

Advanced Computer Networks

Gateway in Data Center Networks (II)

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

Ming Liu
mgliu@cs.wisc.edu

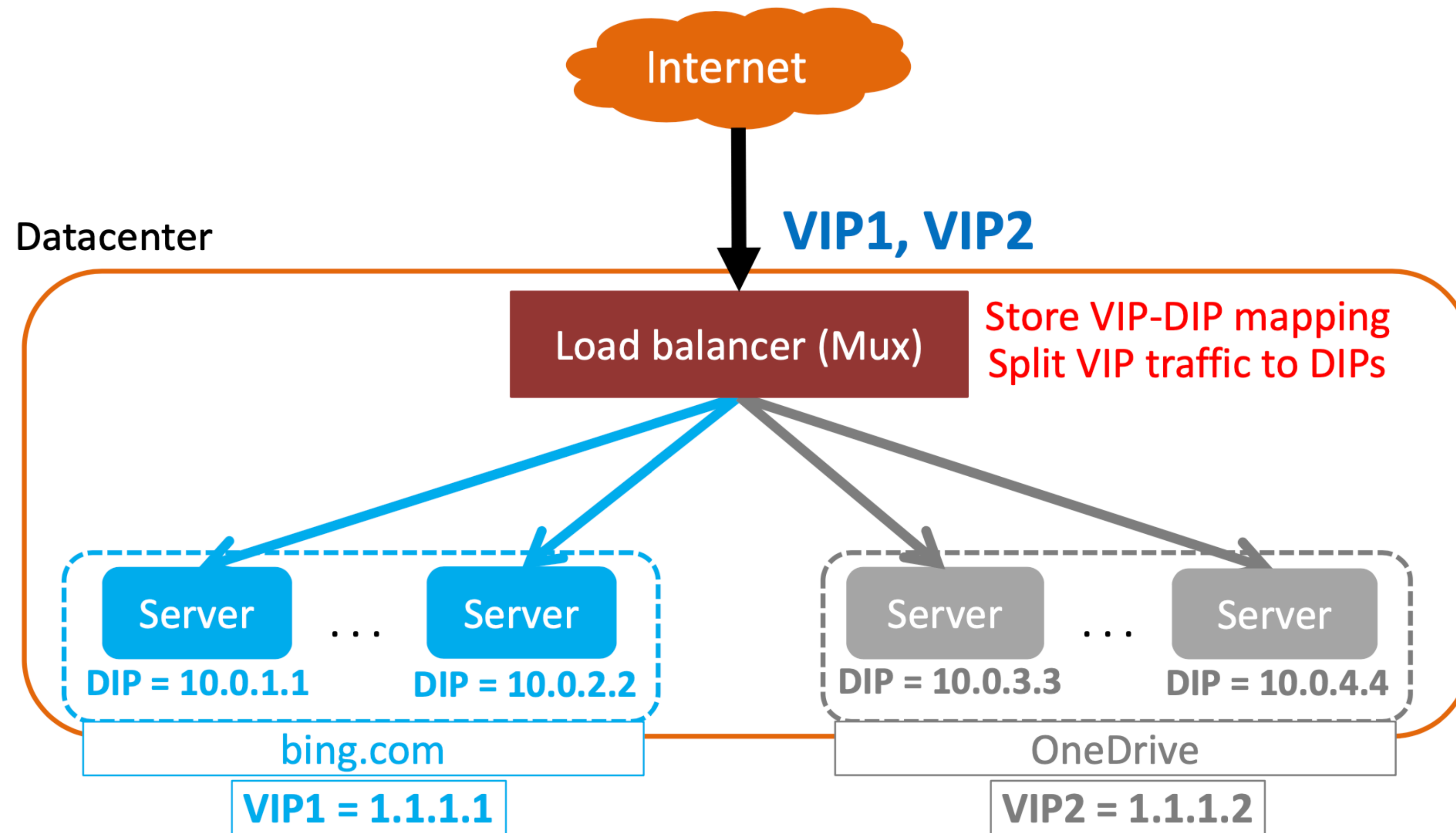
Outline

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

Design requirements of a DCN Gateway

- **Multiple NIC interfaces**
- **Fast memory**
- **Line-rate processing**
- **Maintain a load-balancing table**
- **Perform hashing and lookup**
- **Maintain a connection tracking table**

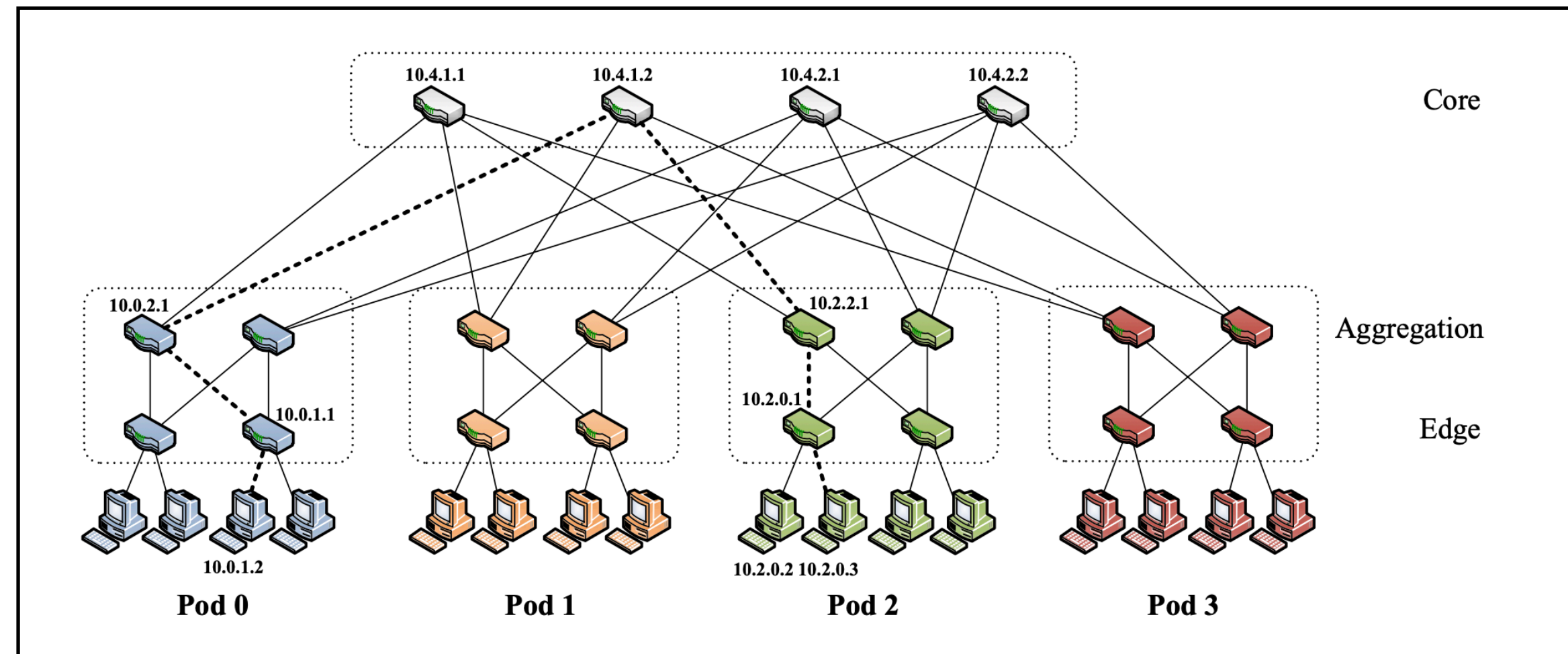
Microsoft Azure Example



DCN Gateway Overview

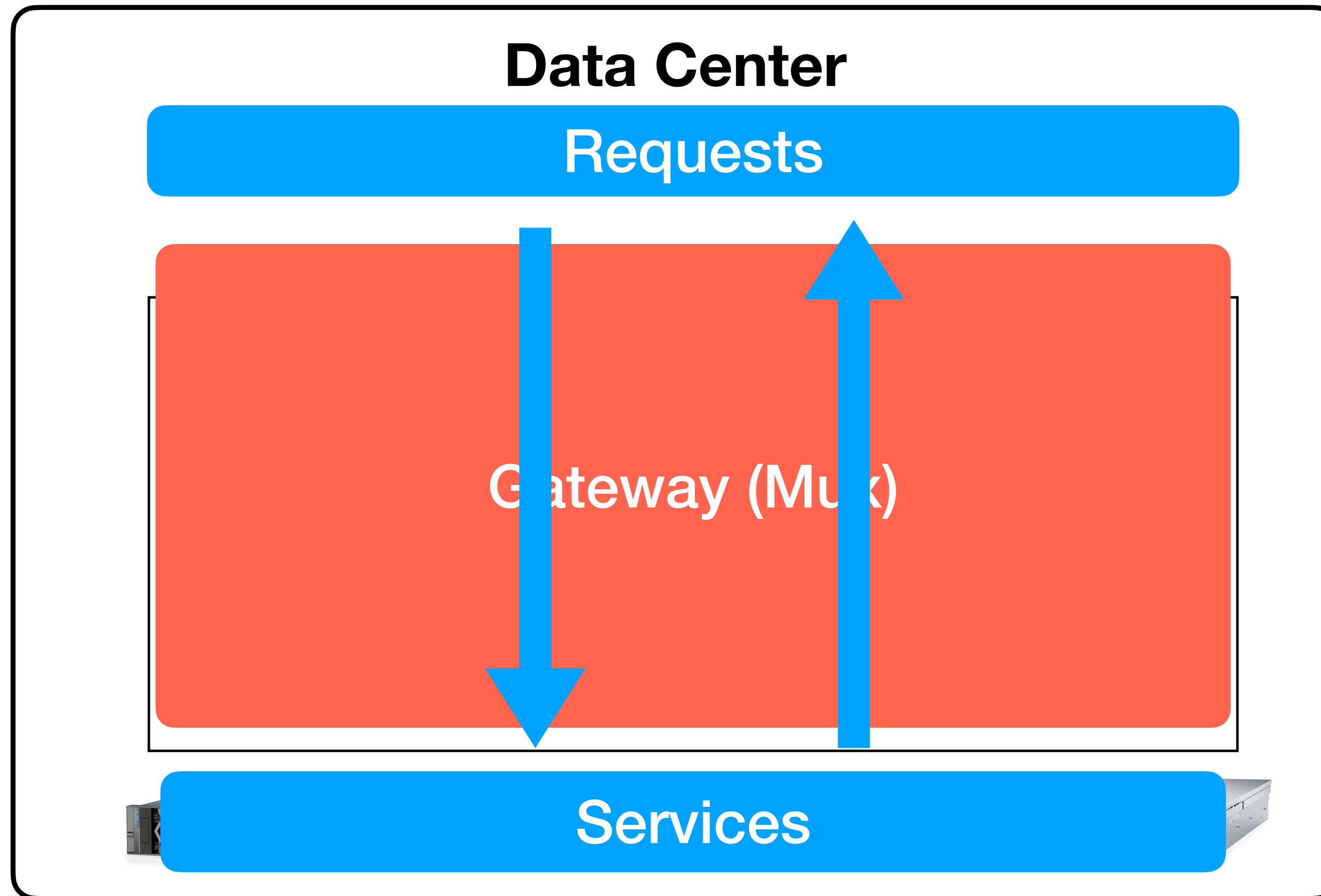
- Like what we discussed in the last lecture

Data Center



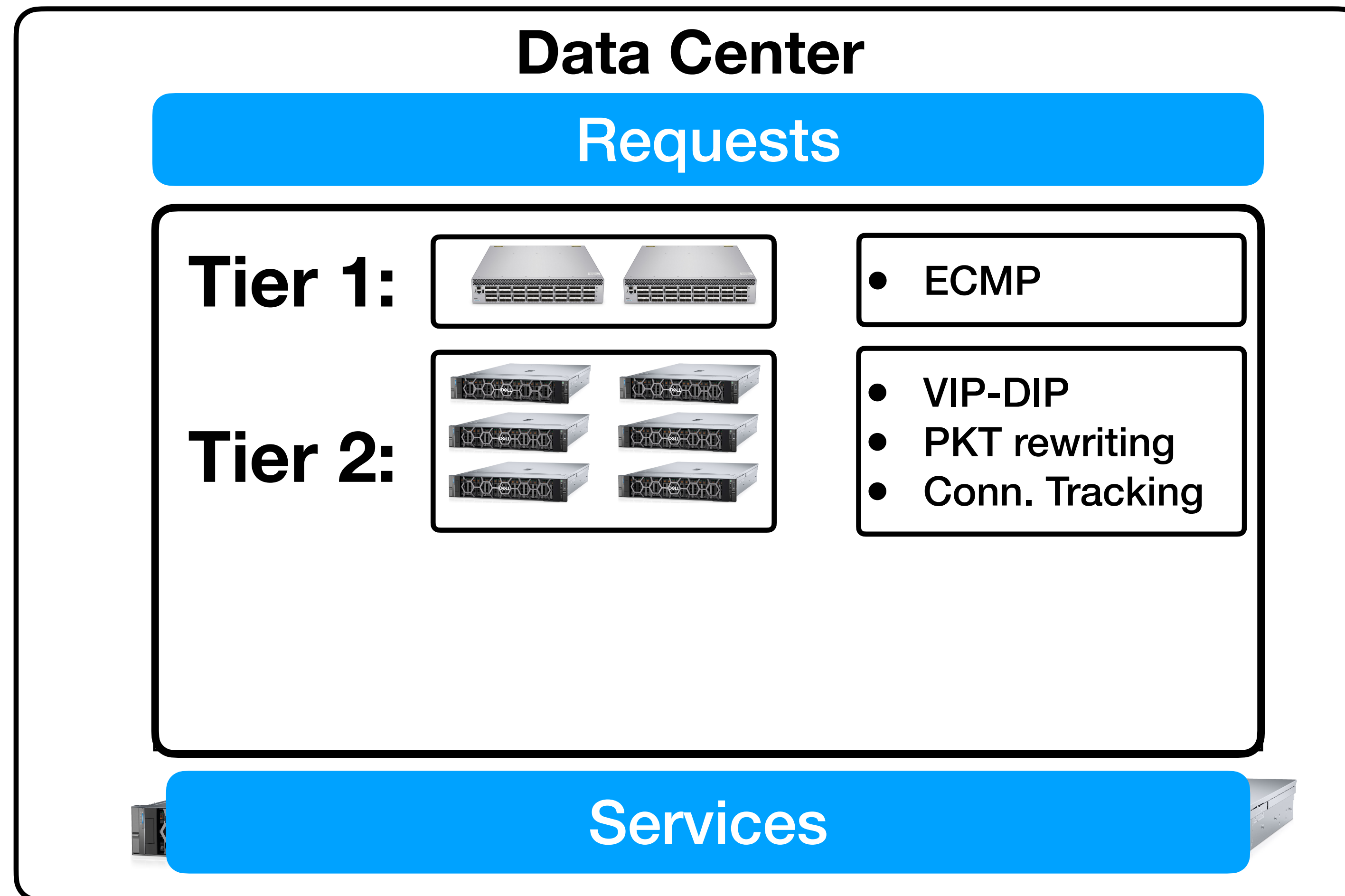
DCN Gateway Overview

- Like what we discussed in the last lecture



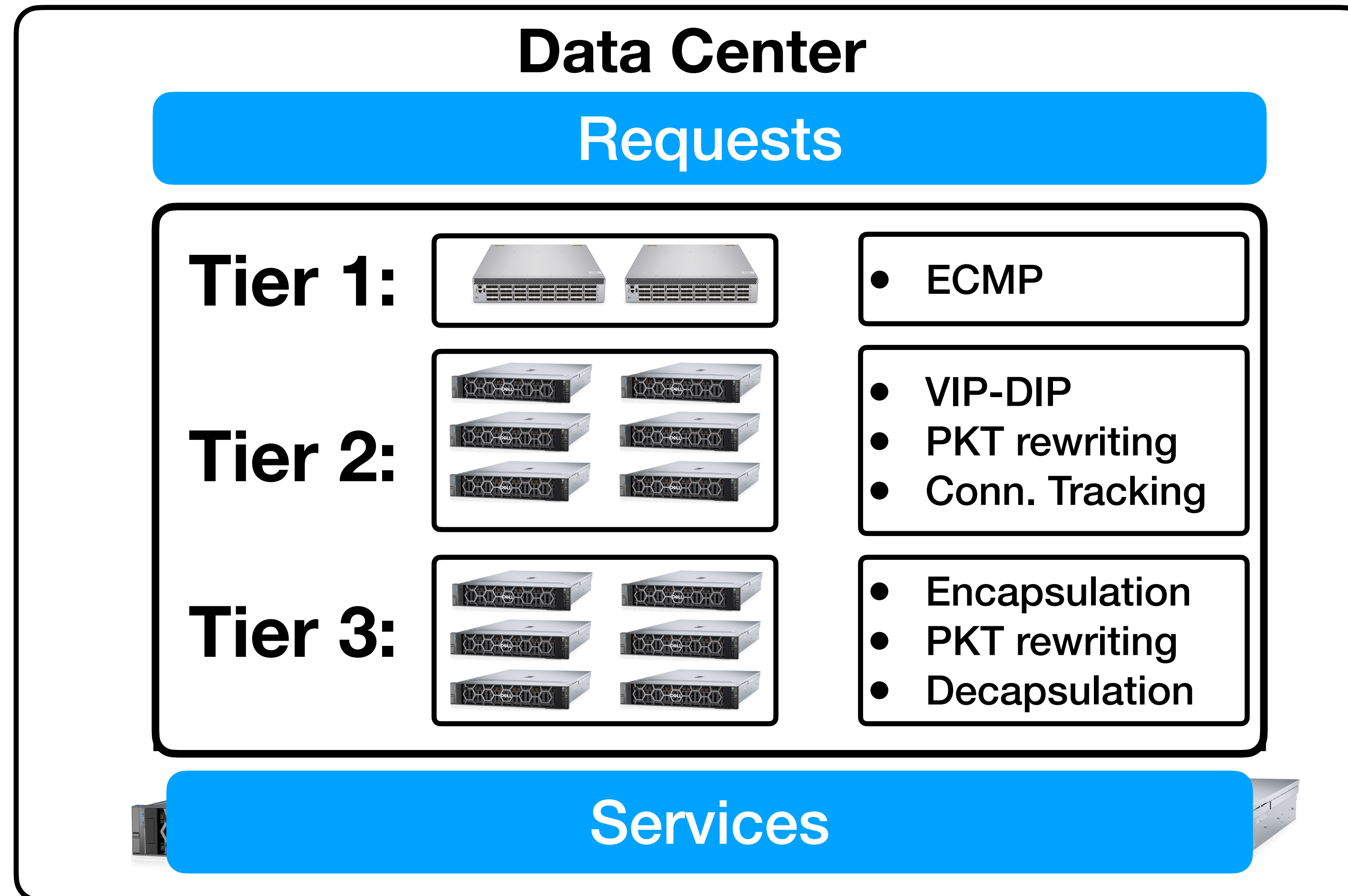
System Architecture of a SW Gateway

- Distributed systems



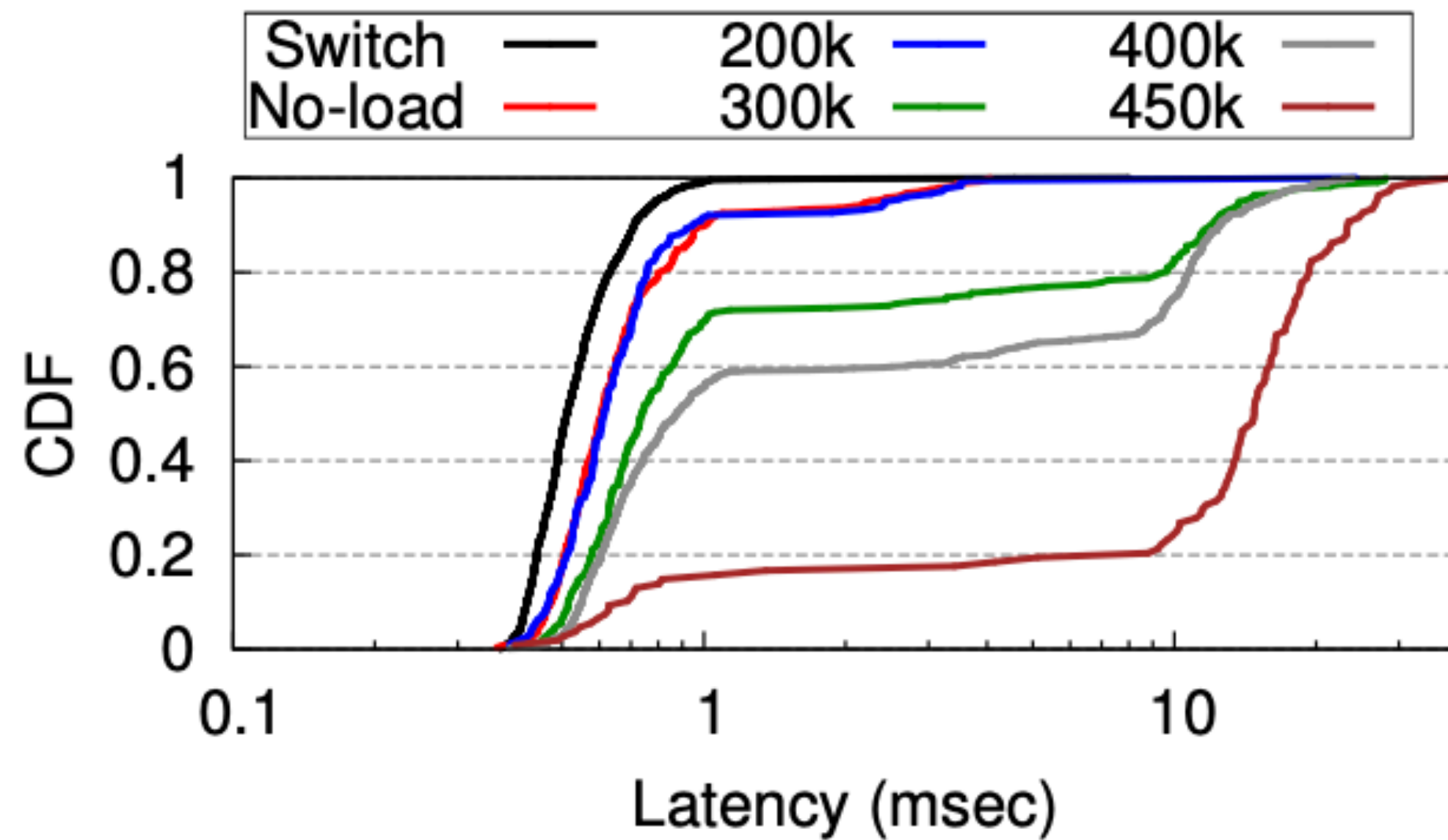
System Architecture of a SW Gateway

- Distributed systems
- Tunneling adds more complexity



What are the limitations of a software gateway?

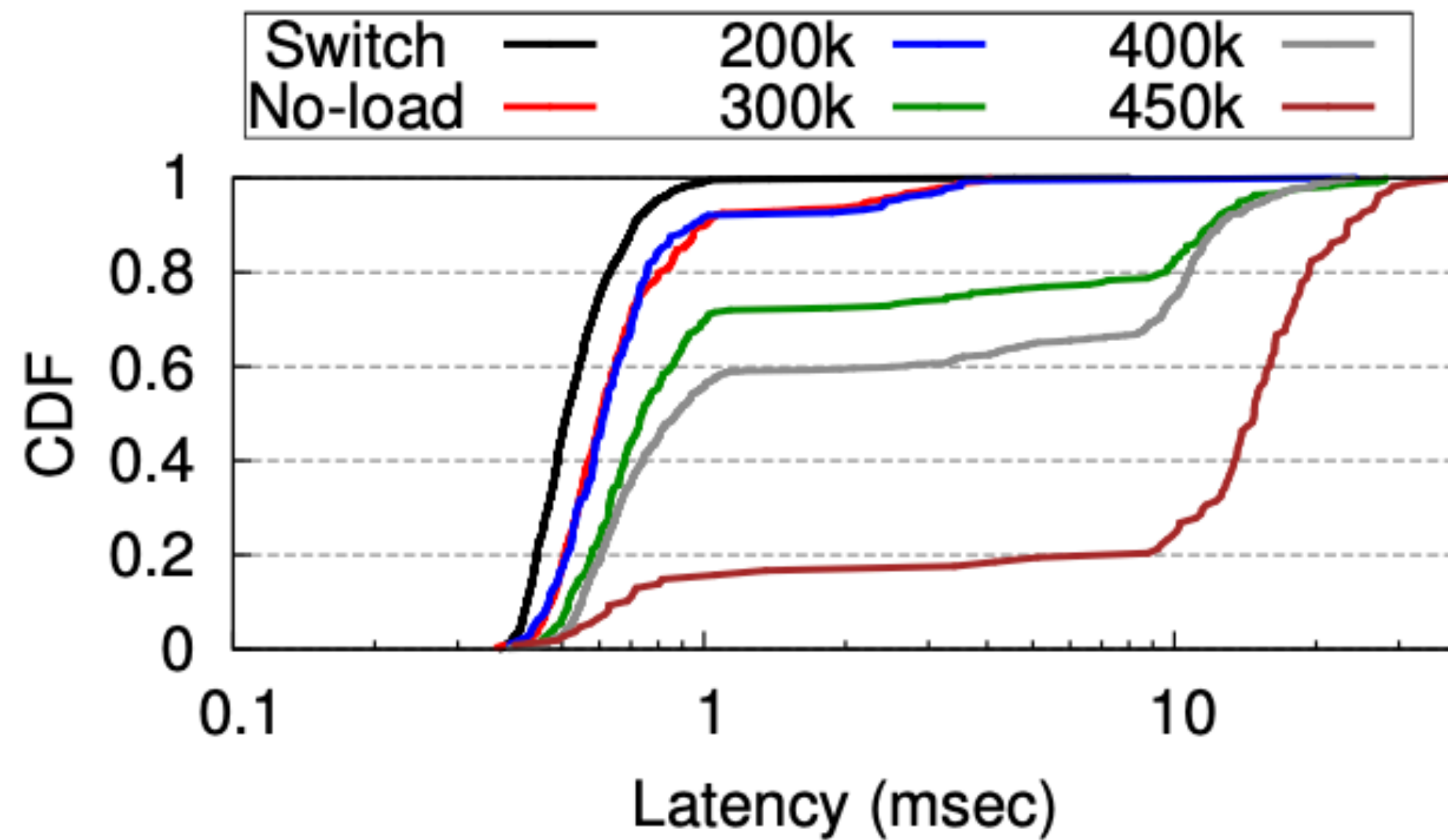
Limitations of the Software Gateway



#1: Slow

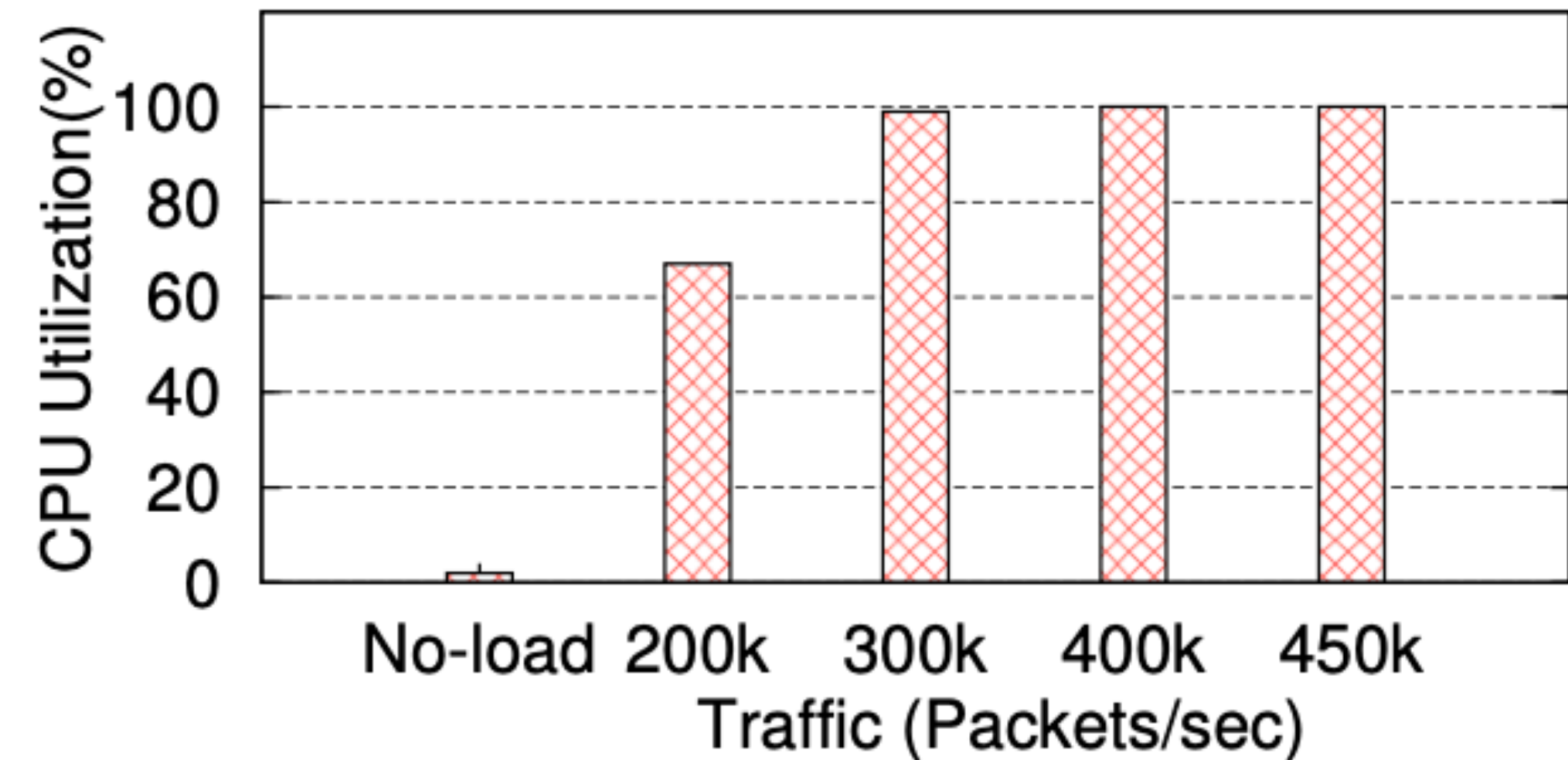
- SMux adds a media latency of 196us
- High latency variance

Limitations of the Software Gateway



#1: Slow

- SMux adds a media latency of 196us
- High latency variance



#2: Costly

- Up to 300K pkts/s per server => 3.6Gbps
- 15Tbps => 5K servers

How does Duet solve it?

How does Duet solve it?
Hardware-software co-design

How does Duet solve it?
Hardware-software co-design

But, why can we co-design the hardware and software?

Key Observation: Commodity switches transit to a gray box. They provide APIs for fine-grained control over ECMP and tunneling functionality.

Key ideas:

- **#1: Use commodity switches as hardware Muxes**
- **#2: Use software Muxes as a backstop**

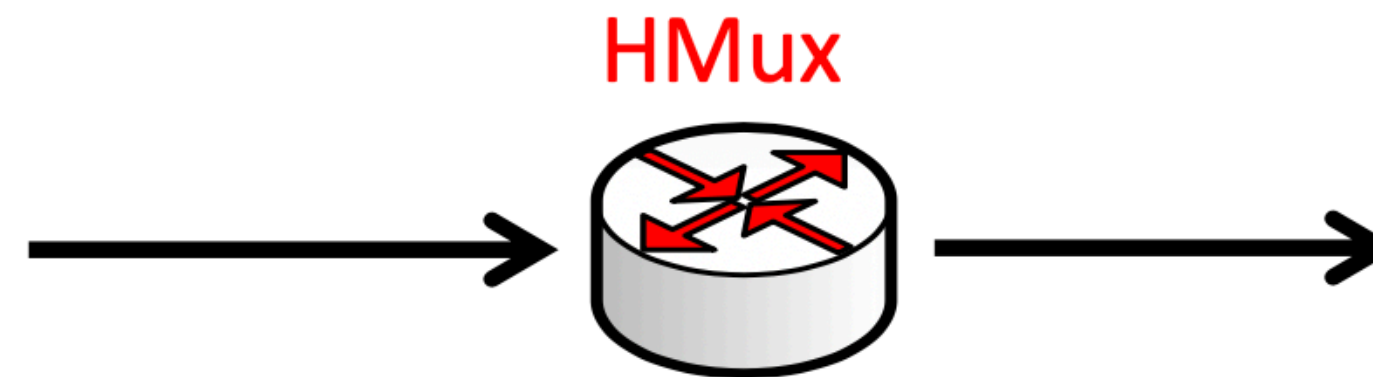
HW Mux Design

- Technique #1: chained table execution

HW Mux Design

- Technique #1: chained table execution

VIP	DIPs
1.1.1.1/32	10.0.0.1 10.0.0.2
1.1.1.2/32	20.0.0.1 20.0.0.2

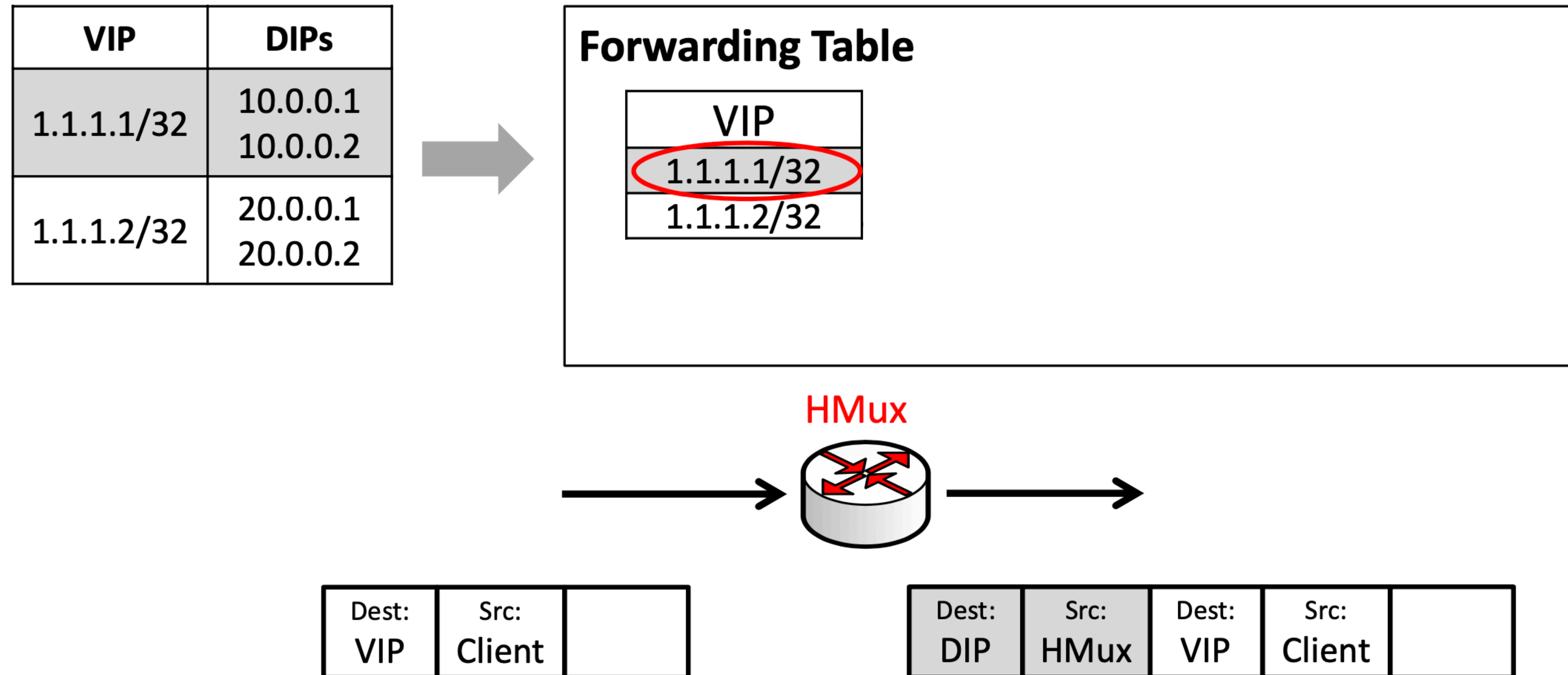


Dest: VIP	Src: Client	
--------------	----------------	--

Dest: DIP	Src: H_mux	Dest: VIP	Src: Client	
--------------	---------------	--------------	----------------	--

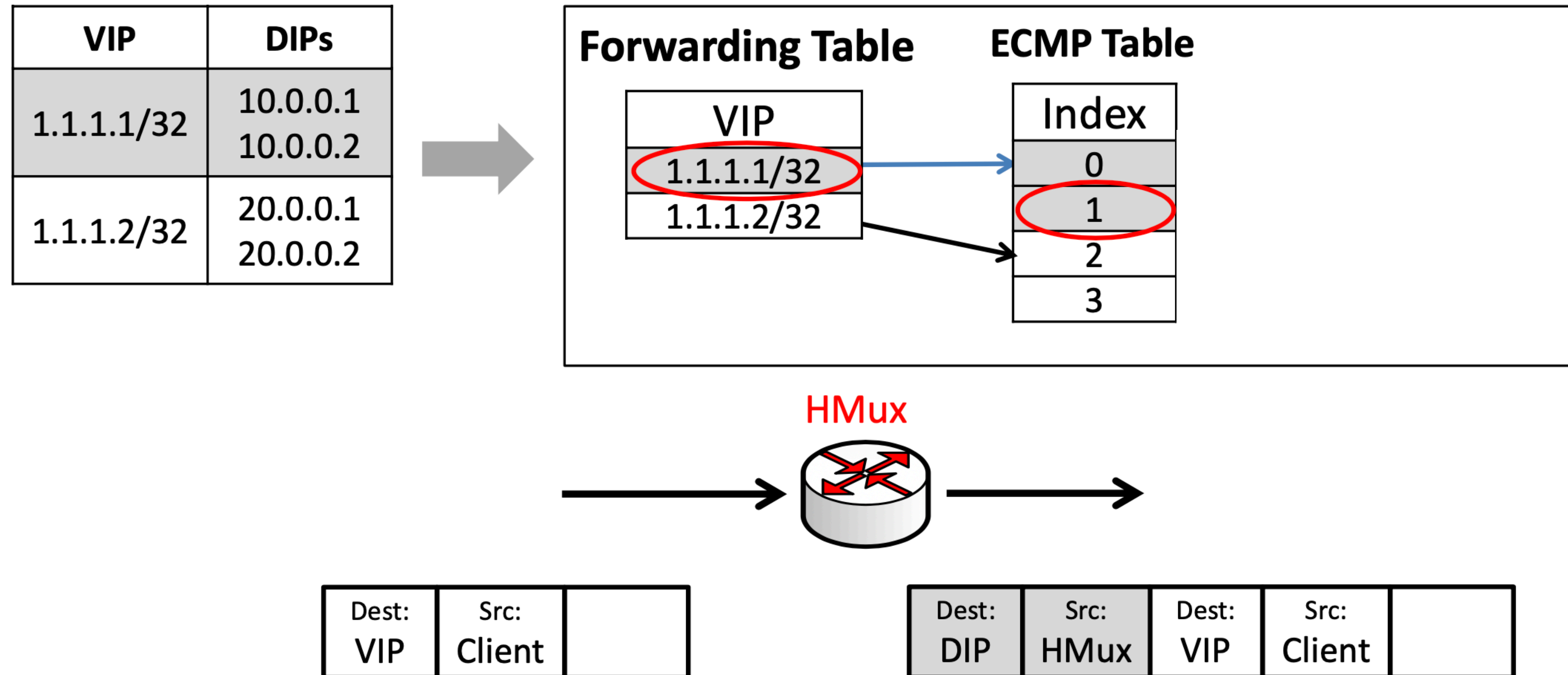
HW Mux Design

- Technique #1: chained table execution



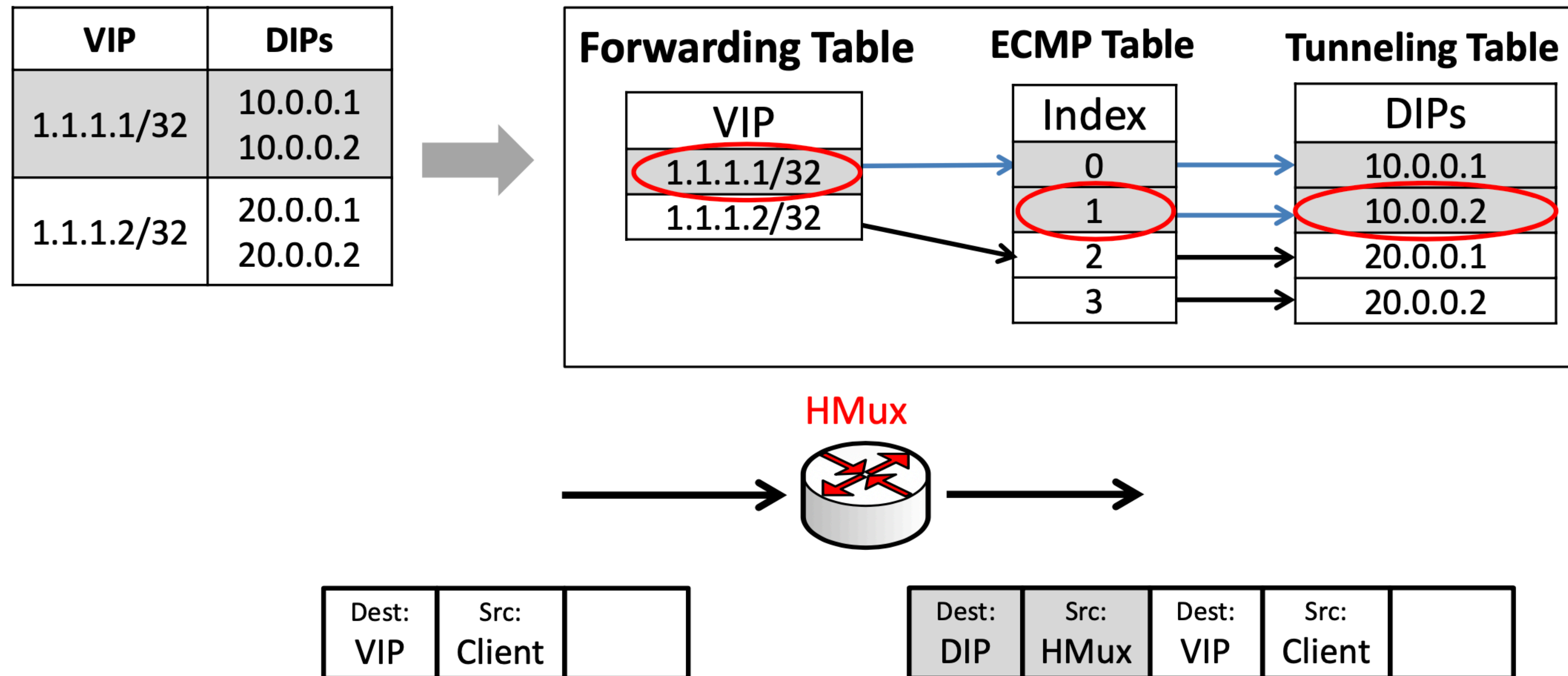
HW Mux Design

- Technique #1: chained table execution



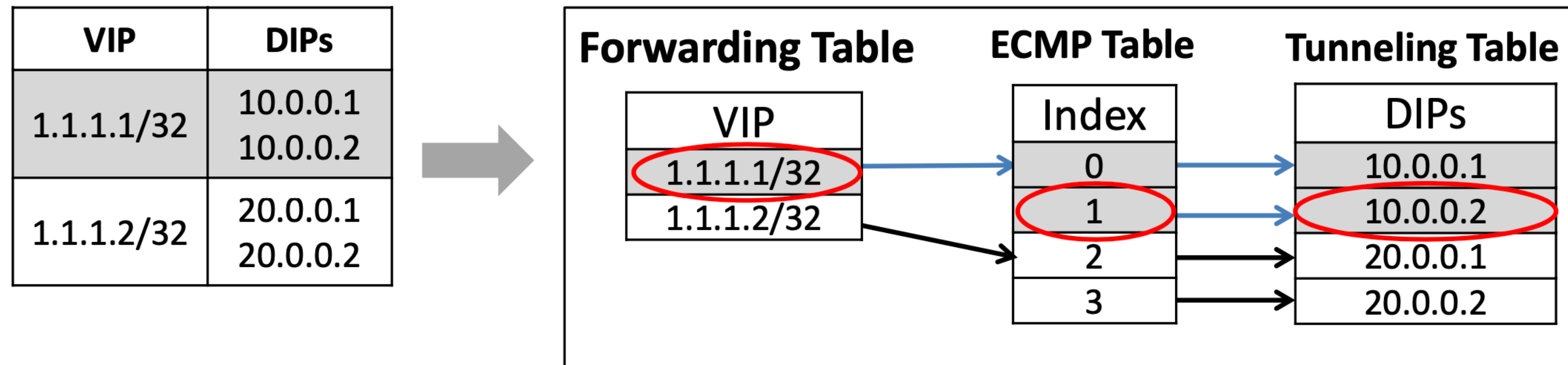
HW Mux Design

- Technique #1: chained table execution



HW Mux Design

- Technique #1: chained table execution



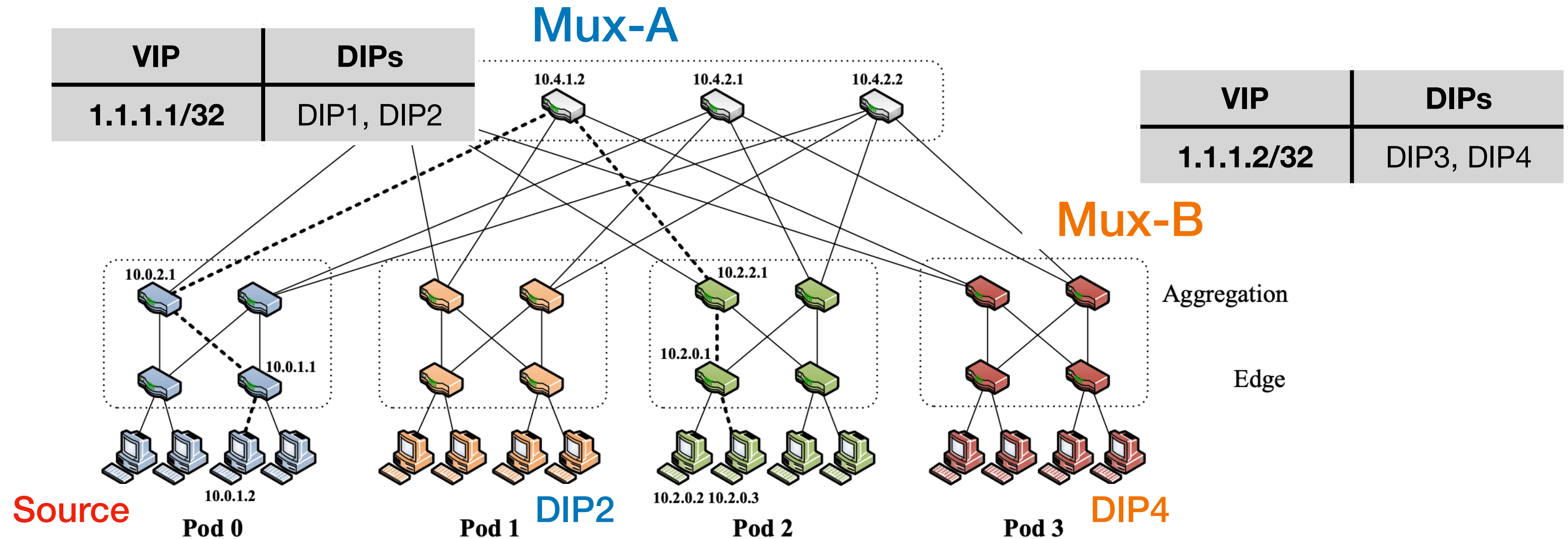
But a switch has limited table entries!

- 16K in the forwarding table
- 512 in the tunneling table

How can we solve it?

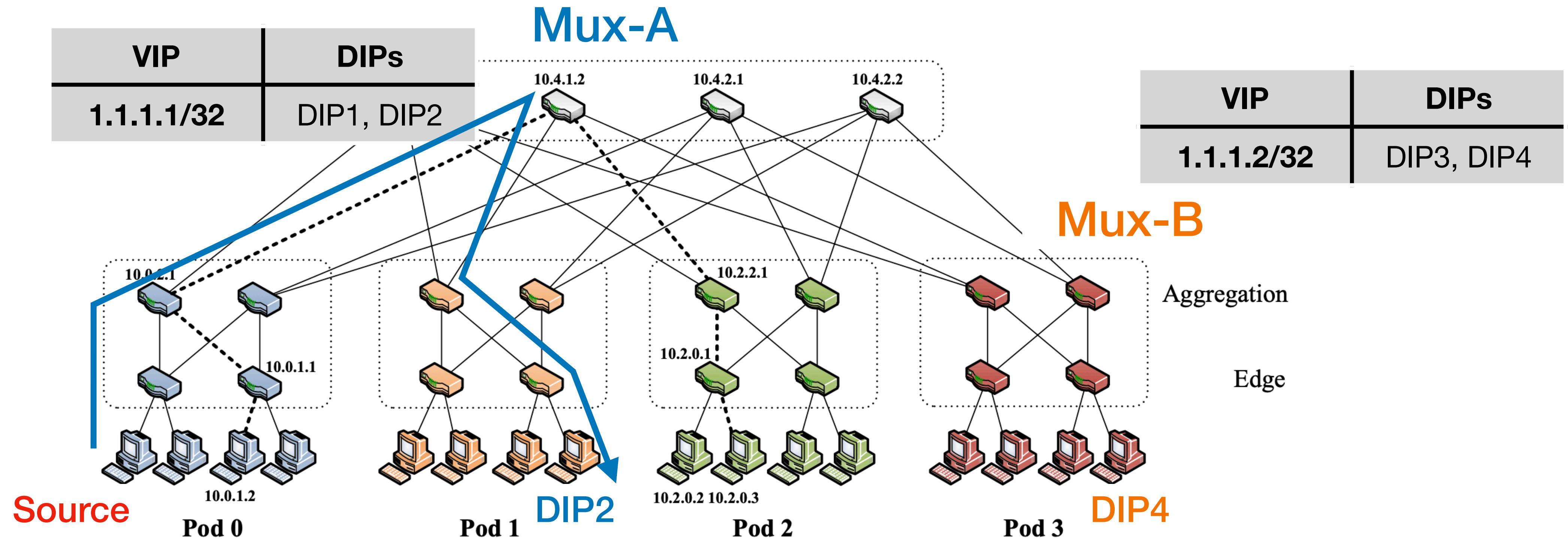
Technique #2: VIP Partition

- Divide VIP-to-DIP mapping across multiple switches



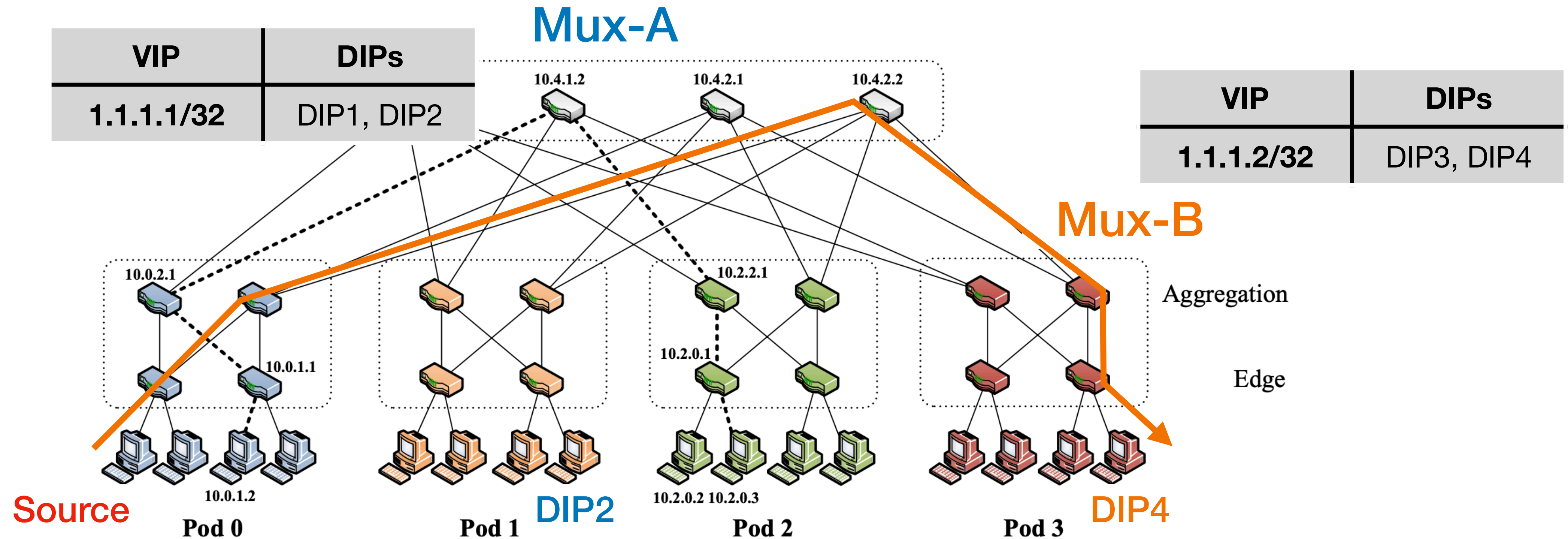
Technique #2: VIP Partition

- Divide VIP-to-DIP mapping across multiple switches



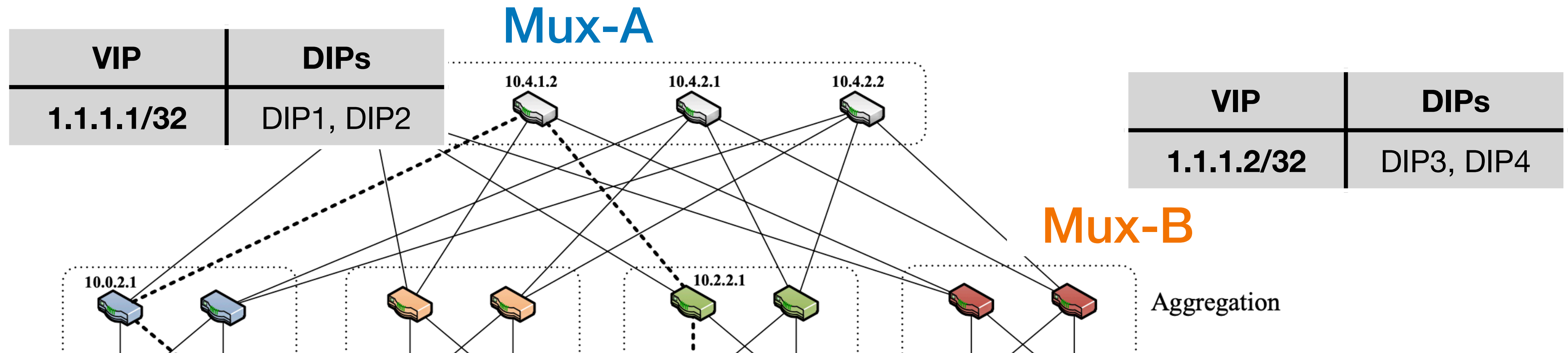
Technique #2: VIP Partition

- Divide VIP-to-DIP mapping across multiple switches



Technique #2: VIP Partition

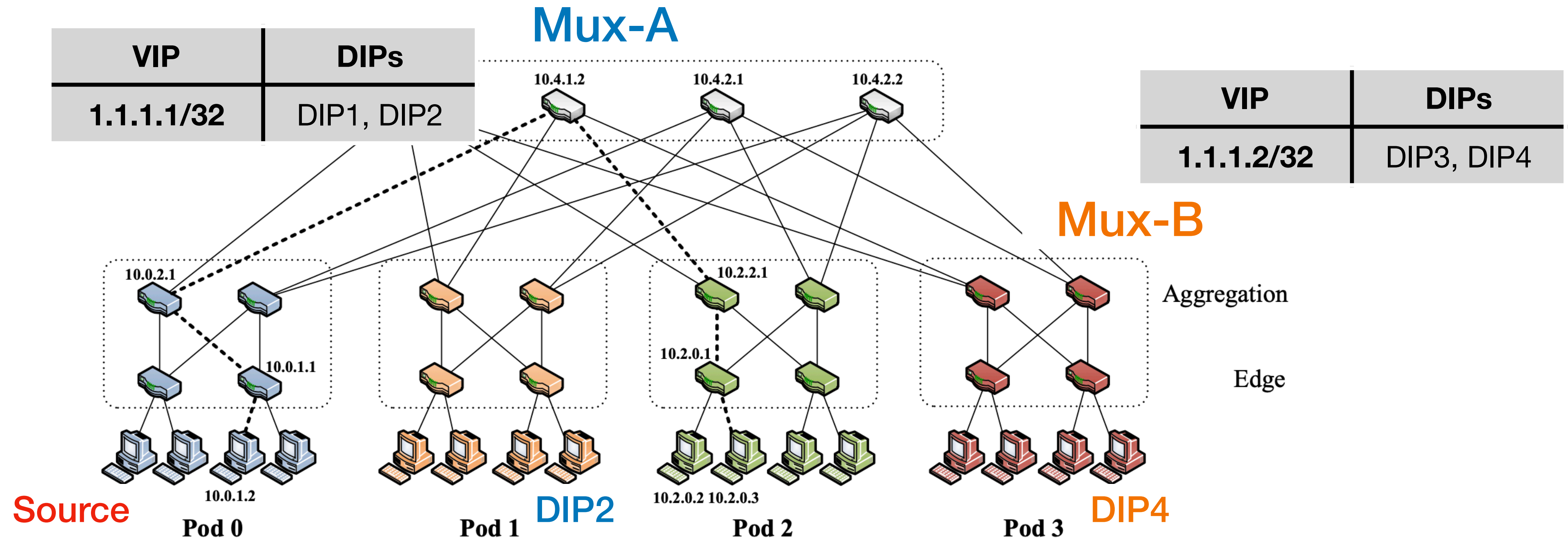
- Divide VIP-to-DIP mapping across multiple switches



But what happens if none of the on-path switches have this table entry?

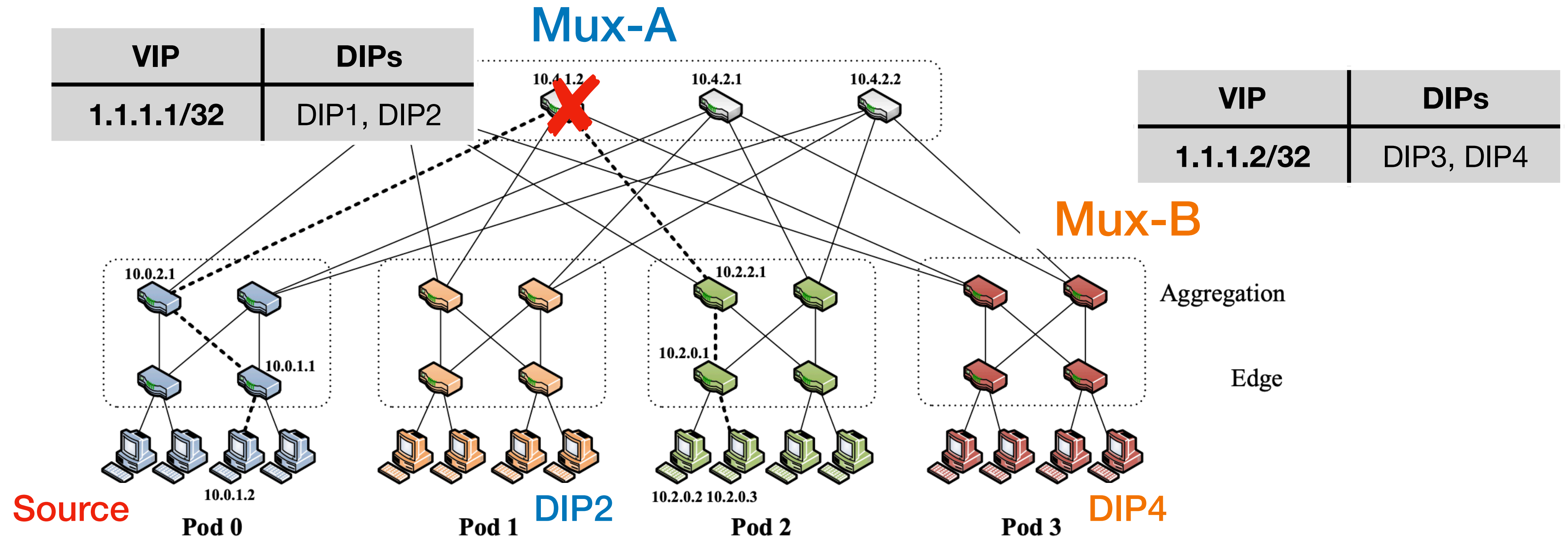
Technique #2: VIP Partition

- Divide VIP-to-DIP mapping across multiple switches
- BGP announcement

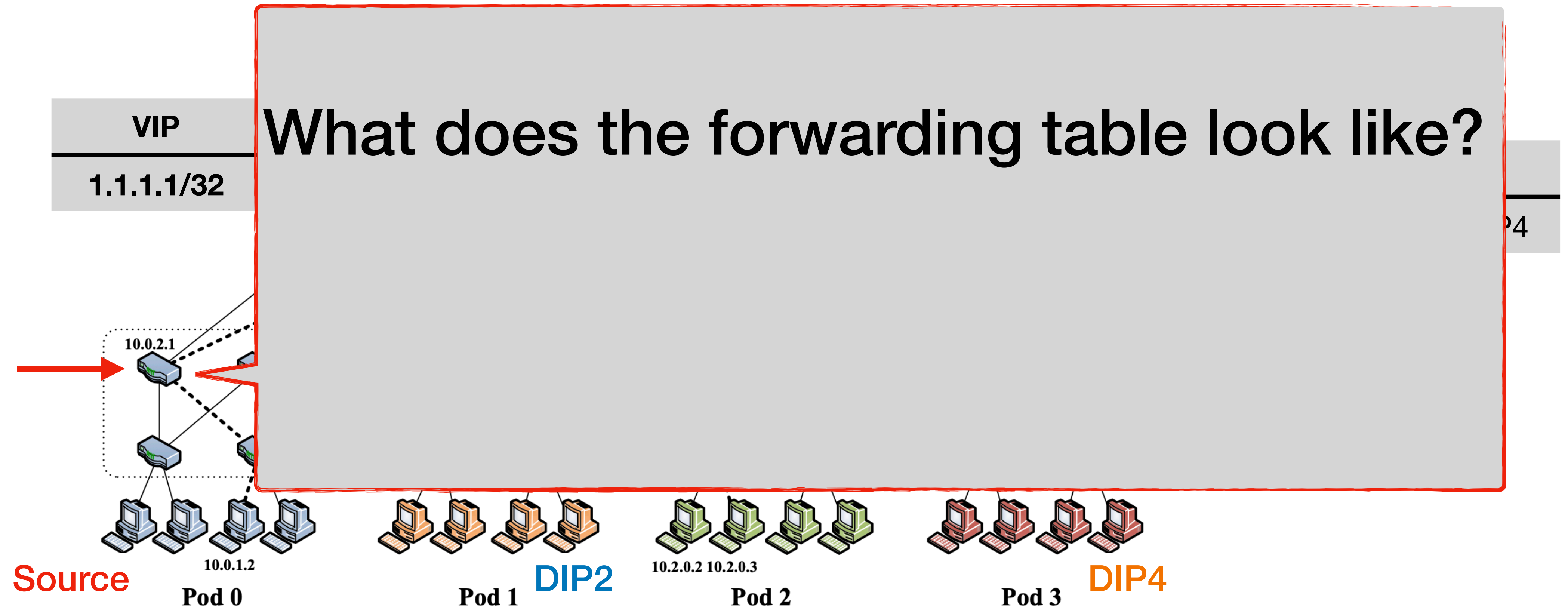


Seems problem solved?

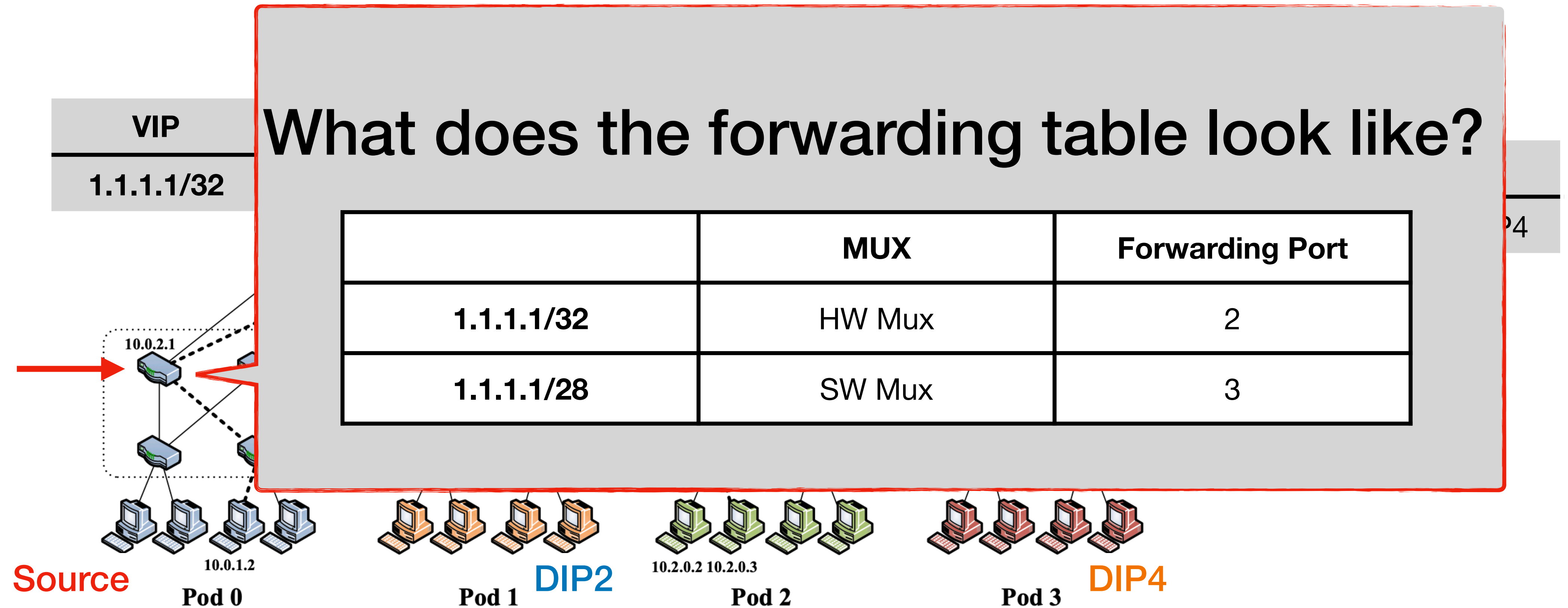
What happens if Mux-A fails



What happens if Mux-A fails

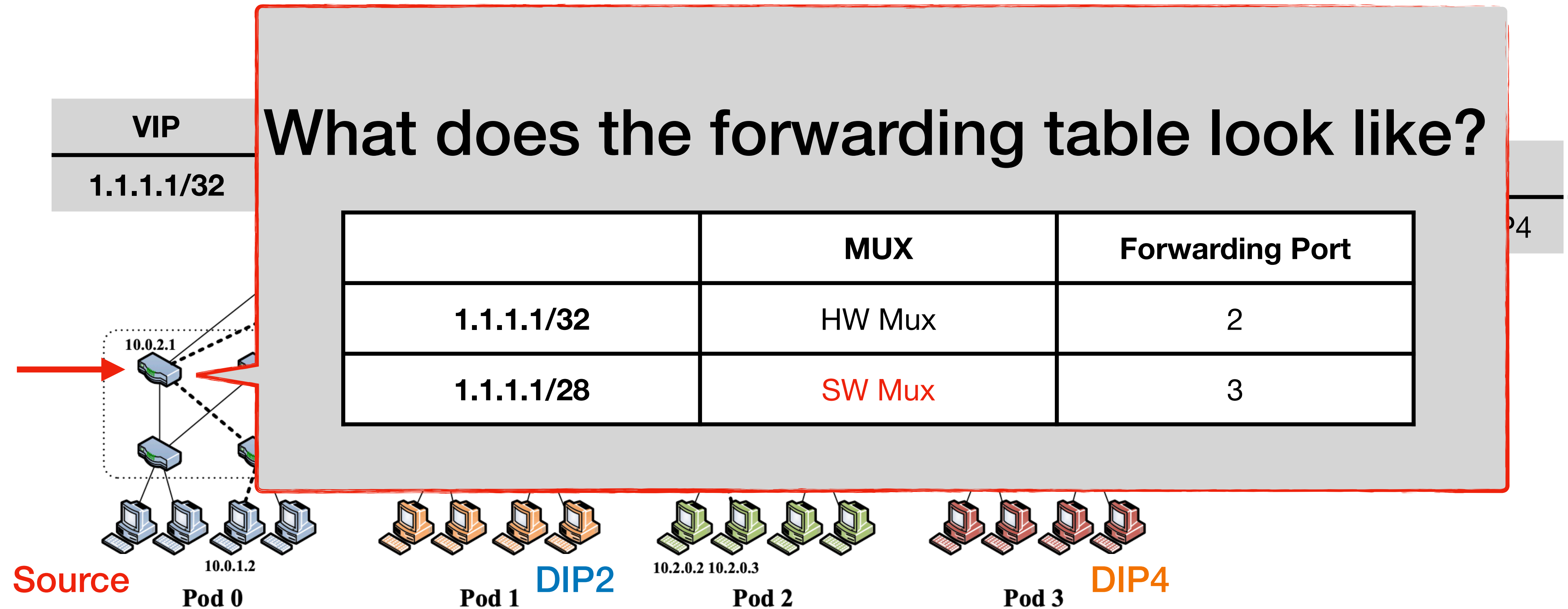


What happens if Mux-A fails



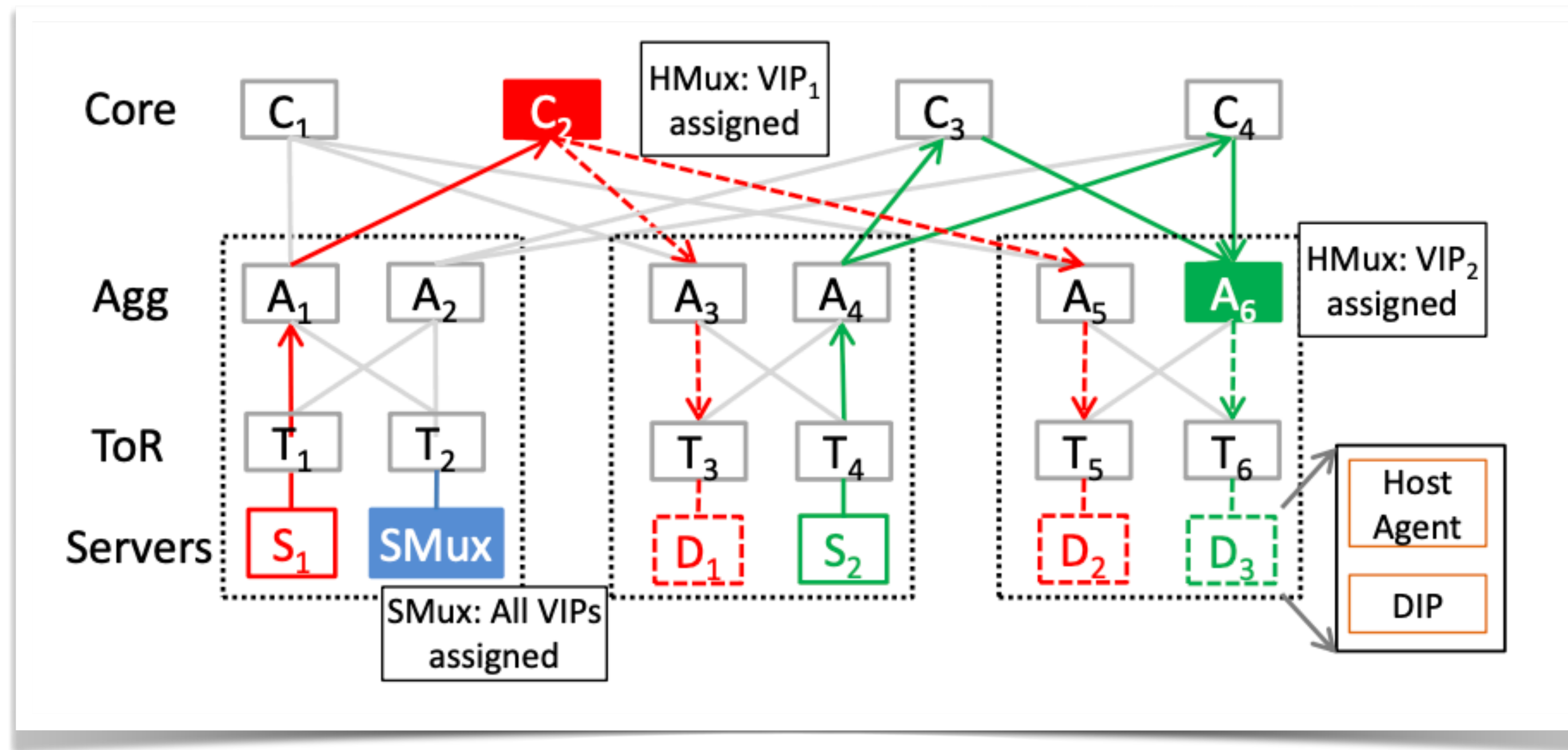
What happens if Mux-A fails

- Use LPM to make the decision



Duet: HW Mux + SW Mux (Technique #3)

- HW: commodity switch => Primary
- SW: x86 server box => Secondary



Are we done?

What is Duet fundamentally?

What is Duet fundamentally?

Load balancer!

What is Duet fundamentally?

Load balancer!

How does Duet balance the traffic load?

What is Duet fundamentally?

Load balancer!

How does Duet balance the traffic load?

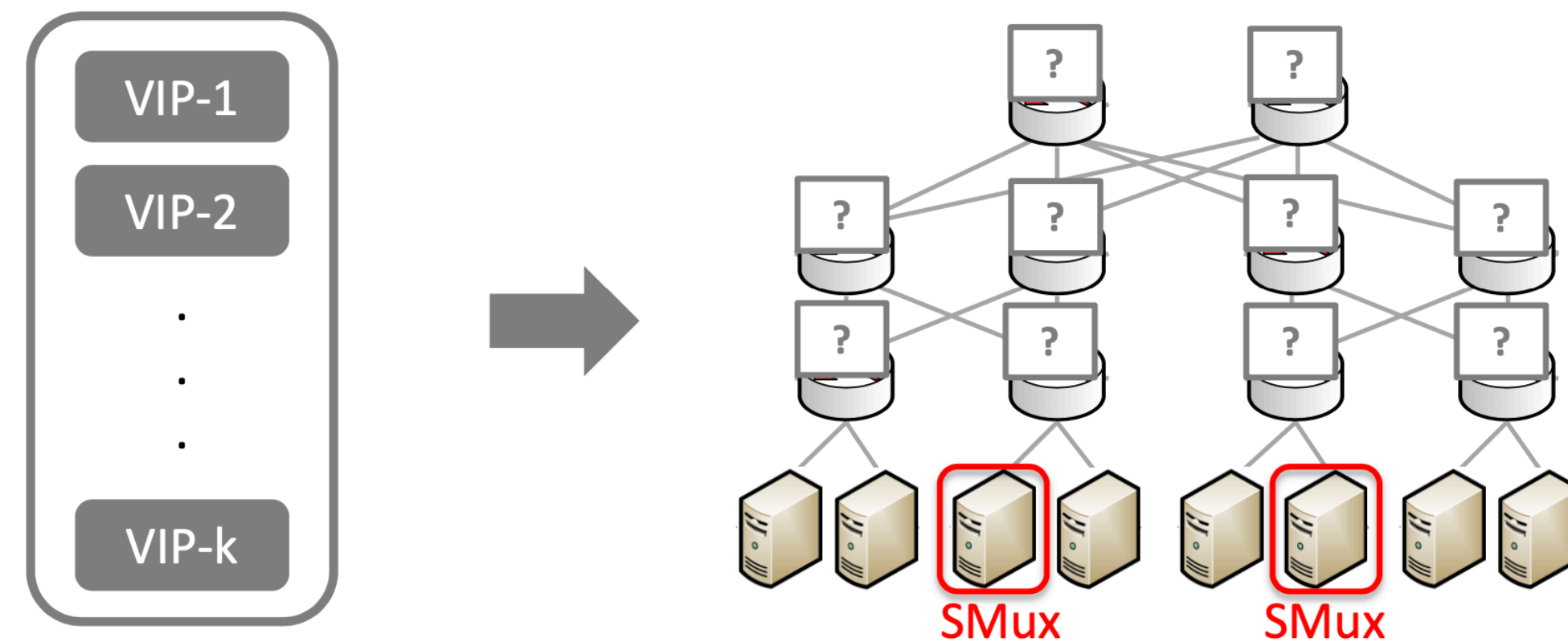
Technique #4: A Greedy VIP Assignment Algorithm

- Problem formulation: multi-dimensional bin-packing problem
 - Resources: switch memory and link bandwidth
 - Objects: VIPs
- Goal: find the VIP-switch assignment maximizing VIP traffic load
 - Maximum resource utilization (MRU)
- Constraints
 - Any VIP-switch assignment should not exceed the capacity of a link
 - Any VIP-switch assignment should not exceed the switch memory size

Algorithm Details

- Sketch: walk through VIPs and maximize MRU
 - MRU can be accumulated

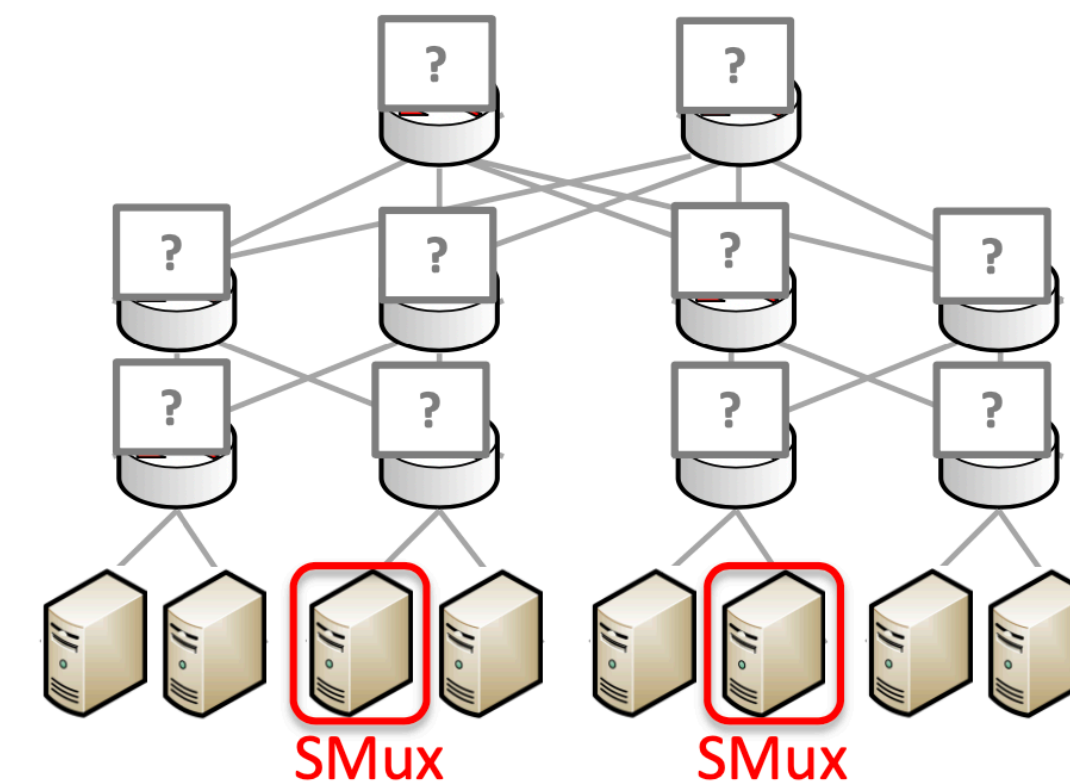
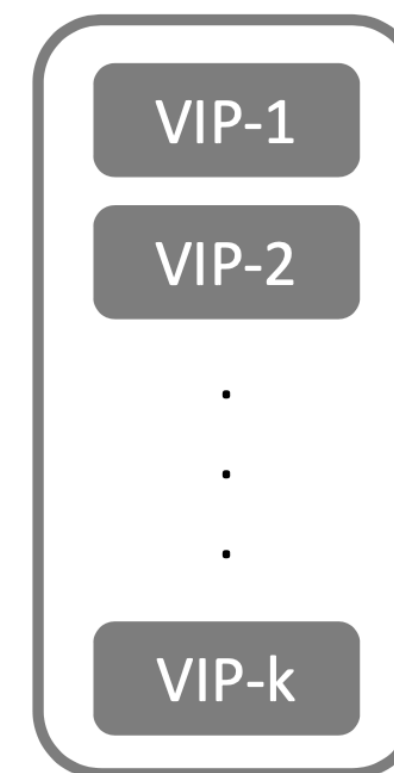
Notation	Explanation
V	Set of VIPs
d_v	Set of DIPs for the v-th VIP
S, E	Set of switches and links respectively
R	Set of resources (switches and links)
C_i	Capacity of i-th resource
$t_{i,s,v}$	v-th VIP's traffic on i-th link, when it is assigned to s-th switch
$L_{i,s,v}$	load (additional utilization) on i-th resource if v-th VIP is assigned to s-th switch
$U_{i,s,v}$	Cumulative utilization of i-th resource if v-th VIP is assigned to s-th switch
$U_{i,v}$	Cumulative utilization of i-th resource after v VIPs have been assigned
$MRU_{s,v}$	Max. Resource Utilization (MRU) after v-th VIP is assigned to s-th switch



Algorithm Details

- Sketch: walk through VIPs and maximize MRU
 - MRU can be accumulated

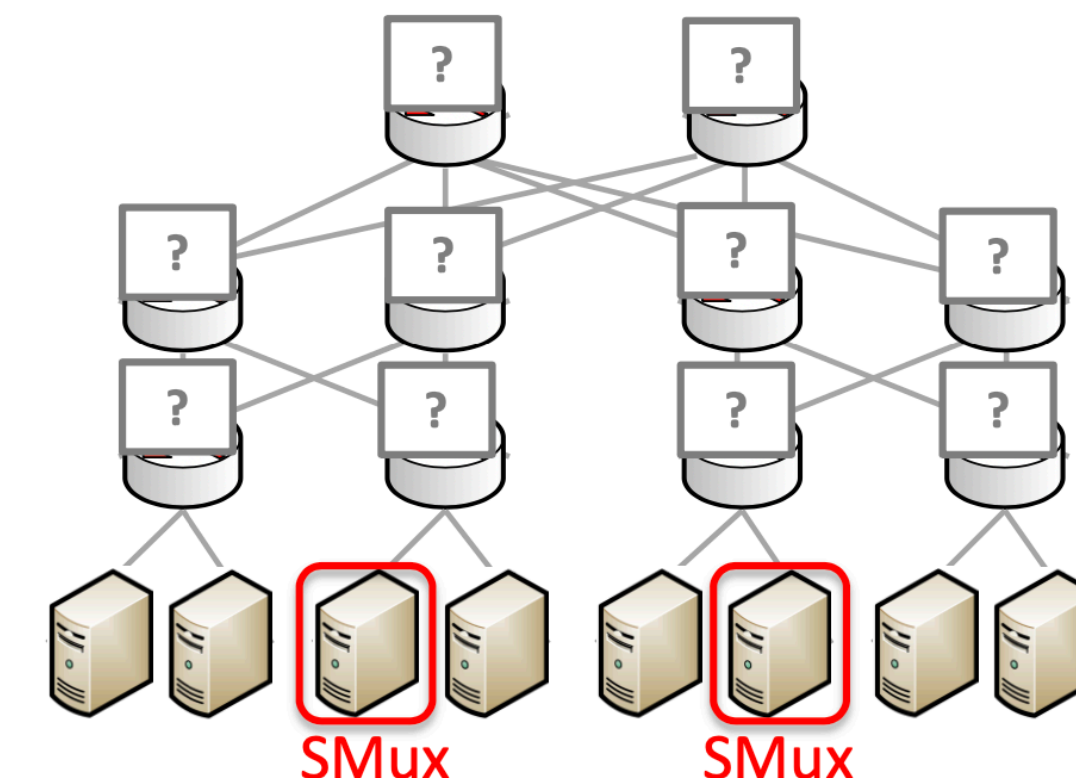
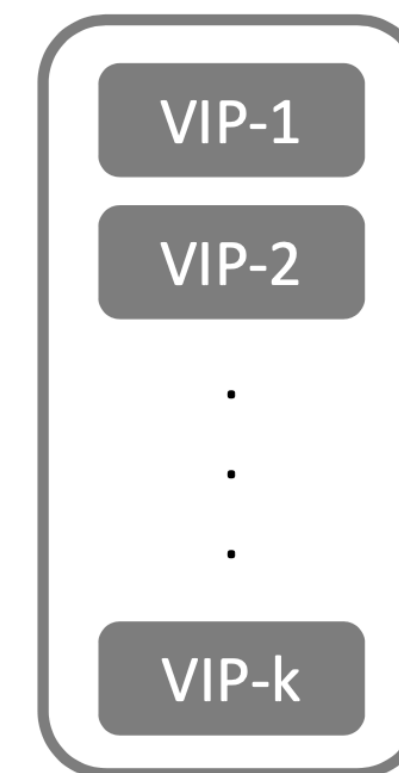
Notation	Explanation
V	Set of VIPs
d_v	Set of DIPs for the v-th VIP
S, E	Set of switches and links respectively
R	Set of resources (switches and links)
C_i	Capacity of i-th resource
$t_{i,s,v}$	v-th VIP's traffic on i-th link, when it is assigned to s-th switch
$L_{i,s,v}$	load (additional utilization) on i-th resource if v-th VIP is assigned to s-th switch
$U_{i,s,v}$	Cumulative utilization of i-th resource if v-th VIP is assigned to s-th switch
$U_{i,v}$	Cumulative utilization of i-th resource after v VIPs have been assigned
$MRU_{s,v}$	Max. Resource Utilization (MRU) after v-th VIP is assigned to s-th switch



Algorithm Details

- Sketch: walk through VIPs and maximize MRU
 - MRU can be accumulated

Notation	Explanation
V	Set of VIPs
d_v	Set of DIPs for the v-th VIP
S, E	Set of switches and links respectively
R	Set of resources (switches and links)
C_i	Capacity of i-th resource
$t_{i,s,v}$	v-th VIP's traffic on i-th link, when it is assigned to s-th switch
$L_{i,s,v}$	load (additional utilization) on i-th resource if v-th VIP is assigned to s-th switch
$U_{i,s,v}$	Cumulative utilization of i-th resource if v-th VIP is assigned to s-th switch
$U_{i,v}$	Cumulative utilization of i-th resource



How do we know this?

Technique #5:

- Rerun the algorithm periodically
 - Perform VIP-DIP migration

Technique #5: VIP Migration

- Rerun the algorithm periodically
 - Perform VIP-DIP migration

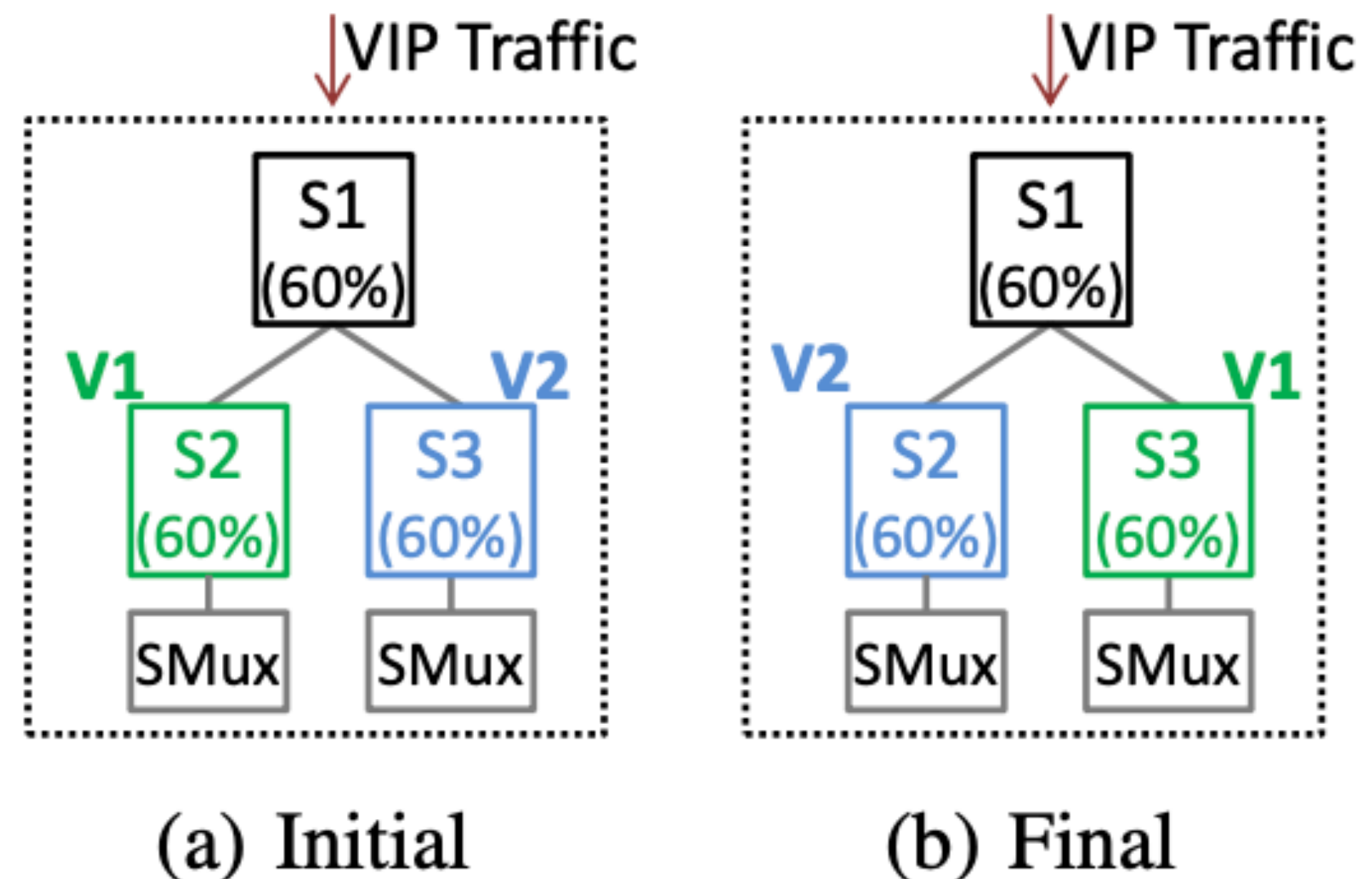
- How to minimize VM reshuffling?
- How to ensure correctness after migration?

Technique #5: VIP Migration

- Rerun the algorithm periodically
 - Perform VIP-DIP migration
- Non-sticky; make-before-break
 - Announce the VIP assignment before withdrawing from the old one

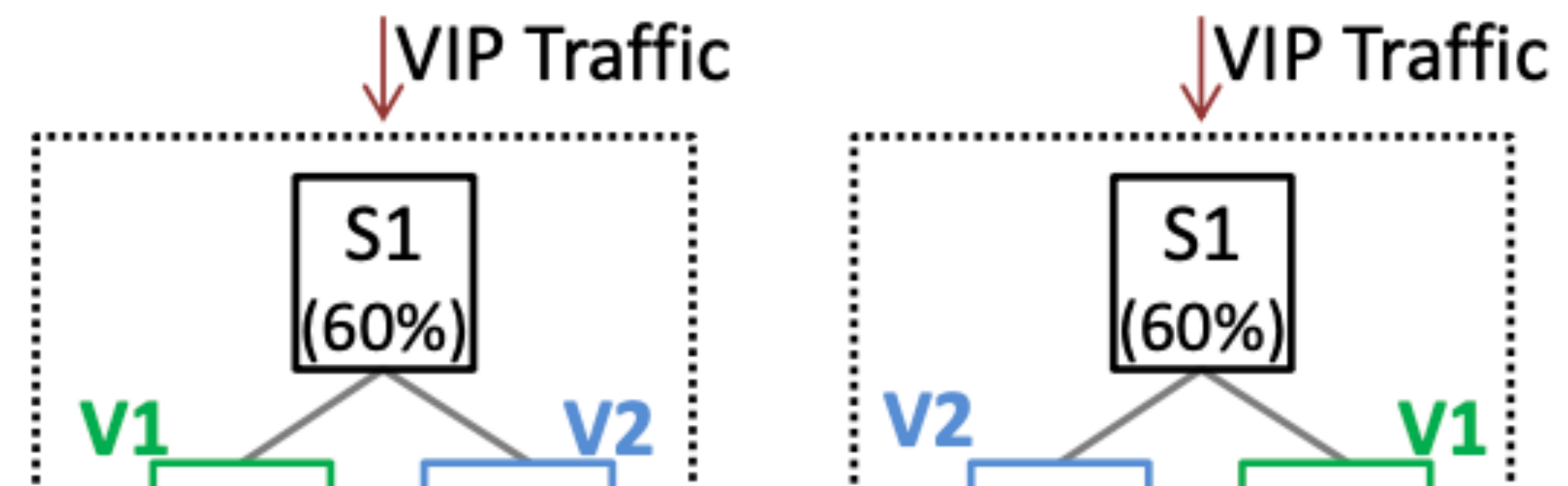
Technique #5: VIP Migration

- Rerun the algorithm periodically
 - Perform VIP-DIP migration
- Non-sticky; make-before-break
 - Announce the VIP assignment before withdrawing from the old one
- Issues:
 - Memory deadlock
 - Reshuffling is heavy



Technique #5: VIP Migration

- Rerun the algorithm periodically
 - Perform VIP-DIP migration
- Non-sticky; make-before-break
 - Announce the VIP assignment before withdrawing from the old one
- Issues:
 - Memory deadlock
 - Reshuffling is heavy



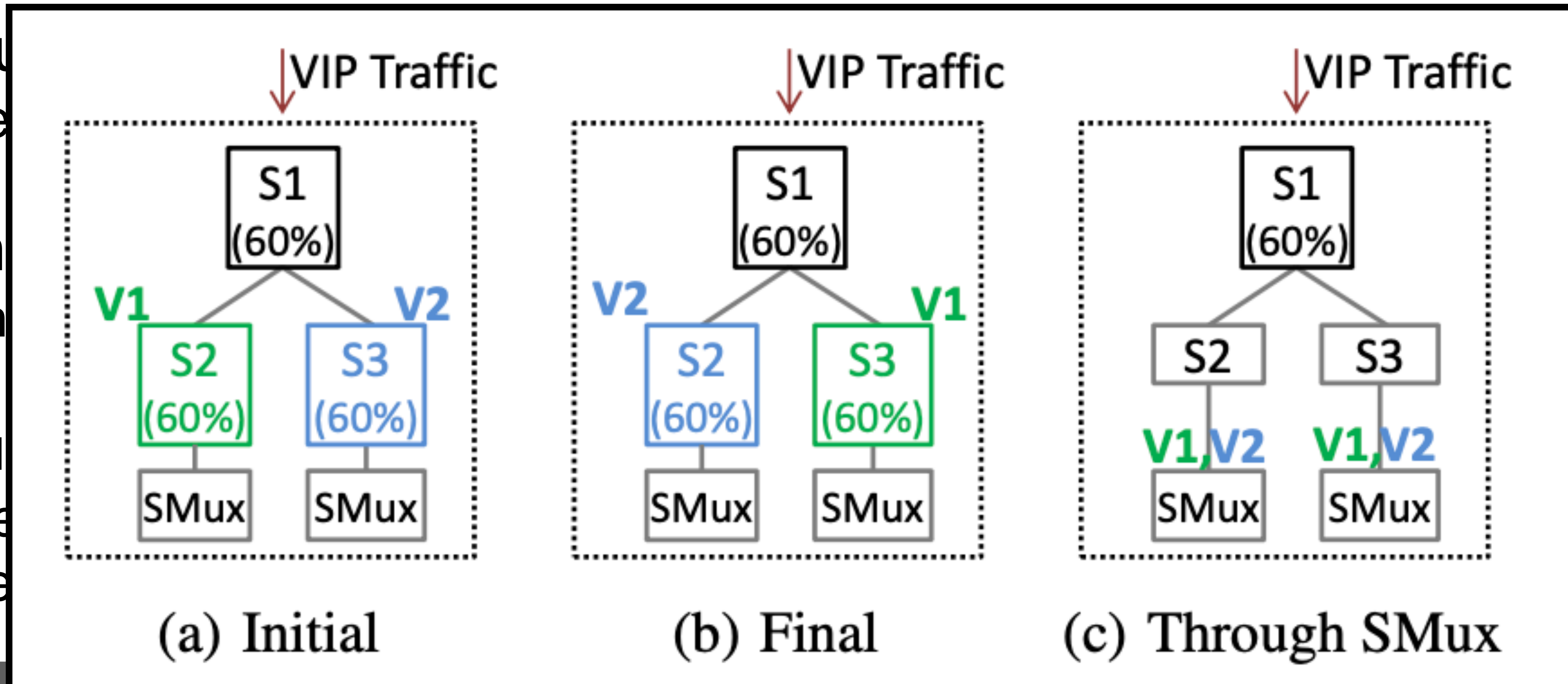
Migration via SMux

(a) Initial

(b) Final

Technique #5: VIP Migration

- Rerouting
- Performance
- Non-intrusive
- Anomaly
- Issues
- Measurement
- Results



Migration via SMux

(a) Initial

(b) Final

Combine Everything Together

- Technique #1: chained table execution in HMux
- Technique #2: VIP partition over multiple HMuxes
- Technique #3: SMux as a backup
- Technique #4: Greedy VIP assignment
- Technique #5: VIP migration

HW-SW co-designed gateway continues!

Azure Boost

- Cloud Infrastructure acceleration layer
 - ASIC/FPGA accelerator
 - MANA network interface
 - Dedicated ARM-based SoC



Summary

- Today
 - Gateway in data center networks (II)
- Next week: host network virtualization
 - Andromeda (NSDI'18)
 - PicNIC (SIGCOMM'19)