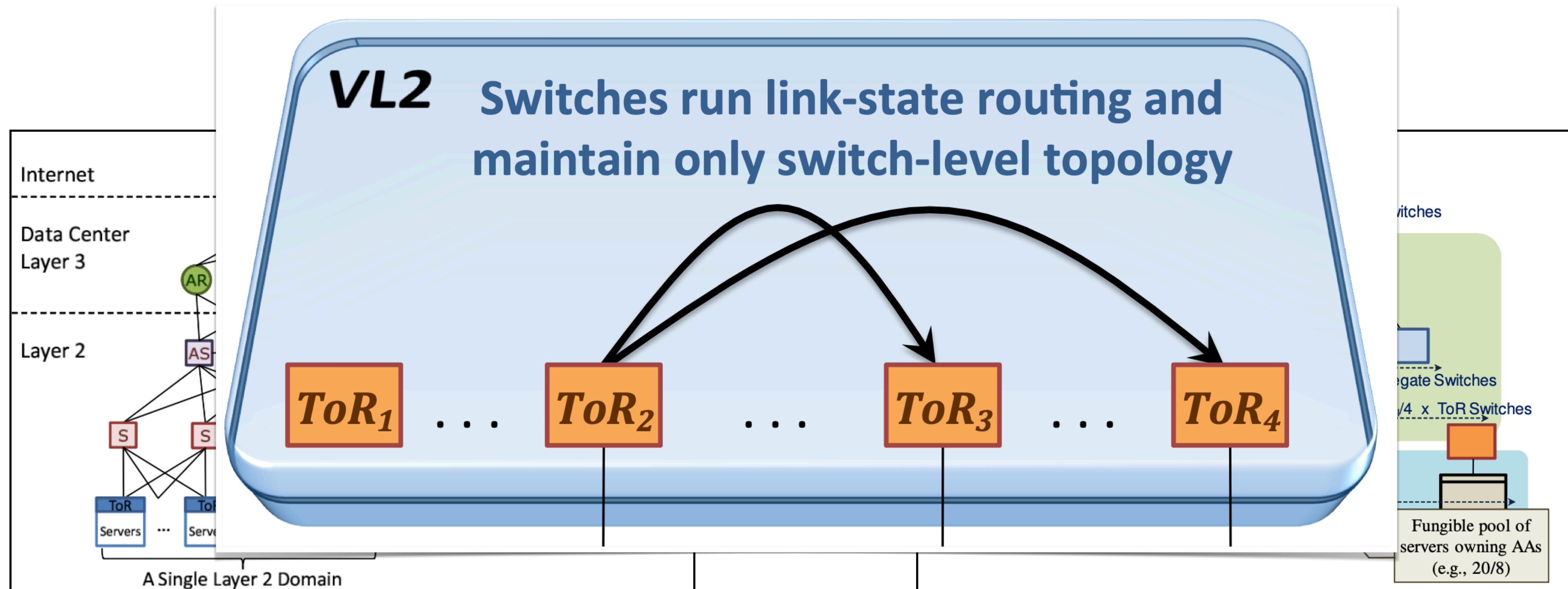
Proposal #4: VL2 (SIGCOMM'09)

- A virtual layer 2 offering massive aggregation bandwidth
 - A big L2 Ethernet switch



Proposal #4: VL2 (SIGCOMM'09)

- A virtual layer 2 offering massive aggregation bandwidth
 - A big L2 Ethernet switch



Addressing

- Key: name-location separation
 - Introduce a directory service!
- Address assignment
 - Applications use application-specific IP addresses (AA)
 - Each AA is associated with a location-specific address (LA)

Addressing

- Key: name-location separation
 - Introduce a directory service!
- Address assignment
 - Applications use application-specific IP addresses (AA)
 - Each AA is associated with a location-specific address (LA)

What is an LA?

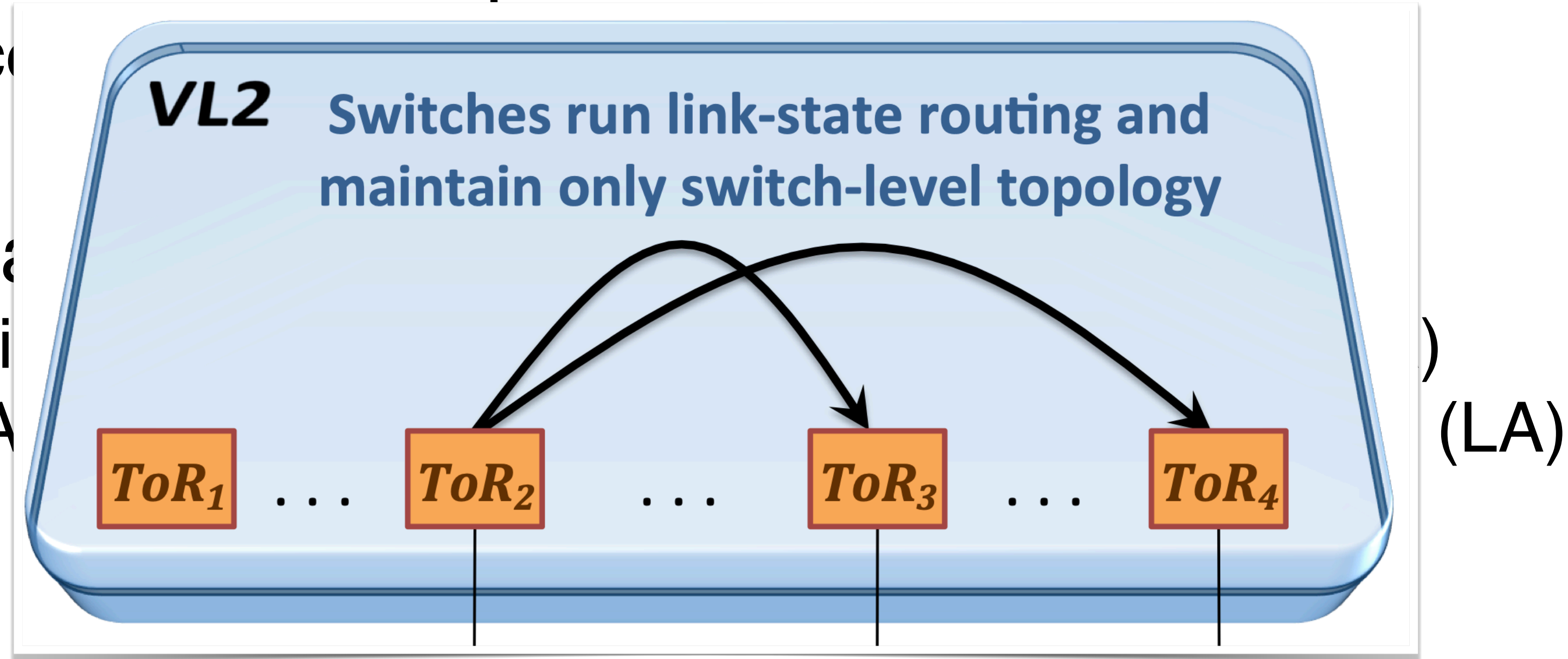
Addressing

- Key: name-location separation

- Introduces

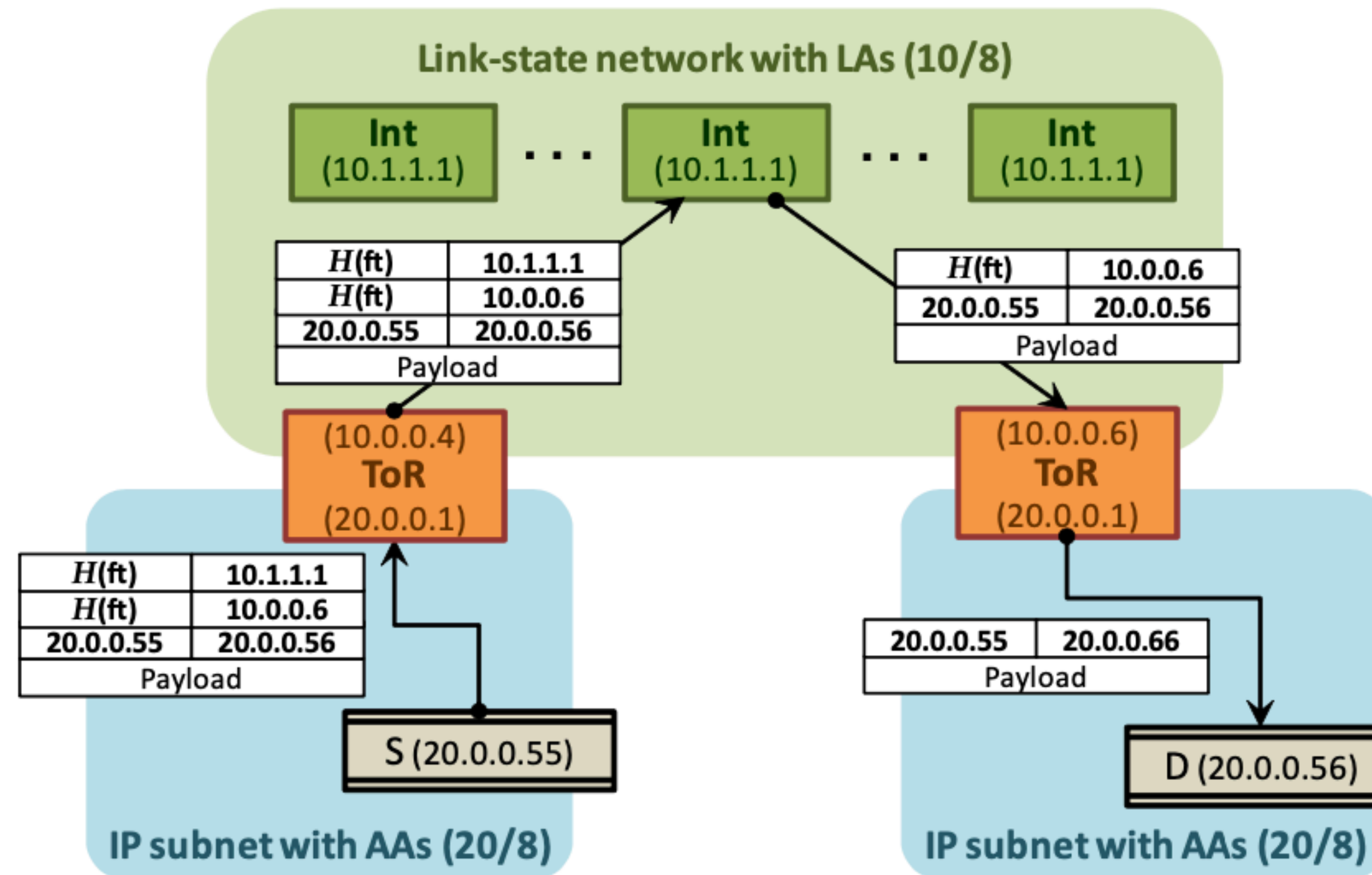
- Addressing

- Application
 - Each AA



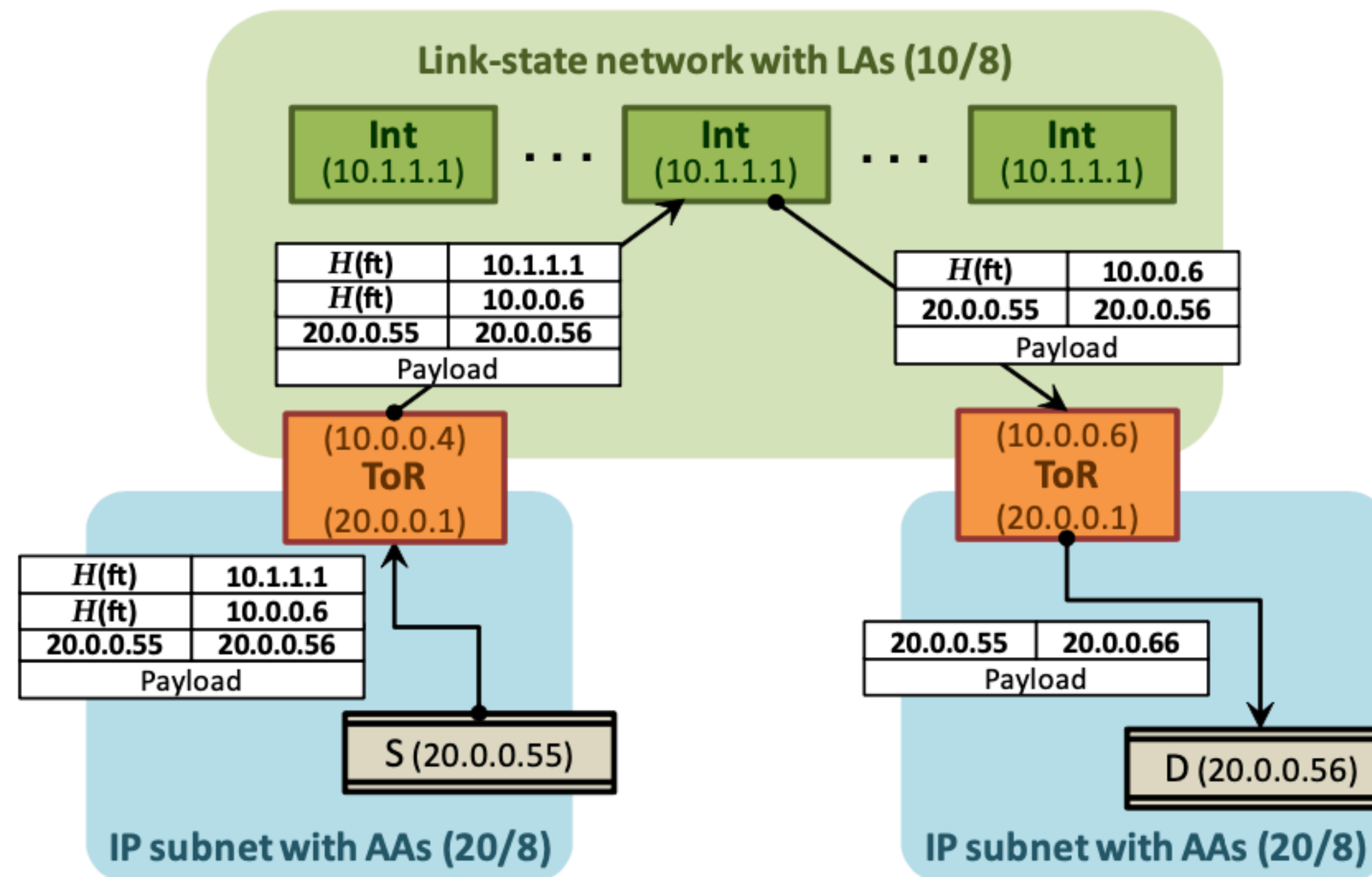
The IP address of the connected ToR!

How to Route Packets



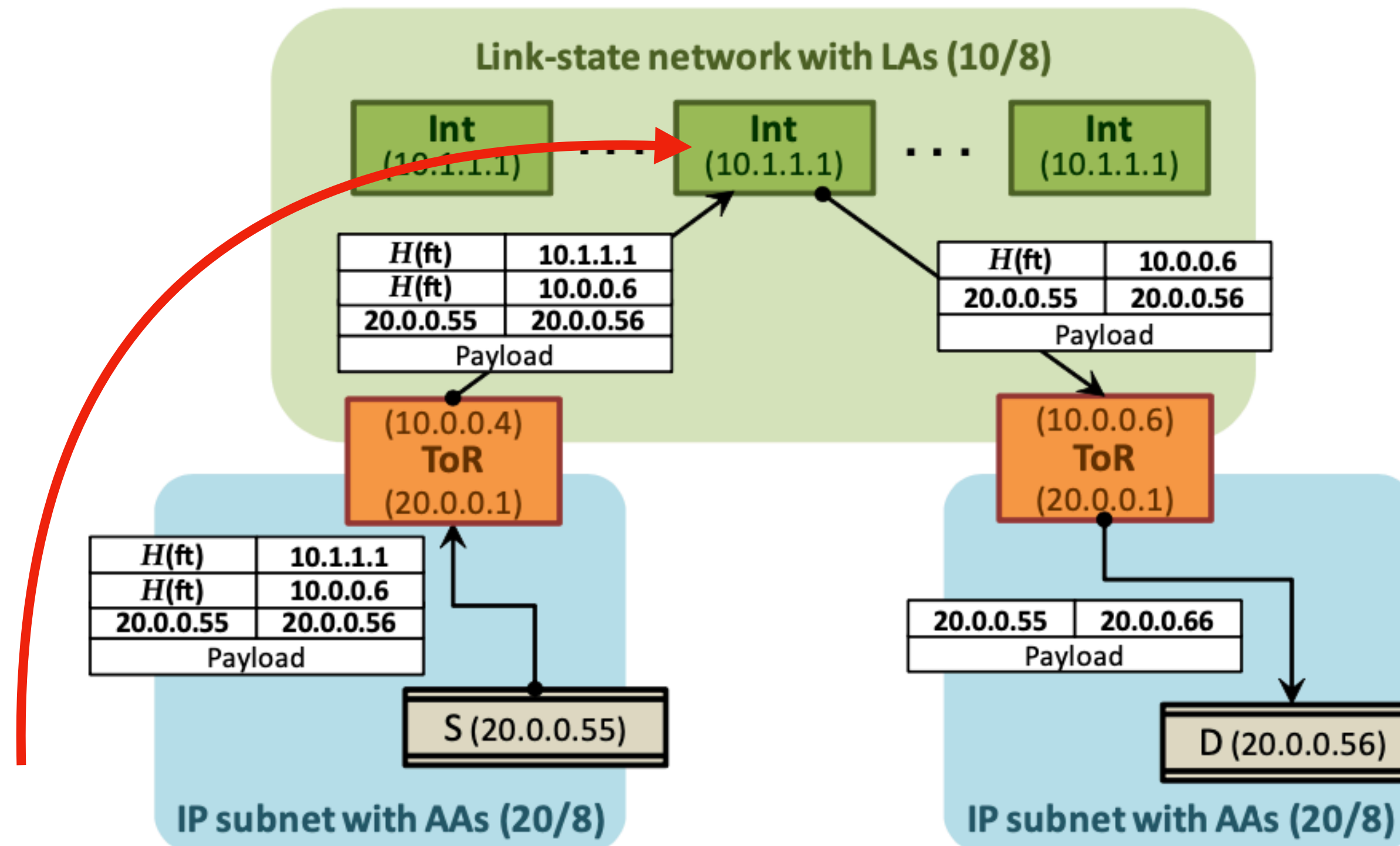
How to Route Packets

- Key: updating the destination address along the path
 - Look up the directory service



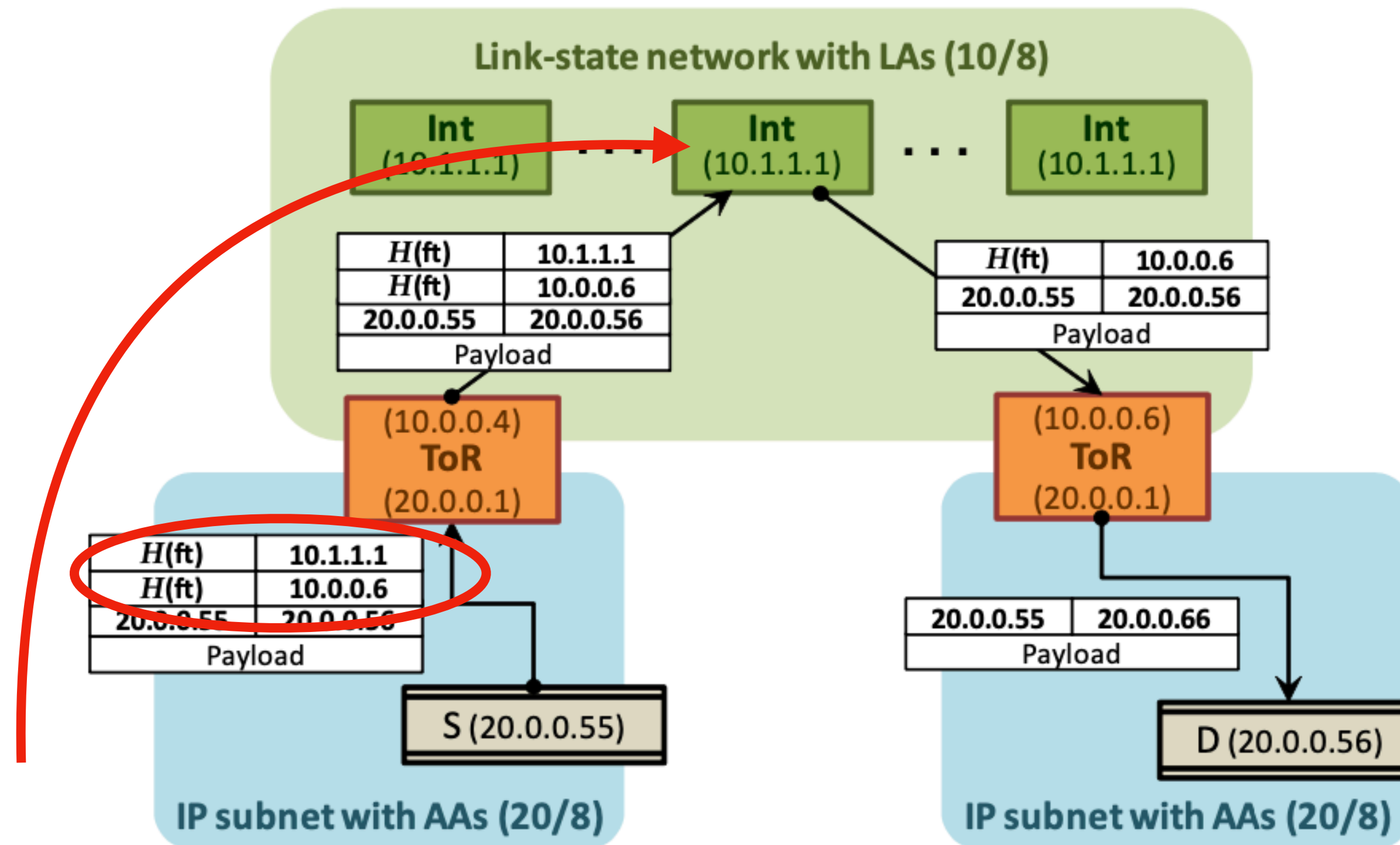
How to Route Packets

- Key: updating the destination address along the path
 - Look up the directory service



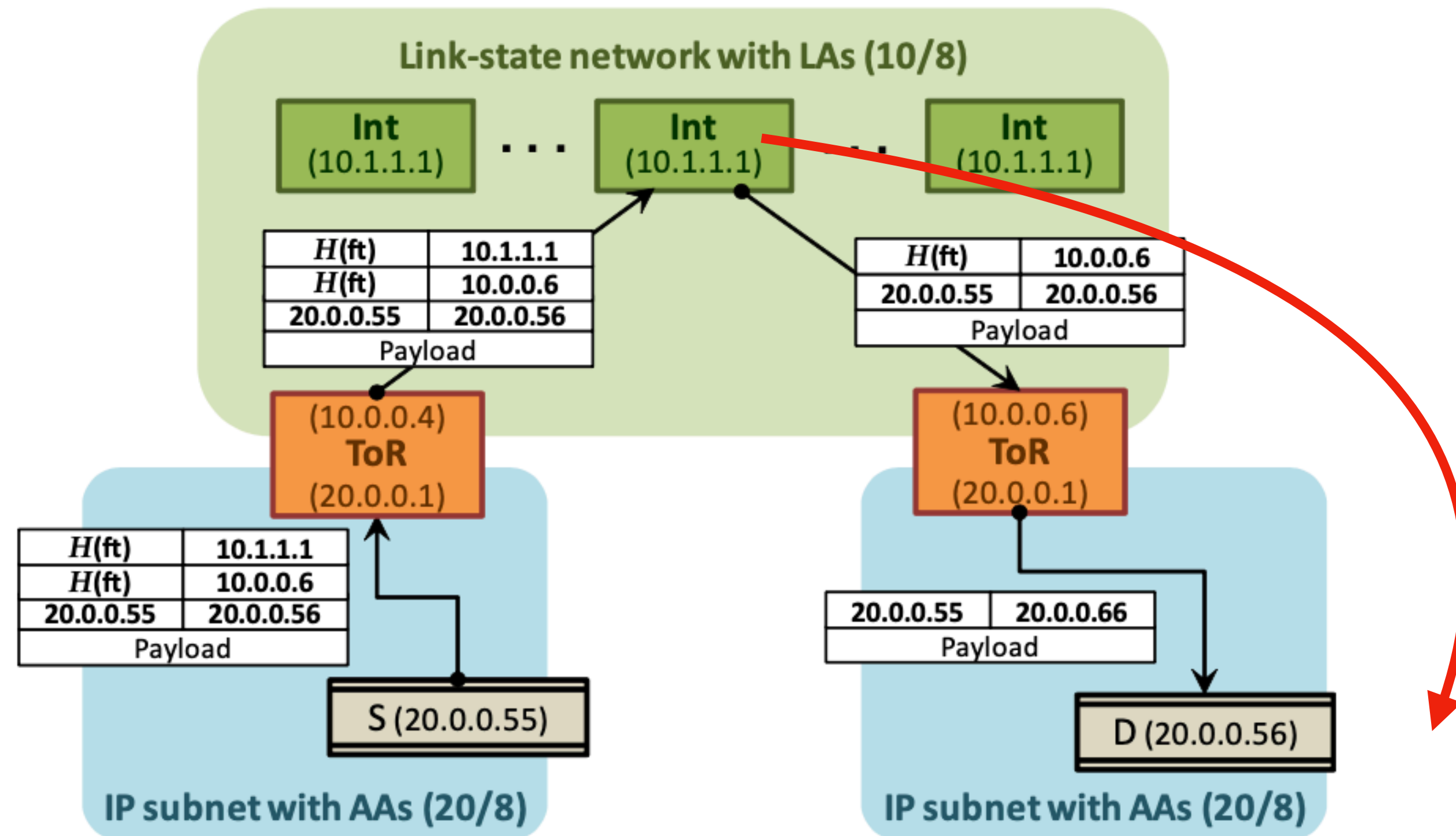
How to Route Packets

- Key: updating the destination address along the path
 - Look up the directory service



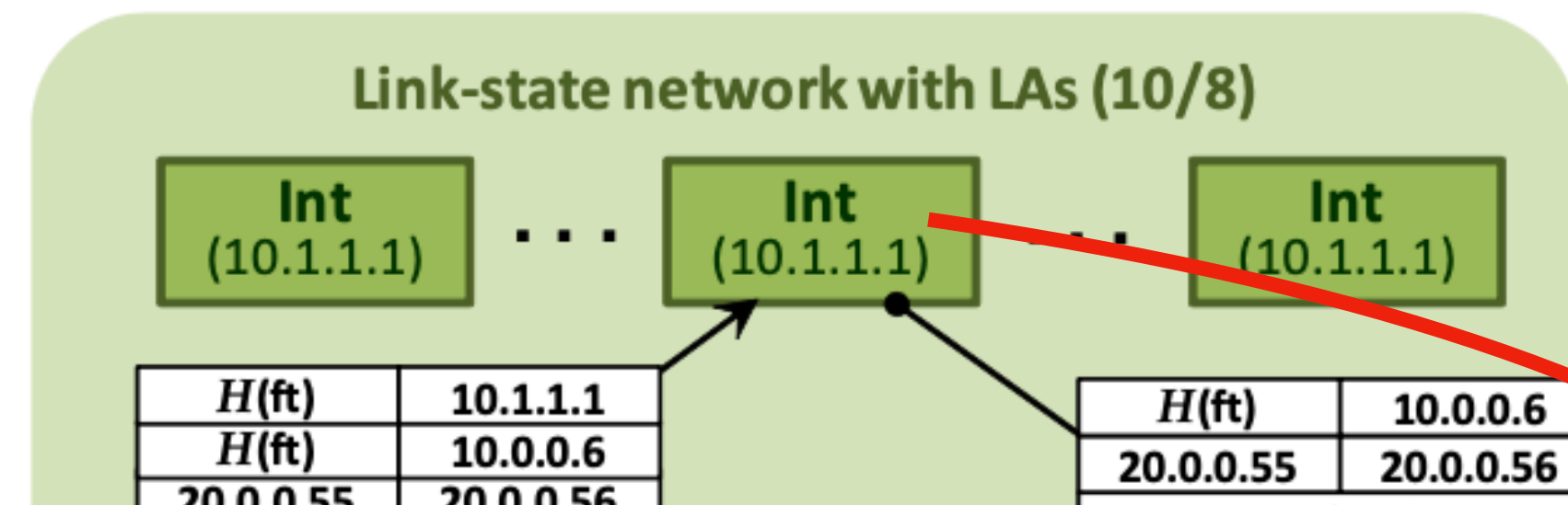
How to Route Packets

- Key: updating the destination address along the path
 - Look up the directory service



How to Route Packets

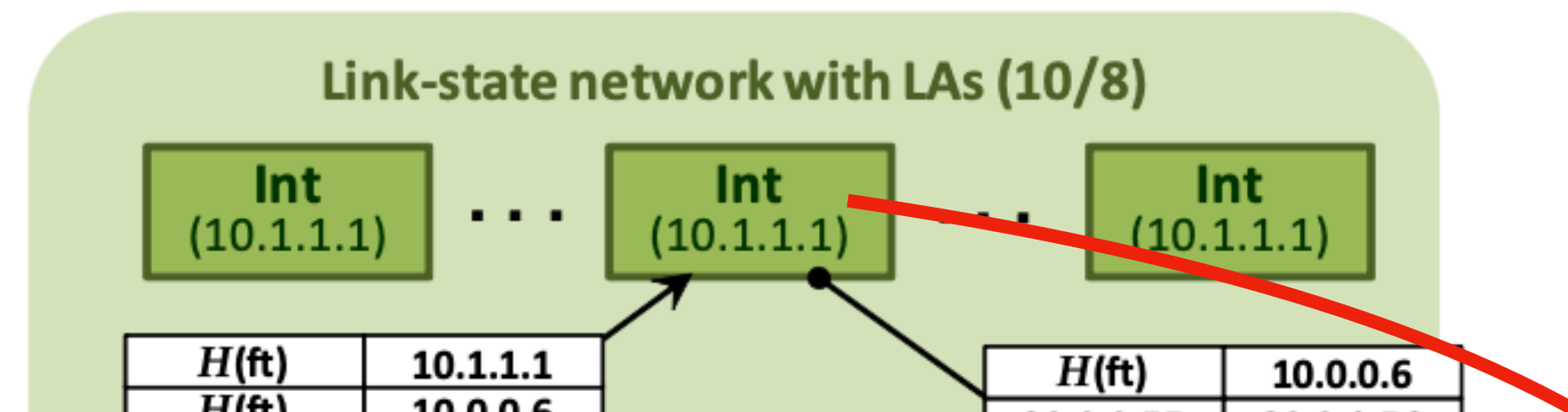
- Key: updating the destination address along the path
 - Look up the directory service



How does VL2 take advantage of multi-path, especially given that the new virtual lay runs link-state routing protocol?

How to Route Packets

- Key: updating the destination address along the path
 - Look up the directory service



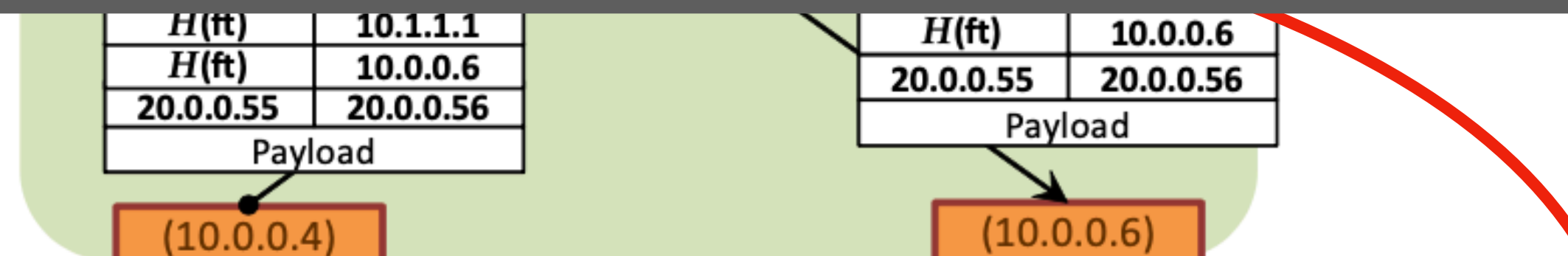
- VL2 assigns the same LA to all intermediate switches
- Rely on the directory service to make the decision!

IP subnet with AAs (20/8)

IP subnet with AAs (20/8)

How to Route Packets

- Key: updating the destination address along the path
- VL2 assigns the same LA to all intermediate switches
- Rely on the directory service to make the decision!



Bring multi-path communication between two hosts!

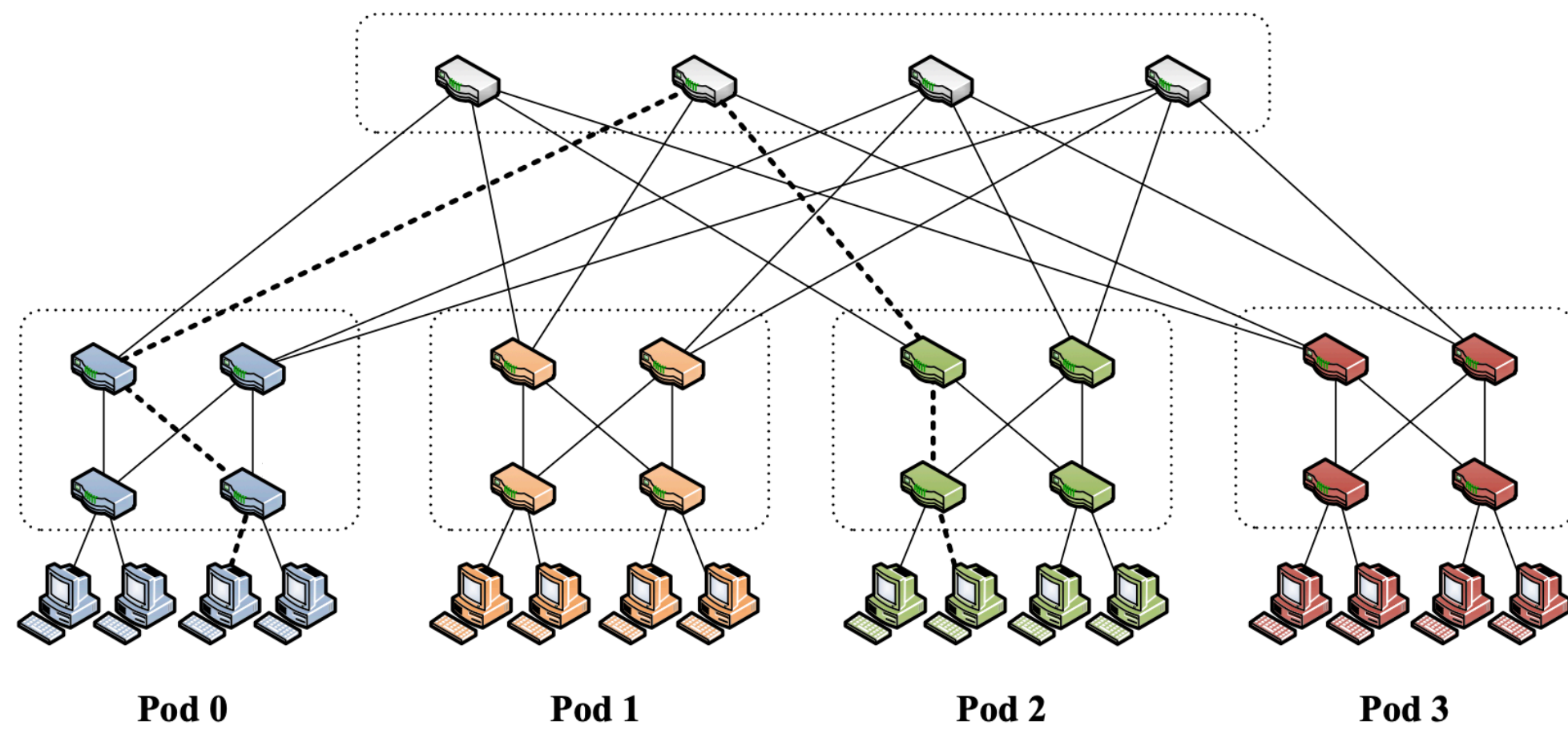
IP subnet with AAs (20/8)

IP subnet with AAs (20/8)

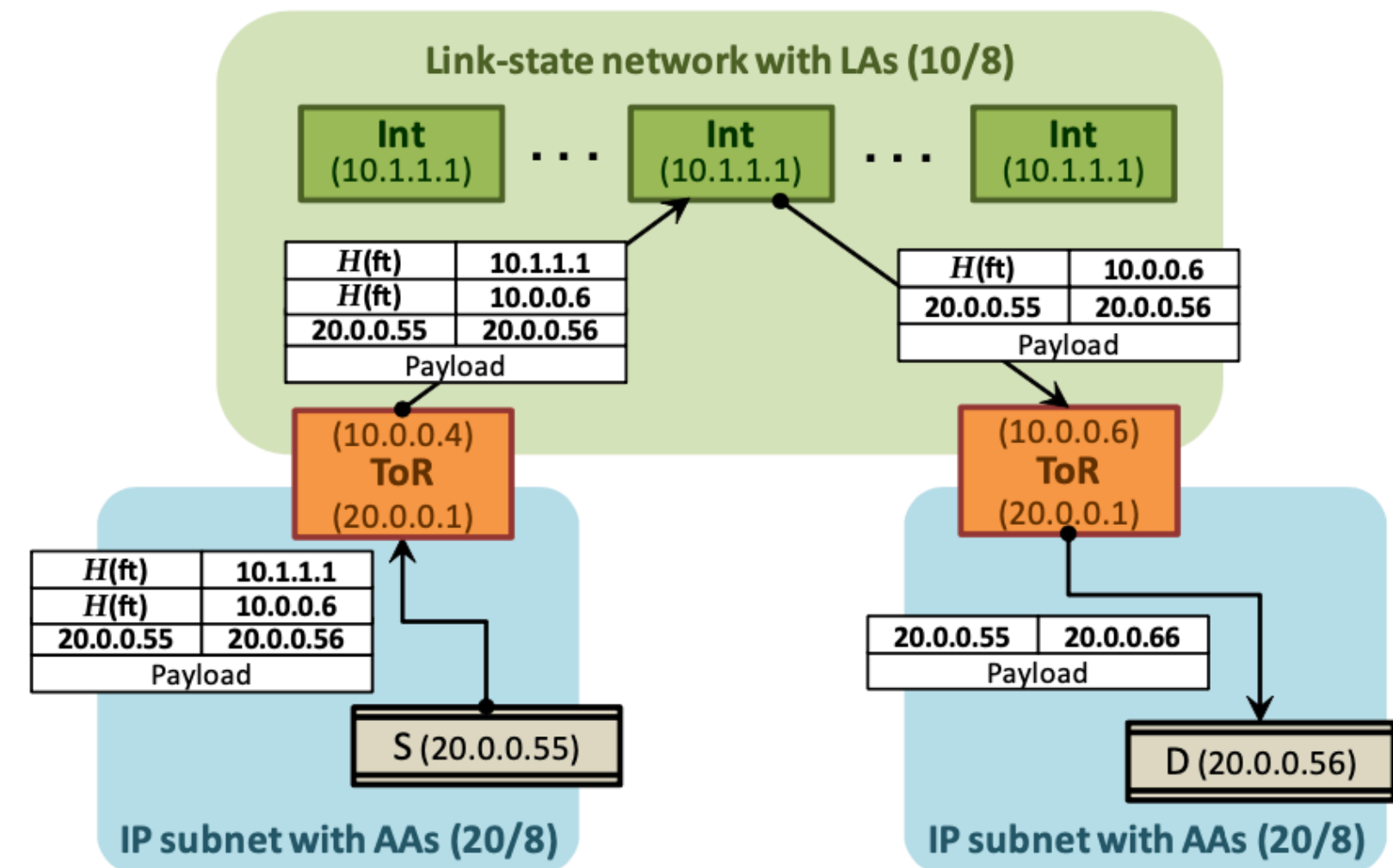
Proposal #4: Pros and Cons

- Pros:
 - Better performance than proposal #3
 - Better availability than proposal #3
- Cons:
 - Complexity, e.g., hardware layering and software directory service

Proposal #3 vs. Proposal #4

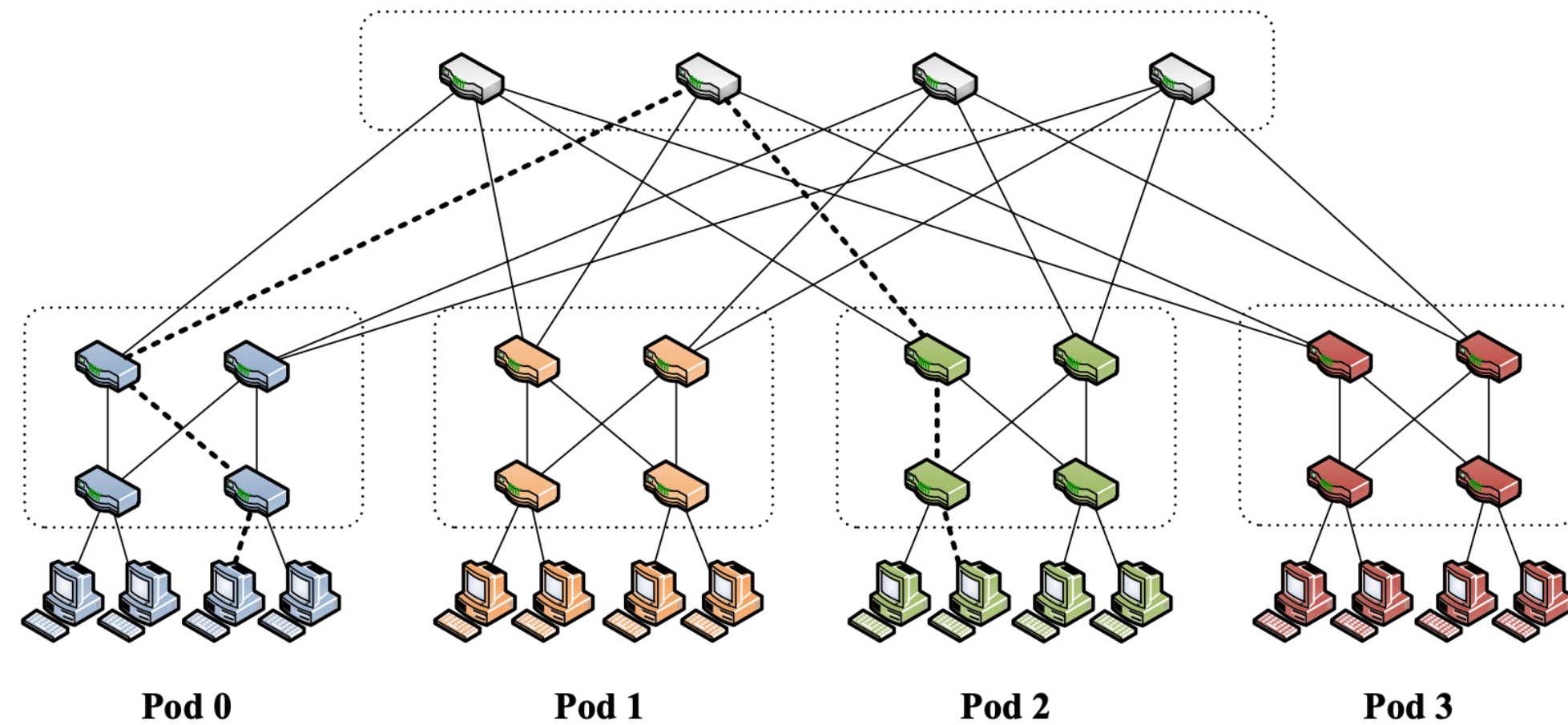


Proposal #3



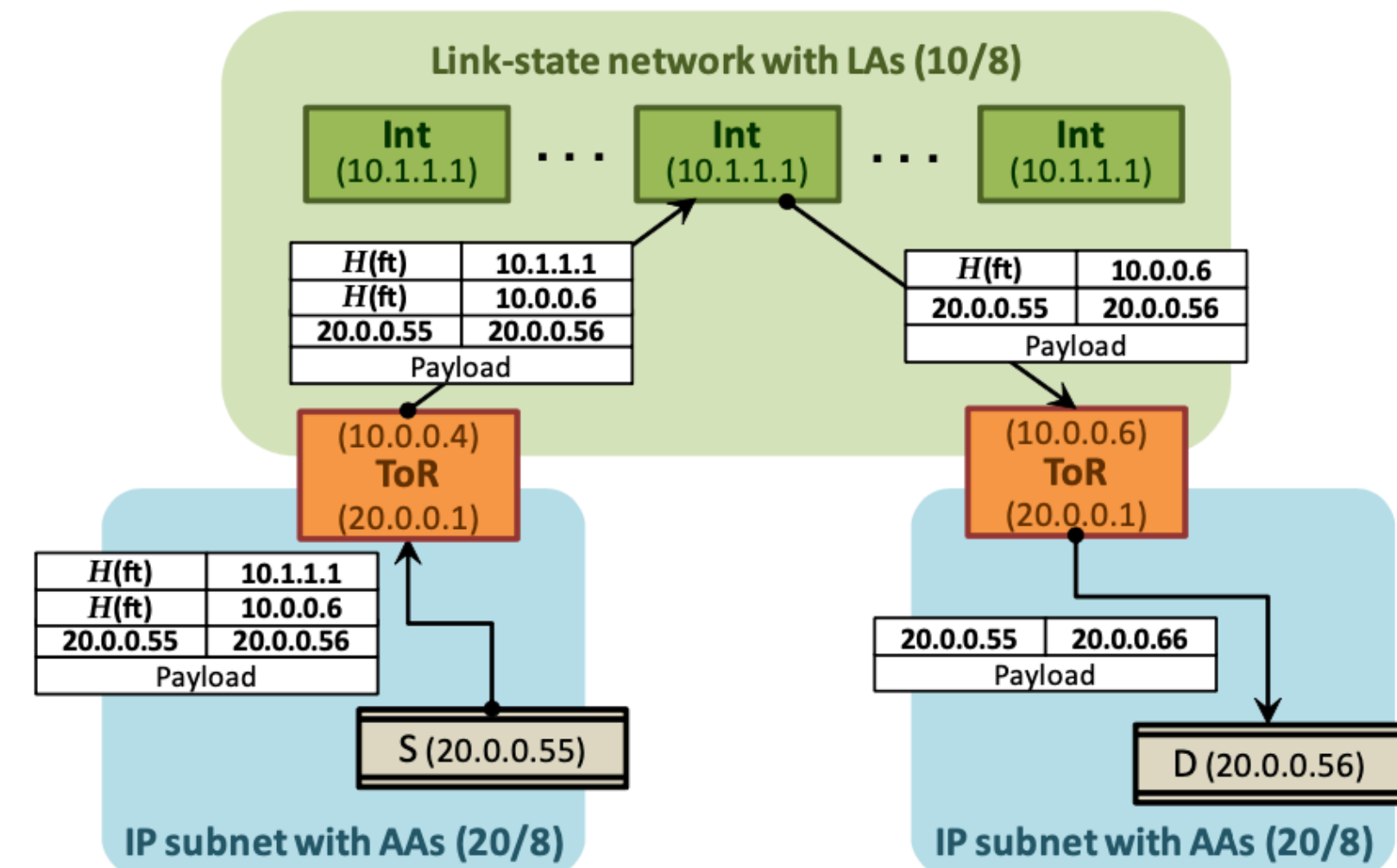
Proposal #4

Proposal #3 vs. Proposal #4



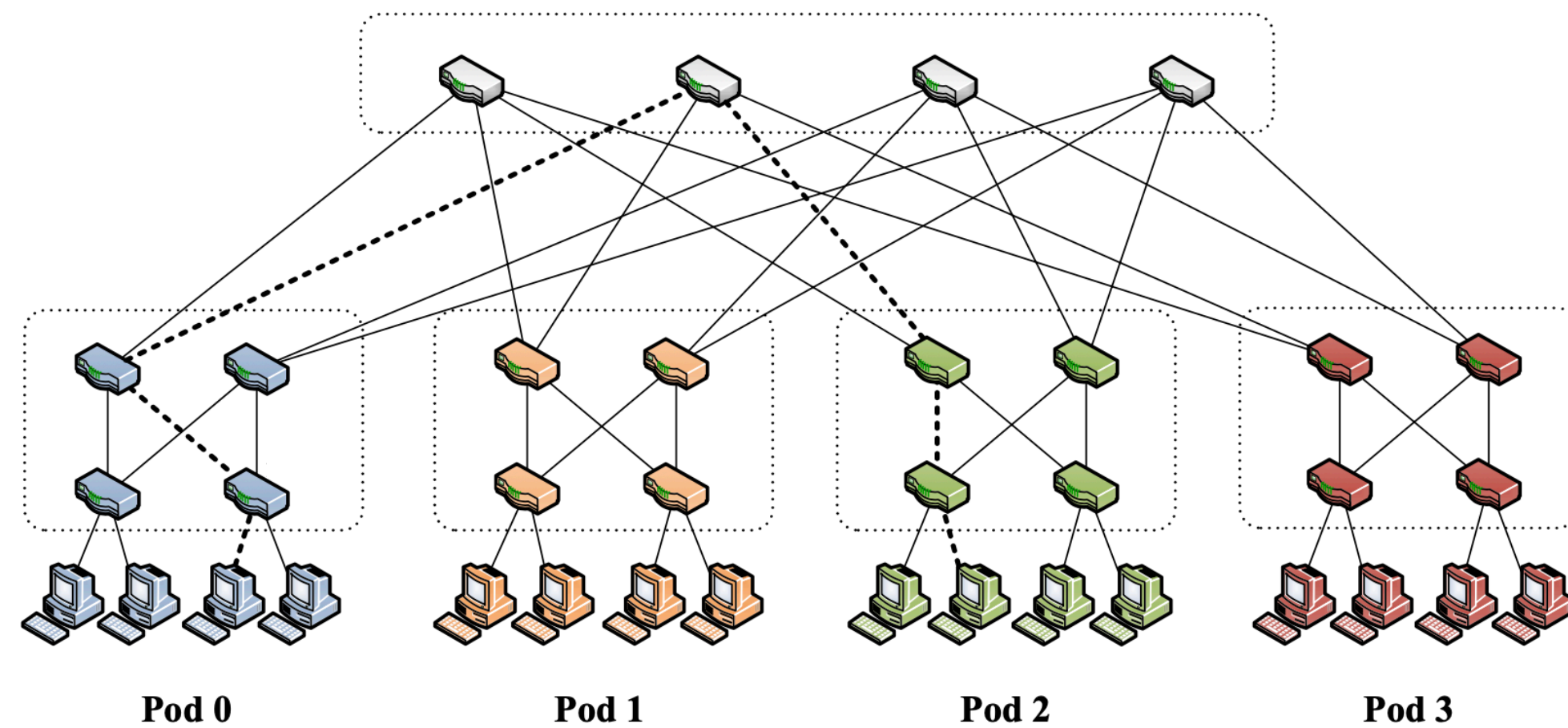
Proposal #3

Co-design routing and addressing



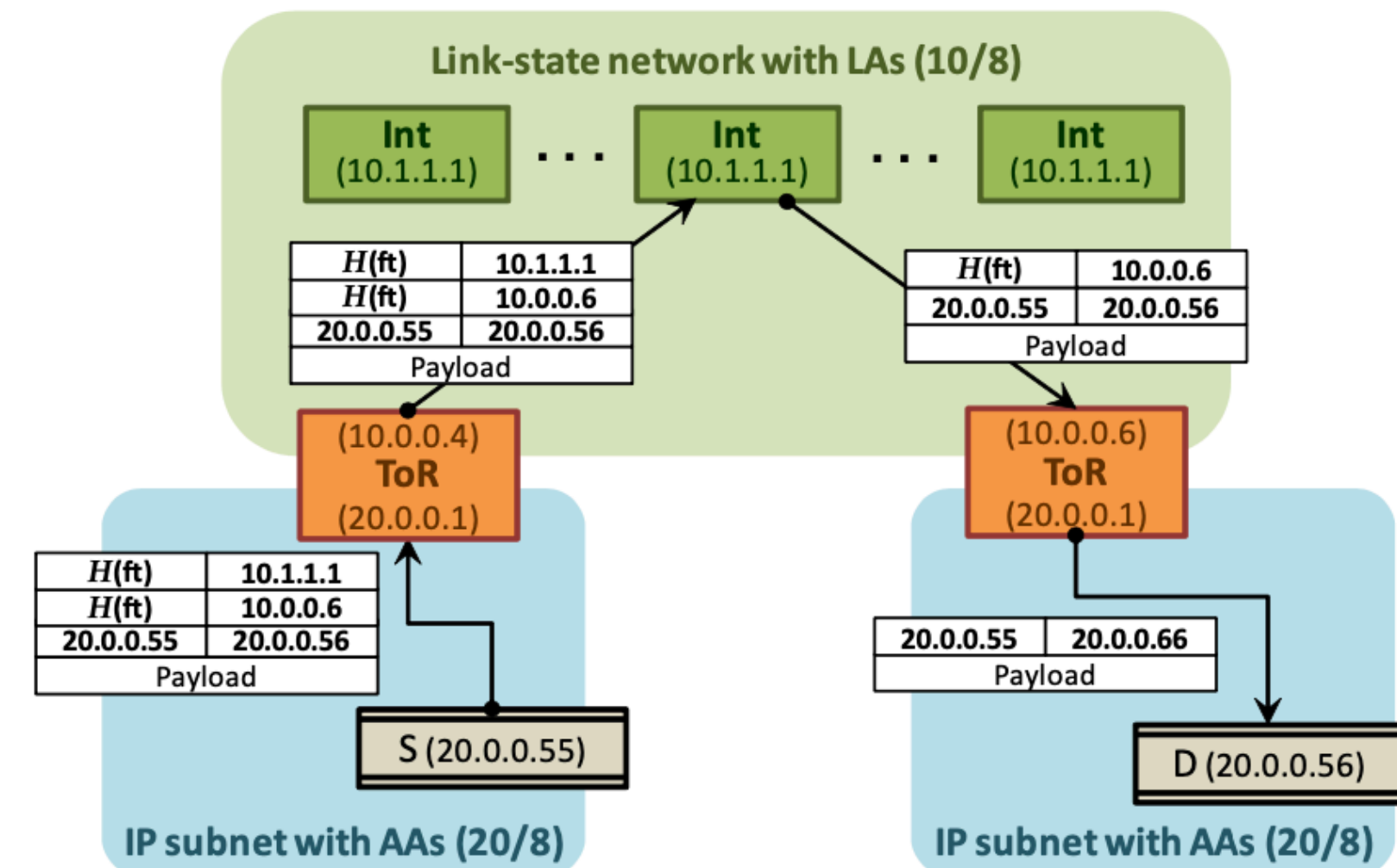
Proposal #4

Proposal #3 vs. Proposal #4



Proposal #3

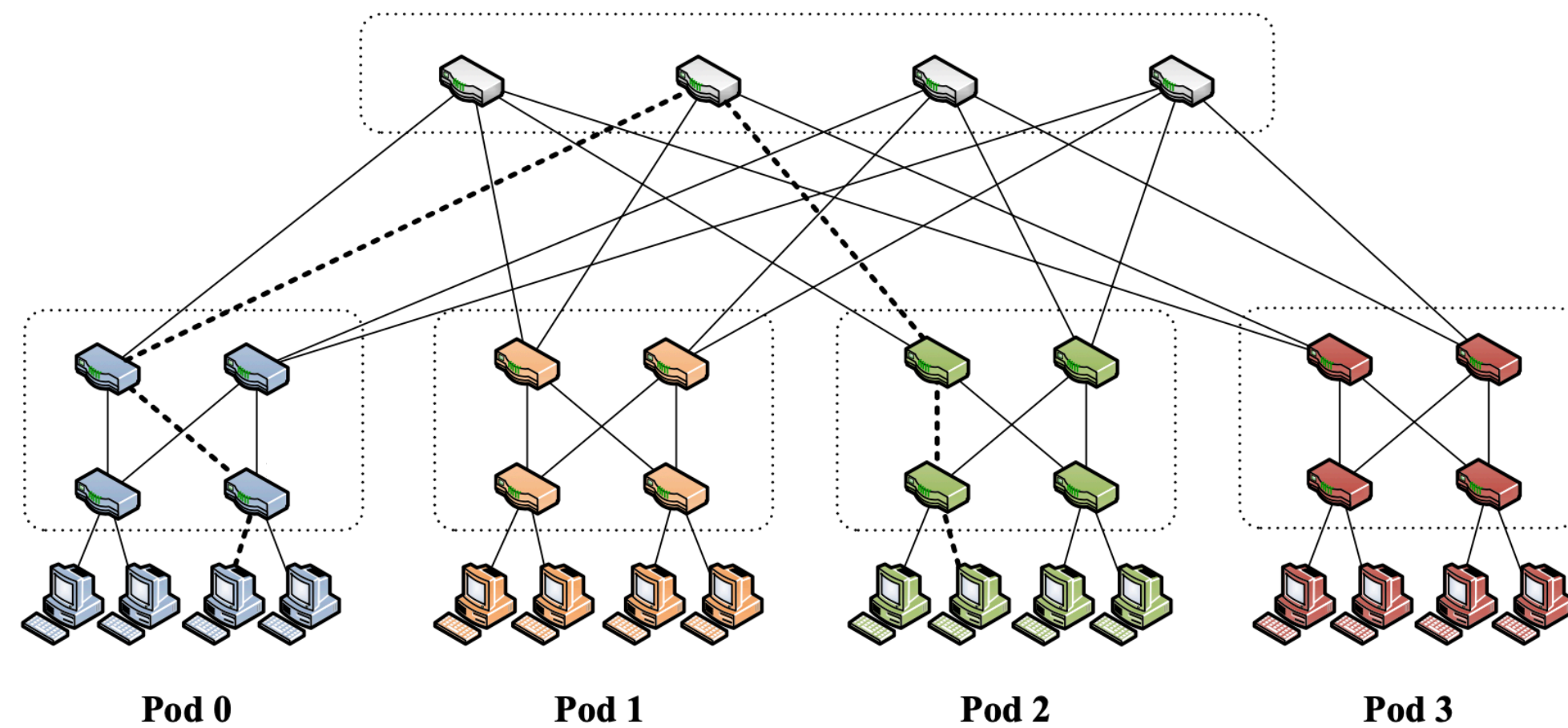
Co-design routing and addressing



Proposal #4

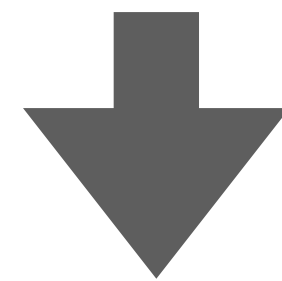
Decouple routing and addressing

Proposal #3 vs. Proposal #4

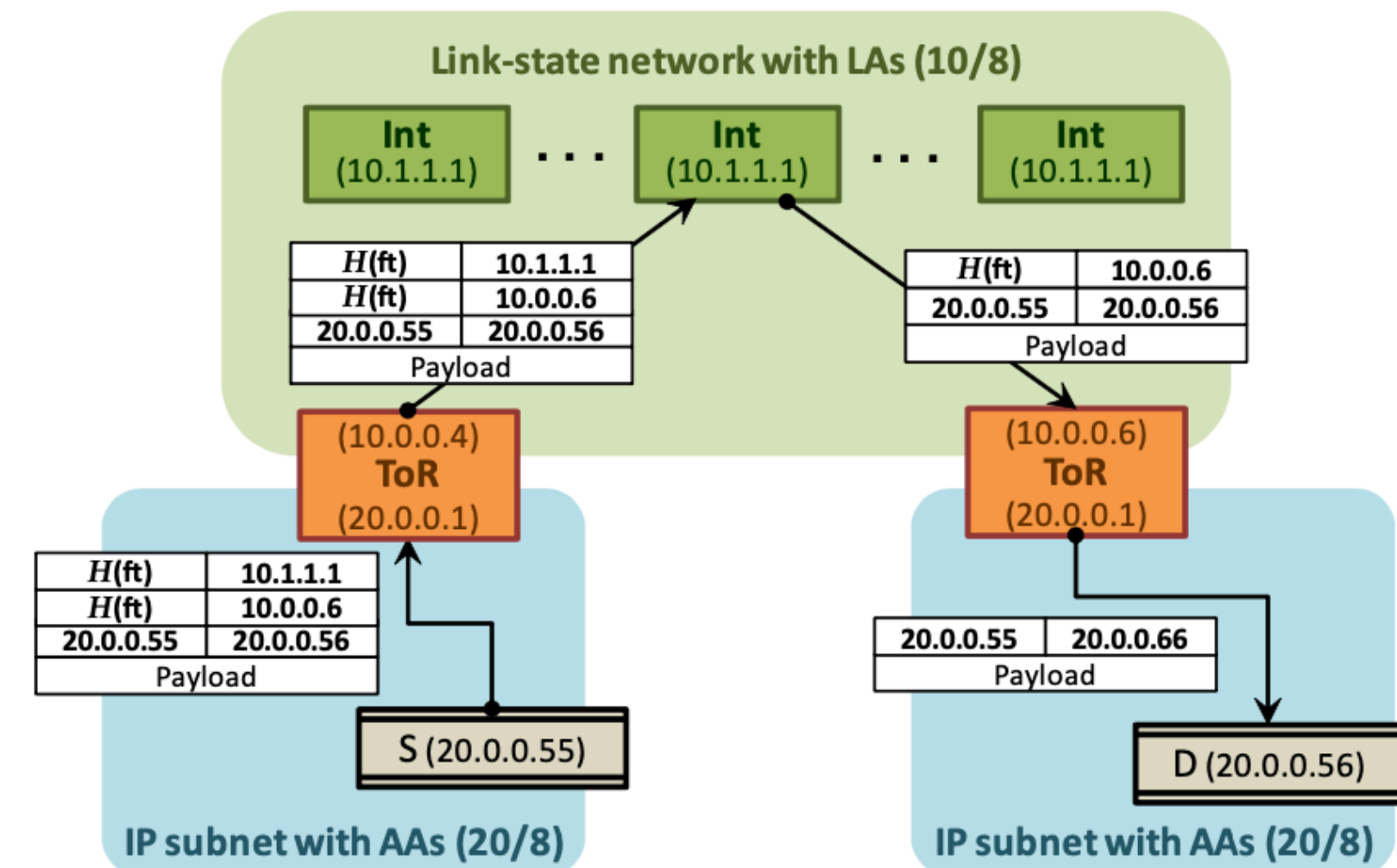


Proposal #3

Co-design routing and addressing



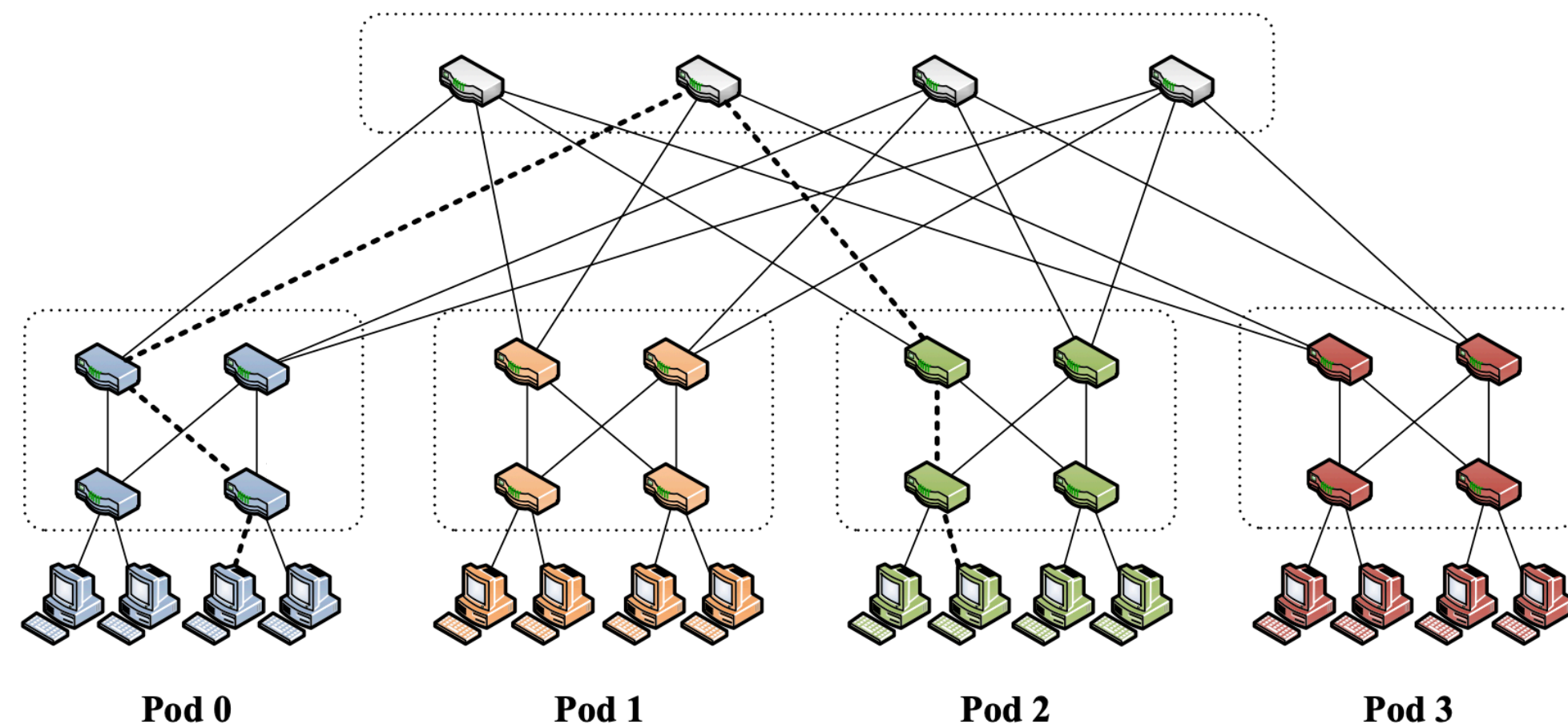
Stateless Routing Table



Proposal #4

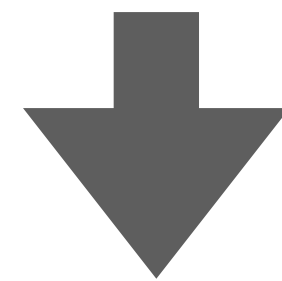
Decouple routing and addressing

Proposal #3 vs. Proposal #4

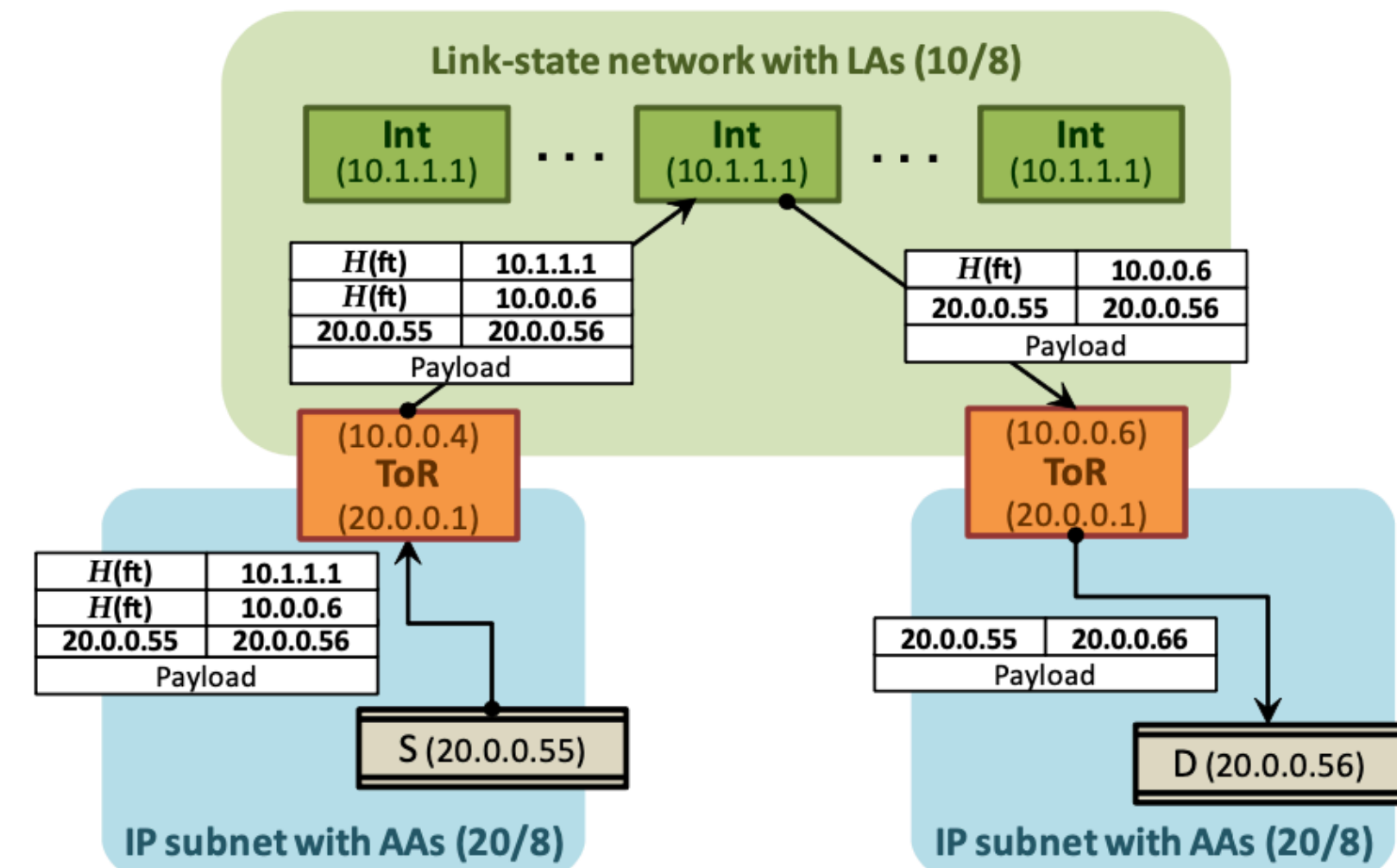


Proposal #3

Co-design routing and addressing

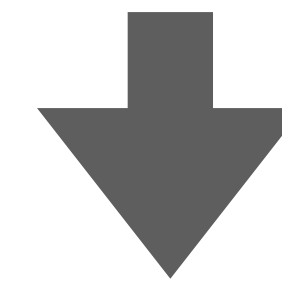


Stateless Routing Table



Proposal #4

Decouple routing and addressing



Stateful Routing Table

Stateless v.s Stateful Routing

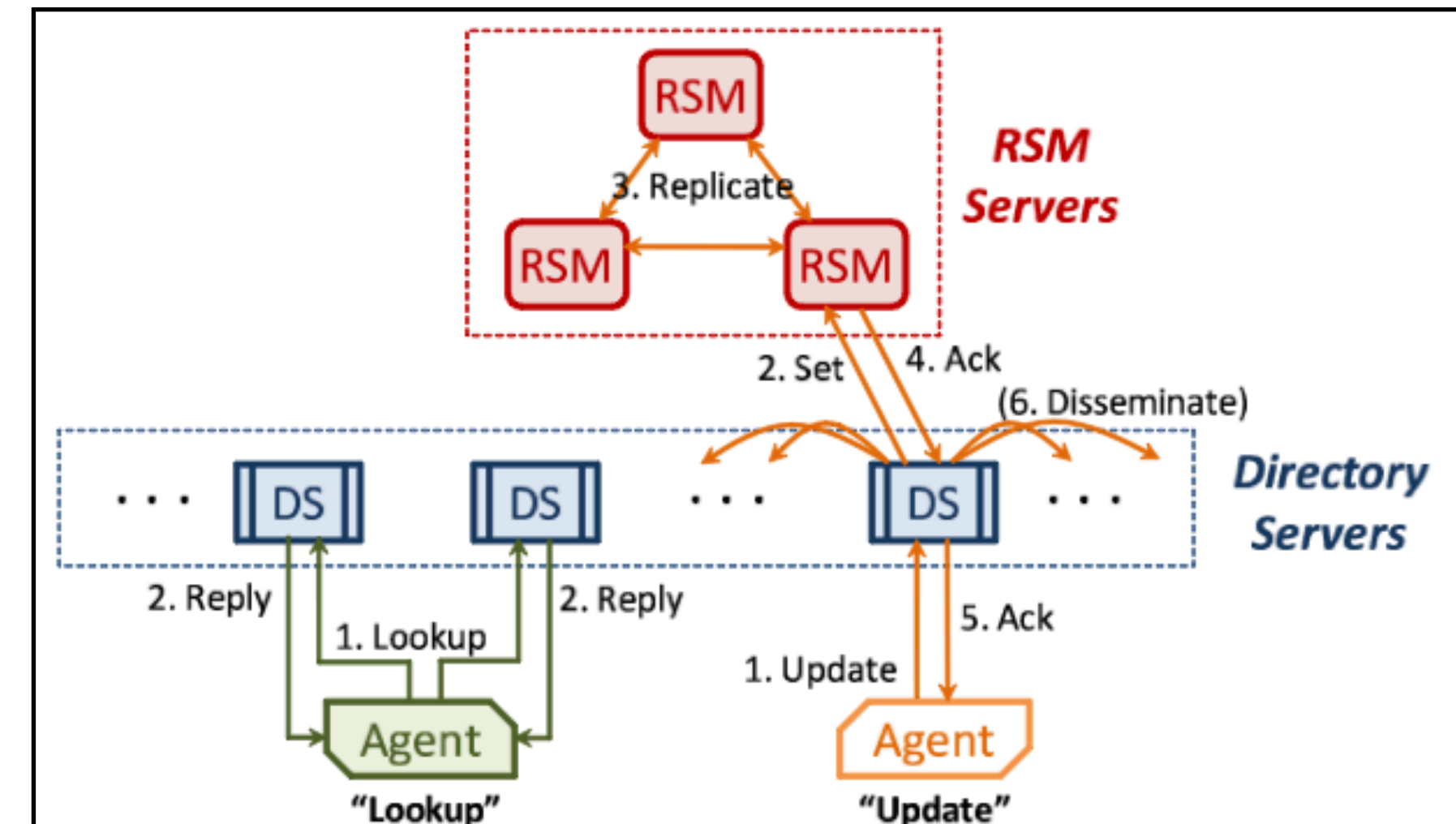
- Stateless Routing

- Easy to manage
- High scalability
- But lacks application awareness

```
1 foreach pod  $x$  in  $[0, k - 1]$  do
2   foreach switch  $z$  in  $[(k/2), k - 1]$  do
3     foreach subnet  $i$  in  $[0, (k/2) - 1]$  do
4       addPrefix( $10.x.z.1$ ,  $10.x.i.0/24$ ,  $i$ );
5     end
6     addPrefix( $10.x.z.1$ ,  $0.0.0.0/0$ ,  $0$ );
7     foreach host ID  $i$  in  $[2, (k/2) + 1]$  do
8       addSuffix( $10.x.z.1$ ,  $0.0.0.i/8$ ,
9          $(i - 2 + z) \bmod (k/2) + (k/2)$ );
10    end
11  end
```

- Stateful Routing

- Maintenance complexity
- Central operational point
- Application adaptability



Obsolete? How do addressing and routing work in data center networks today?

Google Jupiter Evolving: DCN Keeps Expanding!

- Rack-to-rack communication becomes a bottleneck!
 - Compute and storage are updated to satisfy the application demands.
 - But, Clos network's spine is shared.

Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking

Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, Amin Vahdat
Google
sigcomm-jupiter-evolving@google.com

ABSTRACT

We present a decade of evolution and production experience with Jupiter datacenter network fabrics. In this period Jupiter has delivered 5x higher speed and capacity, 30% reduction in capex, 41% reduction in power, incremental deployment and technology refresh all while serving live production traffic. A key enabler for these improvements is *evolving Jupiter from a Clos to a direct-connect topology among the machine aggregation blocks*. Critical architectural changes for this include: A datacenter interconnection layer employing Micro-Electro-Mechanical Systems (MEMS) based Optical Circuit Switches (OCSes) to enable dynamic topology reconfiguration, central-

KEYWORDS

Datacenter network, Software-defined networking, Traffic engineering, Topology engineering, Optical circuit switches.

ACM Reference Format:

Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, Amin Vahdat Google sigcomm-jupiter-evolving@google.com . 2022. Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Network-

Google Jupiter Evolving: DCN Keeps Expanding!

- Rack-to-rack communication becomes a bottleneck!
 - Compute and storage are updated to satisfy the application demands.
 - But, Clos network's spine is shared.

Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Networking

Leon Poutievski, Omid Mashayekhi, Joon Ong, Arjun Singh, Mukarram Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura,

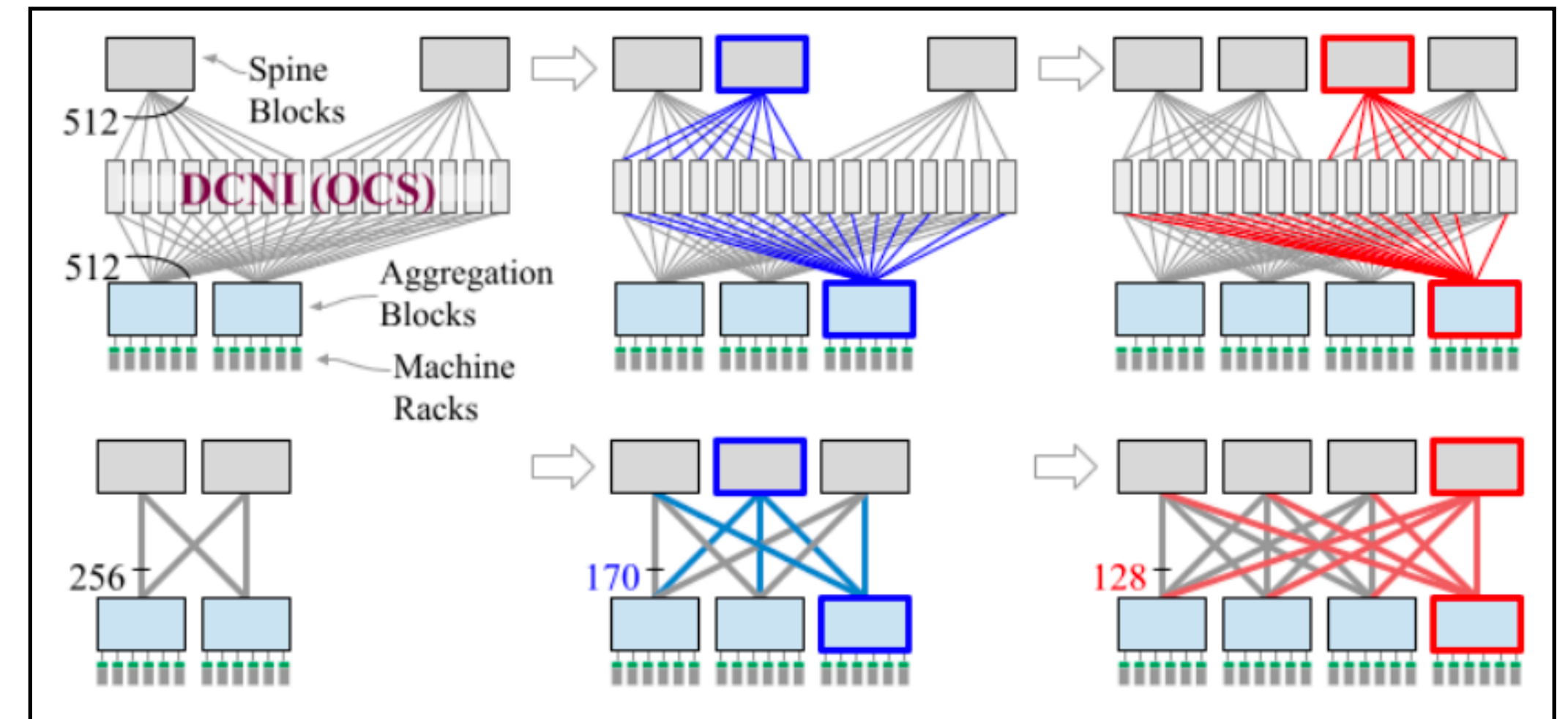
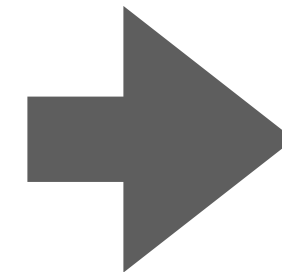
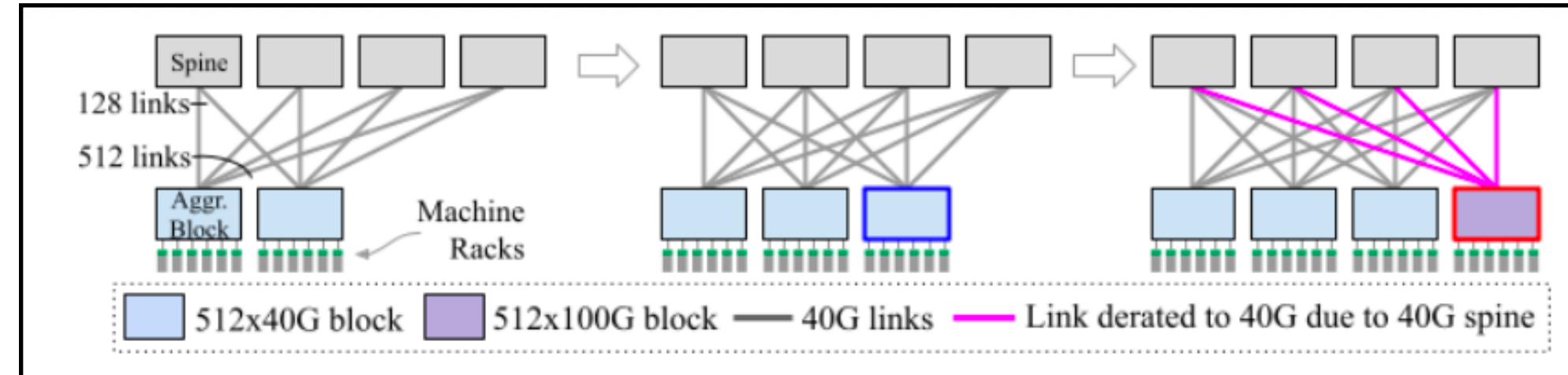
How can we upgrade the network without rebuilding the entire data center?

key enabler for these improvements is *evolving Jupiter from a Clos to a direct-connect topology among the machine aggregation blocks*. Critical architectural changes for this include: A datacenter interconnection layer employing Micro-Electro-Mechanical Systems (MEMS) based Optical Circuit Switches (OCSes) to enable dynamic topology reconfiguration, central-

lion Tariq, Rui Wang, Jianan Zhang, Virginia Beauregard, Patrick Conner, Steve Gribble, Rishi Kapoor, Stephen Kratzer, Nanfang Li, Hong Liu, Karthik Nagaraj, Jason Ornstein, Samir Sawhney, Ryohei Urata, Lorenzo Vicisano, Kevin Yasumura, Shidong Zhang, Junlan Zhou, Amin Vahdat Google sigcomm-jupiter-evolving@google.com . 2022. Jupiter Evolving: Transforming Google's Datacenter Network via Optical Circuit Switches and Software-Defined Network-

Key Idea: Remove the Spine

- Data center network interconnect (DCNI) based on OCS
 - Use the MEMS Optical Circuit Switch
 - The physical fabric is fixed. But logical topology becomes programmable!



Addressing and Routing: SDN

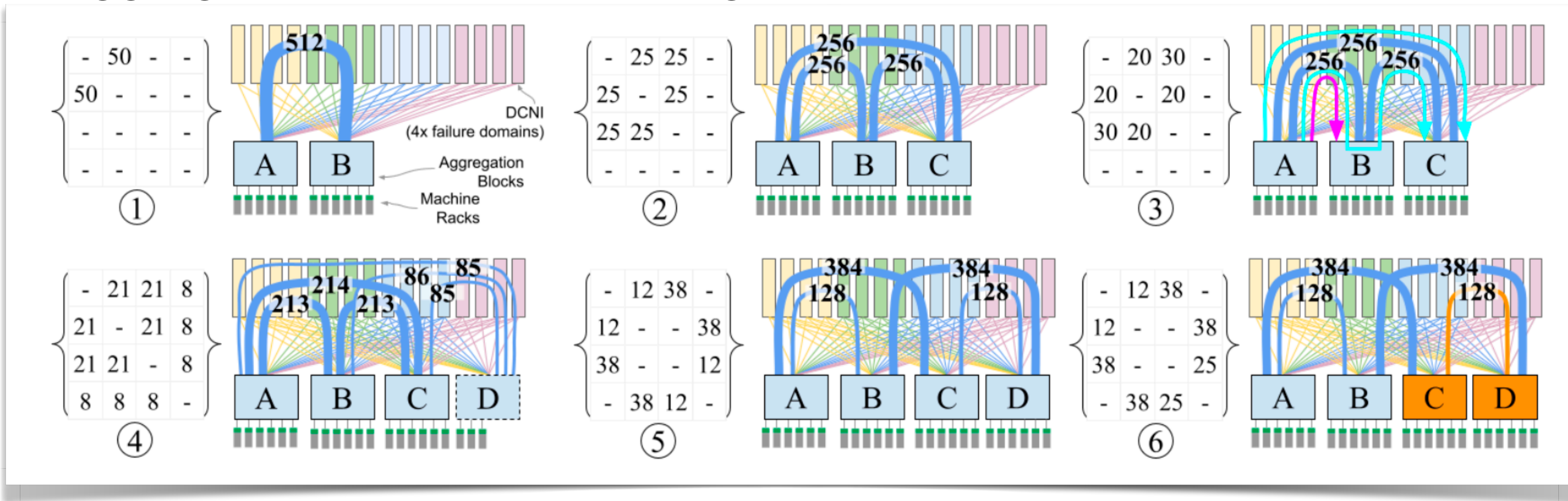
- Addressing
 - Servers retain conventional IP addresses; OCS is transparent at L1.
 - Aggregation blocks are routing domains.
 - Destination IP addresses identify the endpoint.

Addressing and Routing: SDN

- Addressing
 - Servers retain conventional IP addresses; OCS is transparent at L1.
 - Aggregation blocks are routing domains.
 - Destination IP addresses identify the endpoint.
- Routing
 - Orion (SDN controller) computes inter-OCS block forwarding paths.
 - Rely on traffic engineering (TE) to adjust path weights
 - Employ topology engineering (ToE) to reprogram OCS circuits

Addressing and Routing: SDN

- Addressing
 - Servers retain conventional IP addresses; OCS is transparent at L1.
 - Aggregation blocks are routing domains.



Beyond FatTree: Direct-Connected Topology

- Originally proposed in the HPC and supercomputer
 - Use low-radix adapters instead of high-radix switches
 - E.g., Cray XD1, XT3, etc..
- Apply for data center networking, especially rack-scale
 - E.g., BCube (Sigcomm'09), R2C2 (SIGCOMM'15)

R2C2: A Network Stack for Rack-scale Computers

Paolo Costa Hitesh Ballani Kaveh Razavi* Ian Kash

Microsoft Research

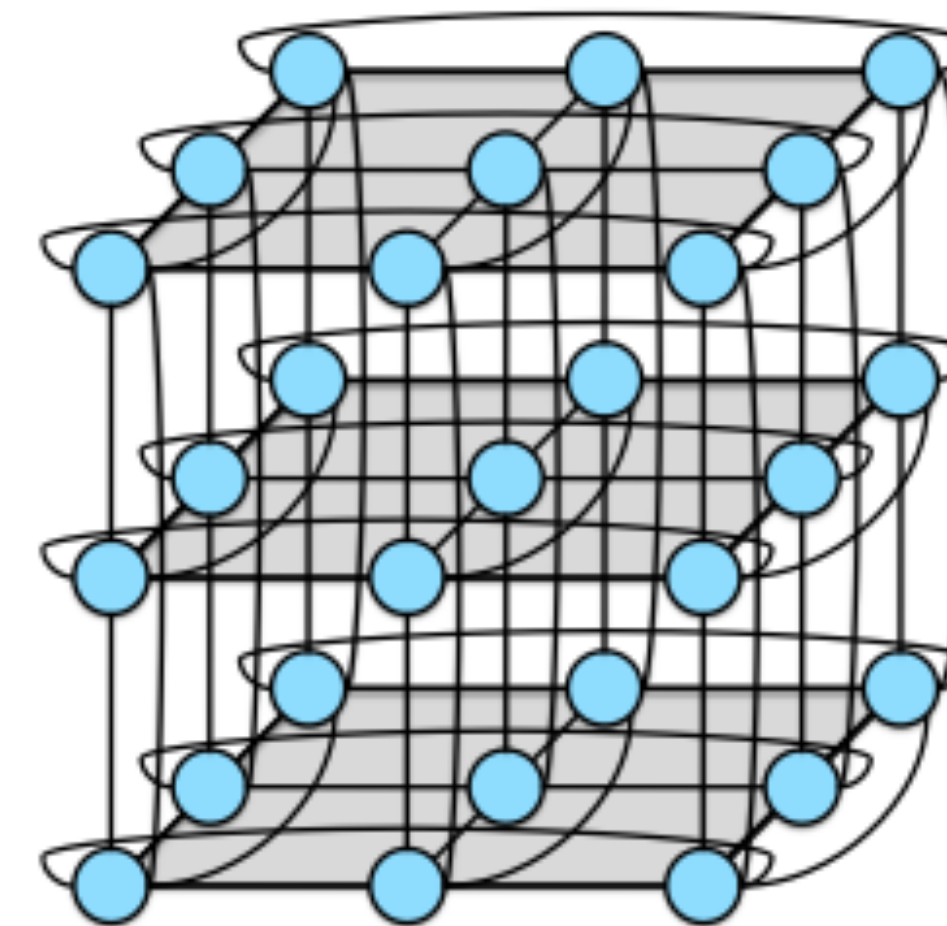
ABSTRACT

Rack-scale computers, comprising a large number of micro-servers connected by a direct-connect topology, are expected to replace servers as the building block in data centers. We focus on the problem of routing and congestion control across the rack's network, and find that high path diversity in rack topologies, in combination with workload diversity across it, means that traditional solutions are inadequate.

We introduce R2C2, a network stack for rack-scale computers that provides flexible and efficient routing and congestion control. R2C2 leverages the fact that the scale of rack topologies allows for low-overhead broadcasting to en-

1. INTRODUCTION

While today's large-scale data centers such as those run by Amazon, Google, and Microsoft are built using commodity off-the-shelf servers, recently there has been an increasing trend towards server customization to reduce costs and improve performance [50, 54, 55, 58]. One such trend is the advent of "rack-scale computing". We use this term to refer to emerging architectures that propose servers or *rack-scale computers* comprising a large number of tightly integrated systems-on-chip, interconnected by a network fabric. This design enables thousands of cores per rack and provides high bandwidth for rack-scale applications. The con-



Beyond FatTree: Direct-Connected Topology

- Originally proposed in the HPC and supercomputer
 - Use low-radix adapters instead of high-radix switches
 - E.g., Cray XD1, XT3, etc..

Very popular in today's AI Supercomputer

Paolo Costa Hitesh Ballani Kaveh Razavi* Ian Kash
Microsoft Research

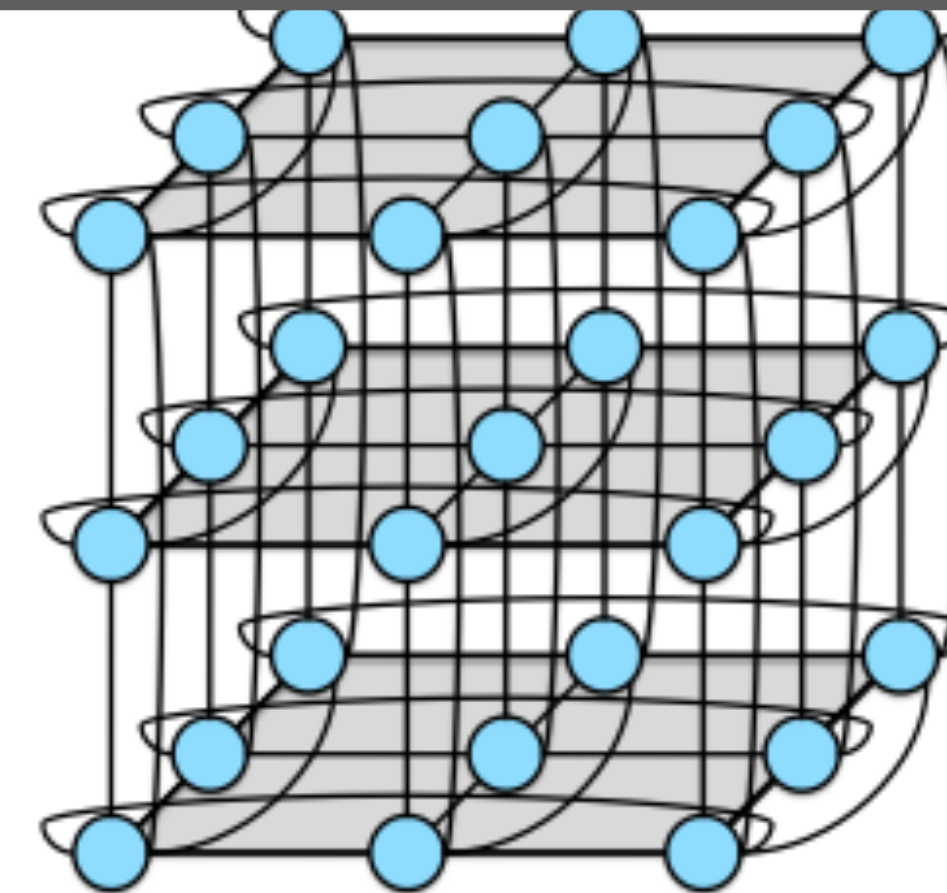
ABSTRACT

Rack-scale computers, comprising a large number of micro-servers connected by a direct-connect topology, are expected to replace servers as the building block in data centers. We focus on the problem of routing and congestion control across the rack's network, and find that high path diversity in rack topologies, in combination with workload diversity across it, means that traditional solutions are inadequate.

We introduce R2C2, a network stack for rack-scale computers that provides flexible and efficient routing and congestion control. R2C2 leverages the fact that the scale of rack topologies allows for low-overhead broadcasting to en-

1. INTRODUCTION

While today's large-scale data centers such as those run by Amazon, Google, and Microsoft are built using commodity off-the-shelf servers, recently there has been an increasing trend towards server customization to reduce costs and improve performance [50, 54, 55, 58]. One such trend is the advent of "rack-scale computing". We use this term to refer to emerging architectures that propose servers or *rack-scale computers* comprising a large number of tightly integrated systems-on-chip, interconnected by a network fabric. This design enables thousands of cores per rack and provides high bandwidth for rack-scale applications. The con-



Beyond FatTree: Direct-Connected Topology

- Originally proposed in the HPC and supercomputer

NVIDIA DGX SuperPOD



gestion control. R2C2 leverages the fact that the scale of rack topologies allows for low-overhead broadcasting to en-

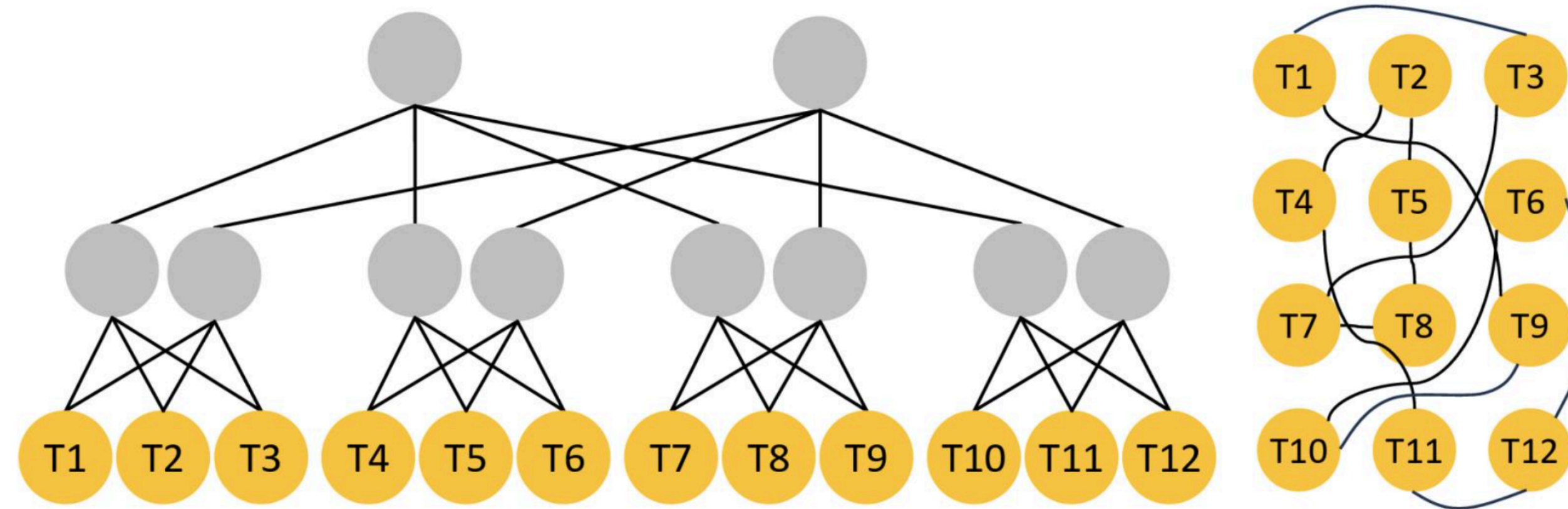
This design enables thousands of cores per rack and provides high bandwidth for rack-scale applications. The con-

Beyond FatTree: Resilient Network Graphs

- Conventional DCN
 - Regular hierarchy w/ scalable communication paths provisioned
 - But many switches/links exist primarily to offer hierarchy and redundancy!
- AWS Resilient Network Graphs (Released May 2026)
 - Apply quasi-random graph theory to construct a flatter network
 - Provide high connectivity and resilience with much fewer elements

How Resilient Network Graphs Works

- Topology: Quasi-random Graph
 - Remove the leaf-spine-core hierarchy
 - Switches connect directly to a small set of other switches
 - Quasi-random connectivity creates many diverse paths w/o spine



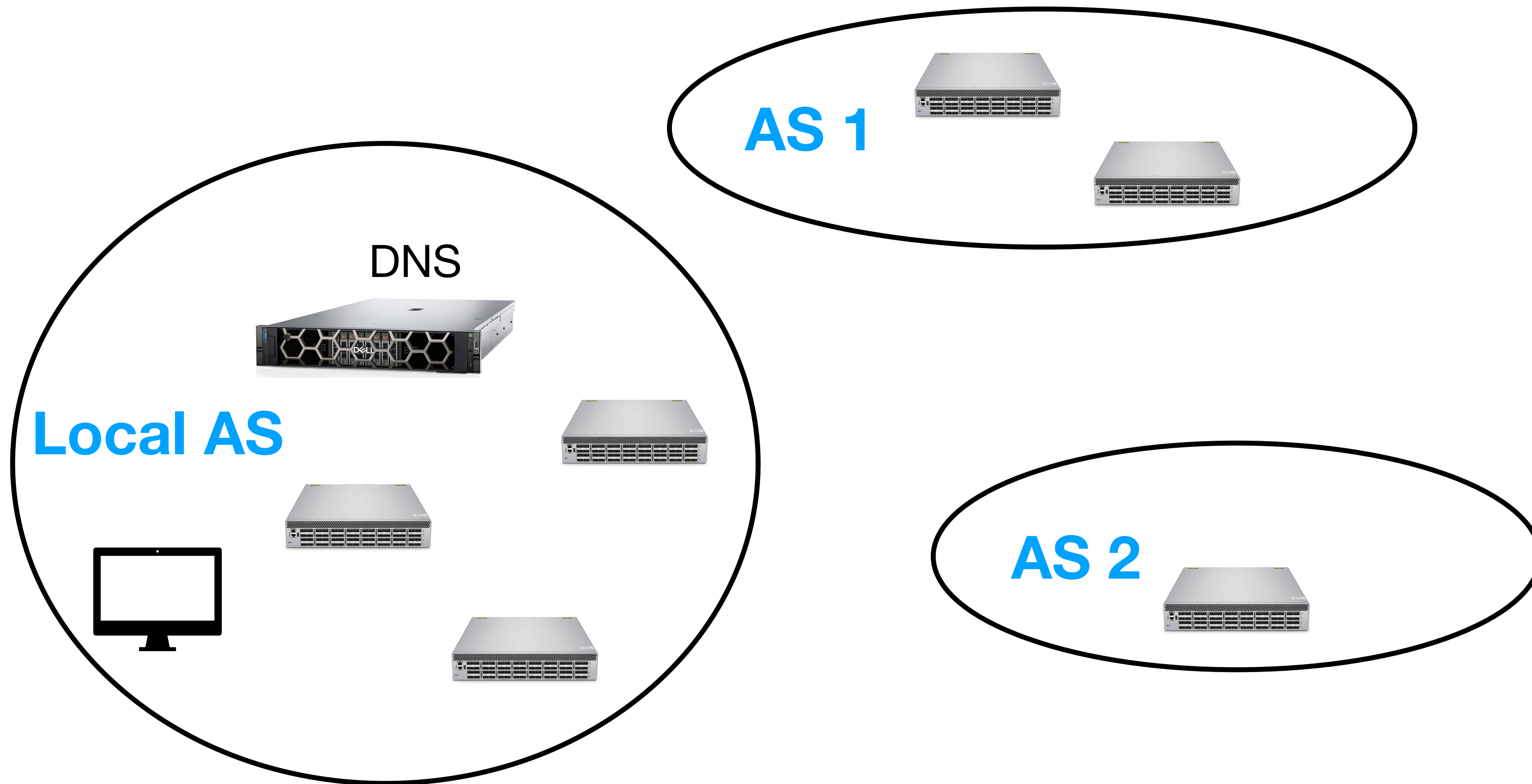
How Resilient Network Graphs Works (cont'd)

- Cabling: ShuffleBox
 - Purely random cabling would be unmanageable.
 - ShuffleBox uses a fixed internal fiber permutation.
 - Simple structured inter-ShuffleBox cabling produces the desired paths.
- Routing: Spraypoint
 - Spray: source distributes traffic randomly across its neighbors
 - Route: from there, use conventional shortest-path routing
 - Converge: destination waypoint “rings” progressively guide packets

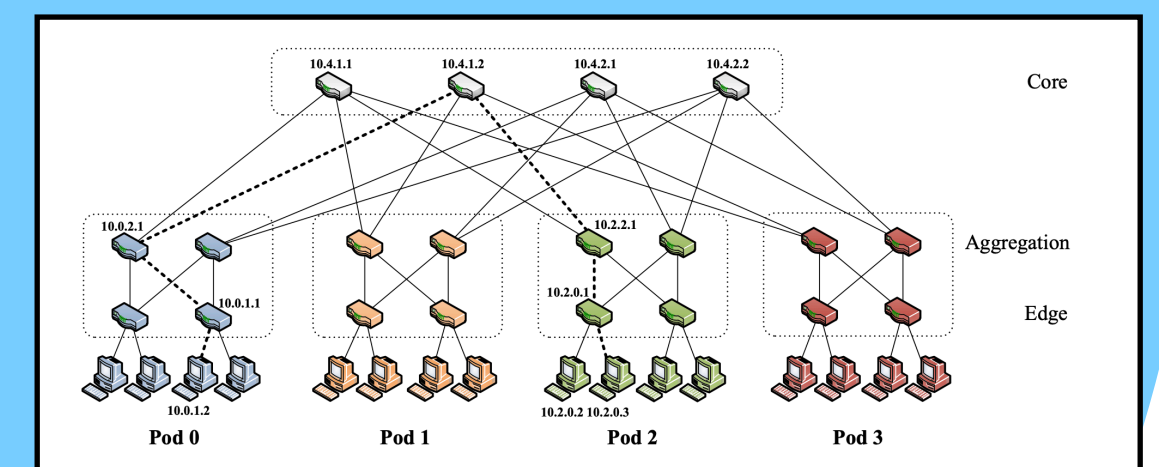
Recap: Addressing and Routing in Data Center Networks

- Type #1: User $\longleftrightarrow$ Data Center Gateway
 - Internet Routing
- Type #2: Data Center Gateway $\longleftrightarrow$ Data Center Server
 - Locate the destination address quickly at scale
- Type #3: Data Center Server $\longleftrightarrow$ Data Center Server
 - We discussed four working proposals
 - Stateless and stateful routing: no one-size-fit-all solution

Type #2: Data Center Gateway <—> Data Center Server

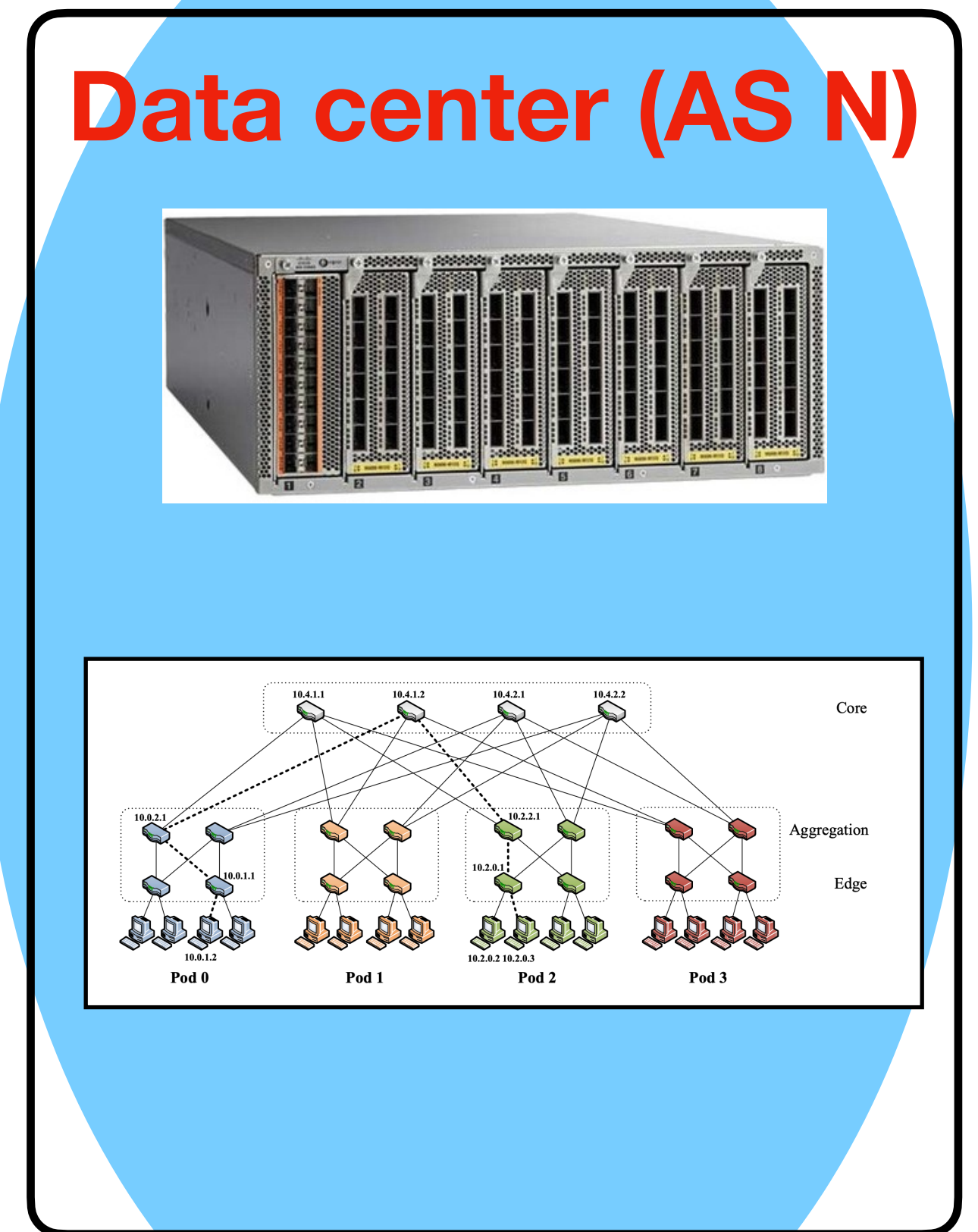
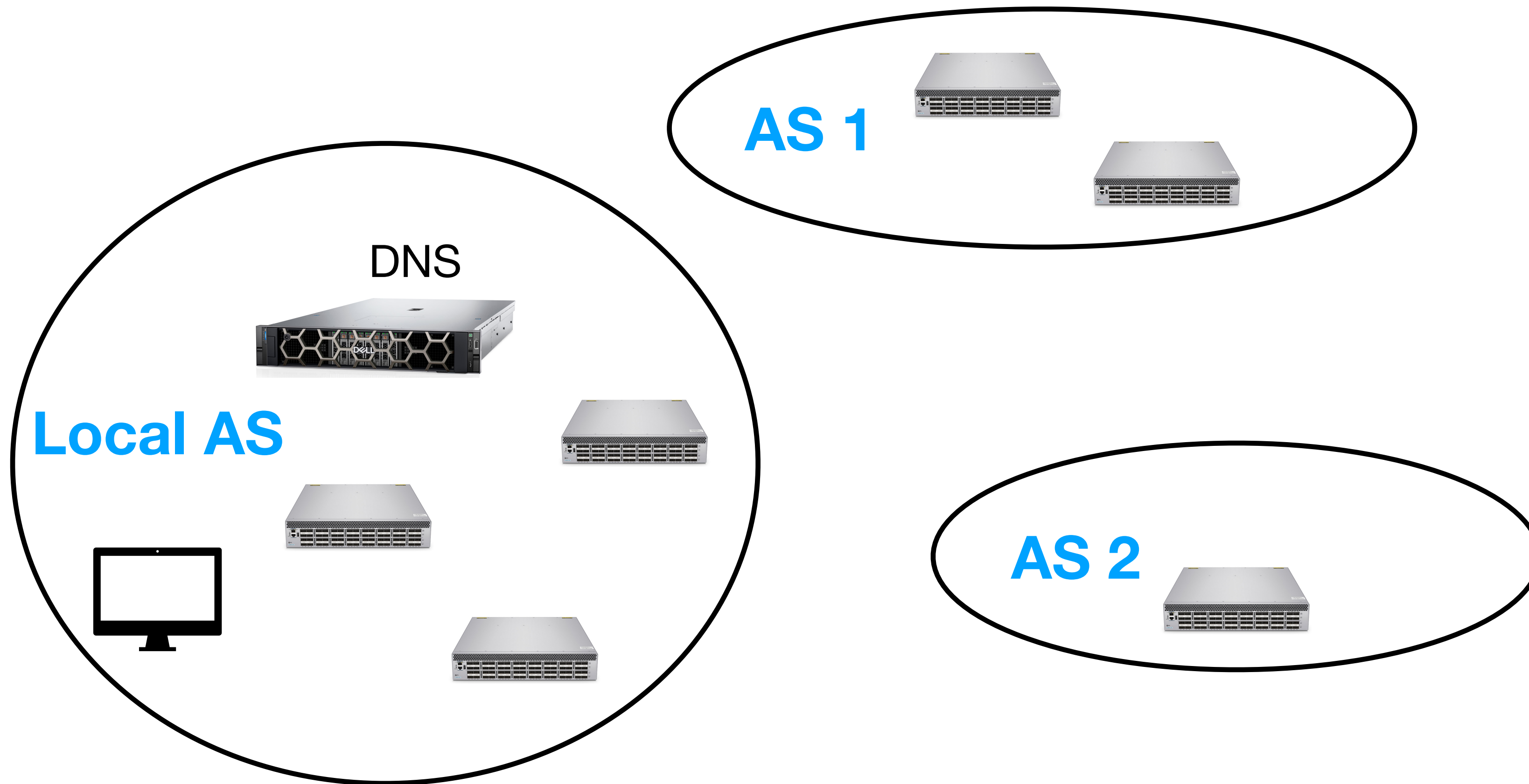


Data center (AS N)



Type #2: Data Center Gateway <—> Data Center Server

- Key: update the service destination address



Type #2: Data Center Gateway <—> Data Center Server

- Key: update the service destination address



Summary

- Today
 - Addressing and routing in data center networks (II)
- Next two lectures
 - Flow scheduling
 - Hedera (NSDI'10)