Microsoft

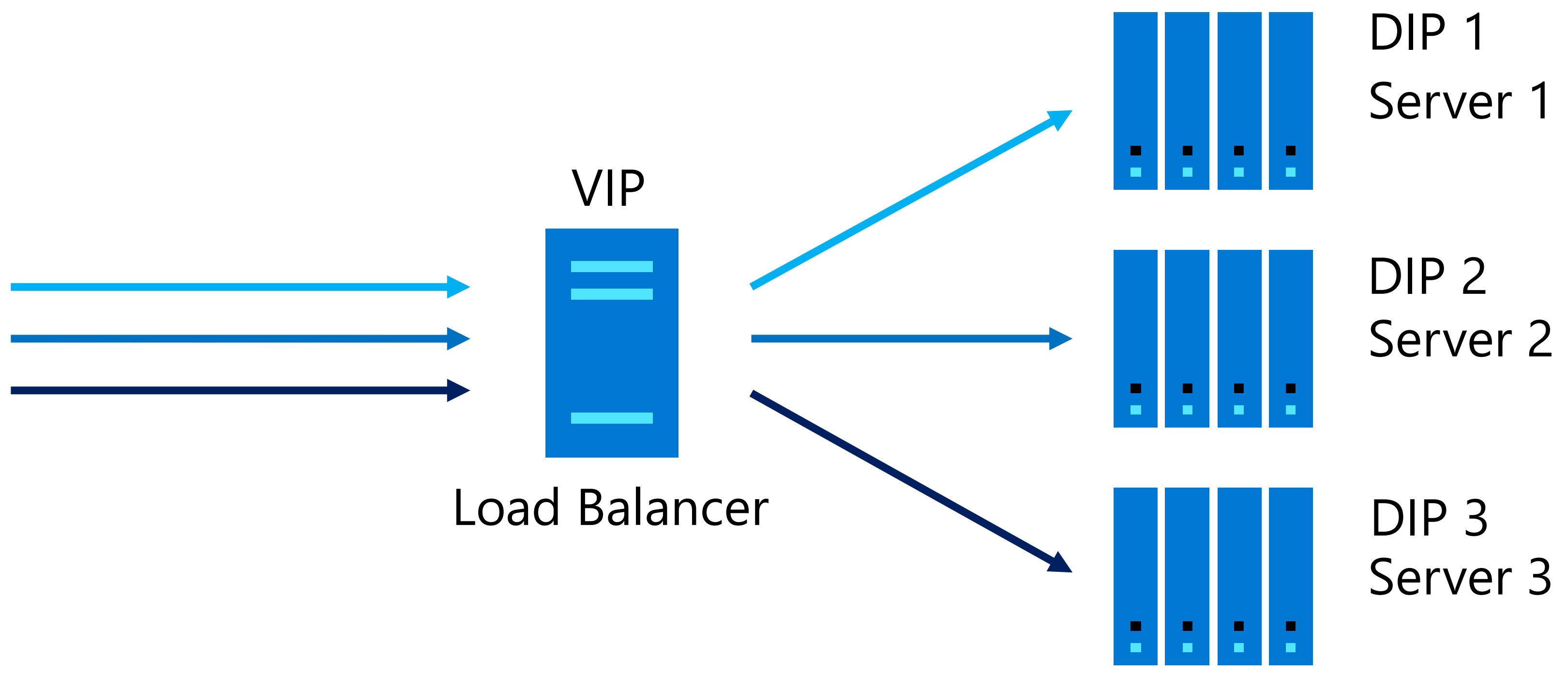
SlimeMold: Hardware Load Balancer at Scale in Datacenter

Ziyuan Liu, Zhixiong Niu, Ran Shu, Liang Gao, Guohong Lai, Na Wang, Zongying He, Jacob Nelson, Dan R. K. Ports, Lihua Yuan, Peng Cheng, Yongqiang Xiong

Beihang University, Microsoft, Ragile Networks Inc., Broadcom Inc.

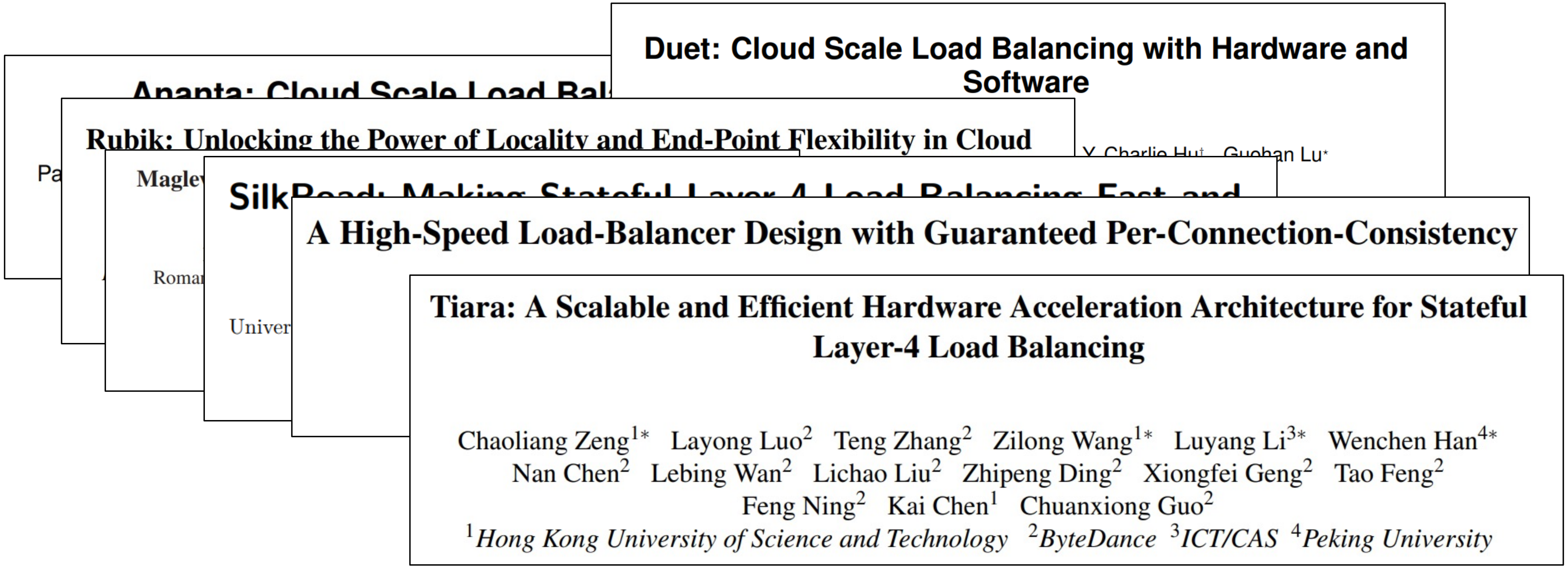
Background – L4 Load Balancer in Data Center

L4 Load Balancer: distribute packets to backend server pool



Background – L4 Load Balancer in Data Center

L4 Load Balancer: distribute packets to backend server pool



Duet: Cloud Scale Load Balancing with Hardware and Software

Y. Charlie Hu† Guohan Lu*

Ananta: Cloud Scale Load Bal

Pa

Rubik: Unlocking the Power of Locality and End-Point Flexibility in Cloud

Maglev

Roman

SilkRoad: Making Stateful Layer-4 Load Balancing Fast and

Univer

A High-Speed Load-Balancer Design with Guaranteed Per-Connection-Consistency

Tiara: A Scalable and Efficient Hardware Acceleration Architecture for Stateful Layer-4 Load Balancing

Chaoliang Zeng[1*] Layong Luo[2] Teng Zhang[2] Zilong Wang[1*] Luyang Li[3*] Wenchen Han[4*]
Nan Chen[2] Lebing Wan[2] Lichao Liu[2] Zhipeng Ding[2] Xiongfei Geng[2] Tao Feng[2]
Feng Ning[2] Kai Chen[1] Chuanxiong Guo[2]
[1]*Hong Kong University of Science and Technology* [2]*ByteDance* [3]*ICT/CAS* [4]*Peking University*

Background – Stateful Load Balancer

L4 Load Balancer: distribute packets to backend server pool

Most production L4 load balancer is **stateful**

- ConnTable: stores flow to DIP mapping
- Examples: Ananta [1], Maglev [2], …

Flow	DIP
5-tuple	IP address
…	…

Often use software LB (SLB) for agility and reliability

[1] Patel, Parveen, et al. "Ananta: Cloud scale load balancing." ACM SIGCOMM'13. 2013.

[2] Eisenbud, Daniel E., et al. "Maglev: A fast and reliable software network load balancer." USENIX NSDI'16. 2016.

Background – HLB

SLB incurs significant costs

- Limited single node bandwidth
- Two orders of magnitude less than requirement
- Hundreds or even thousands of SLB nodes

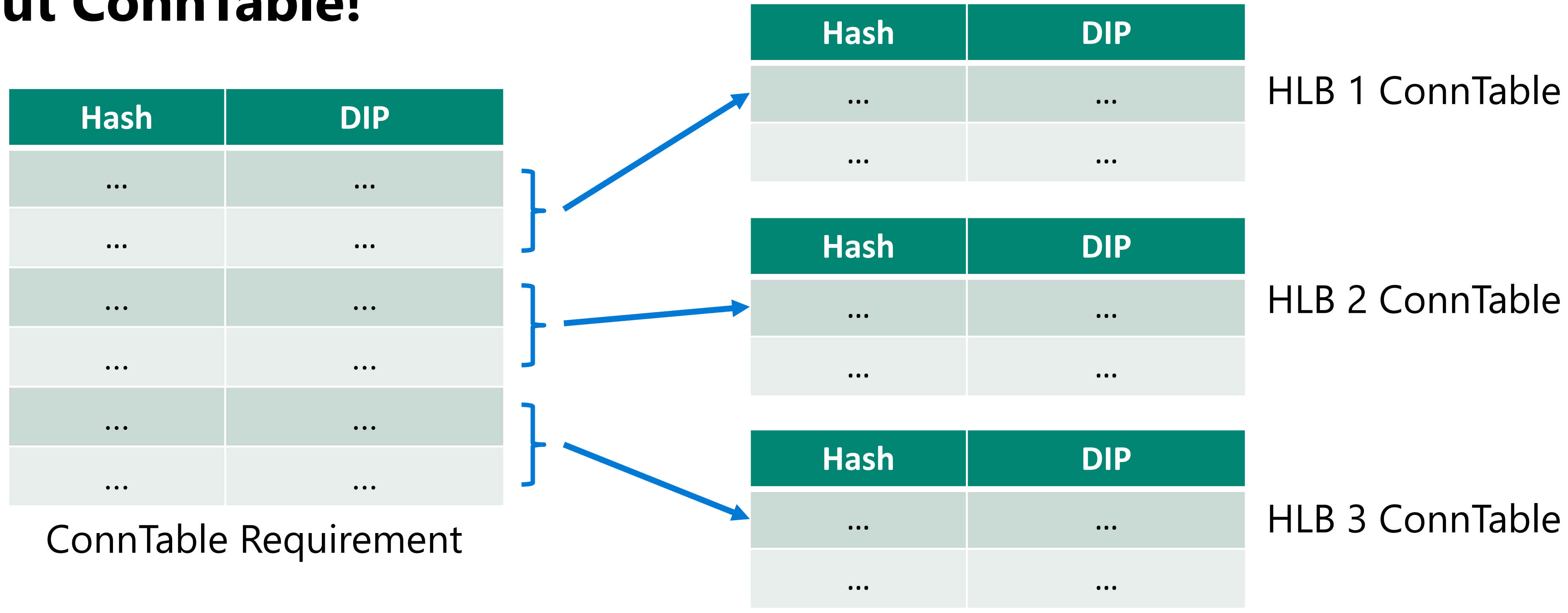
Trend: build hardware LB (HLB) using programmable switches

- Scale up performance
- High throughput density

Scale out HLB

HLB bottleneck: ConnTable capacity

Scale out ConnTable!

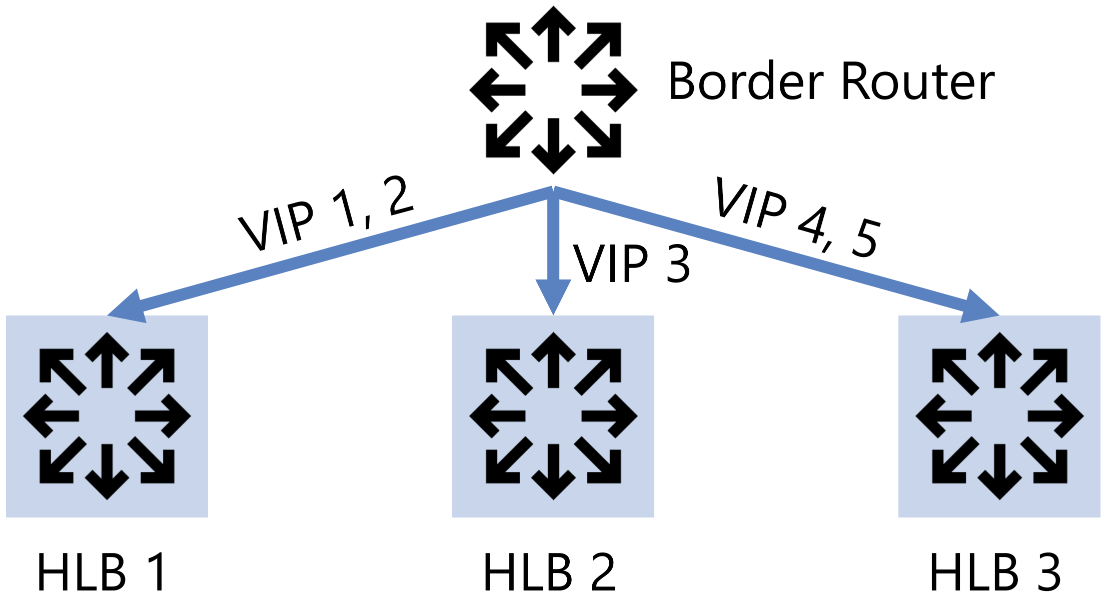


Existing Solution – VIP Partition

Each HLB only serves part of VIPs

Limitation: capacity and efficiency

- Cannot serve giant VIPs
- Load imbalance due to static partition

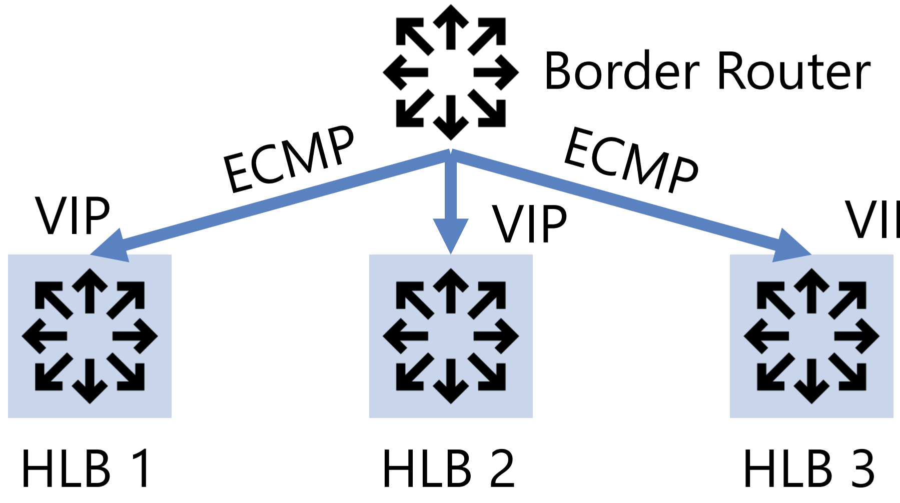


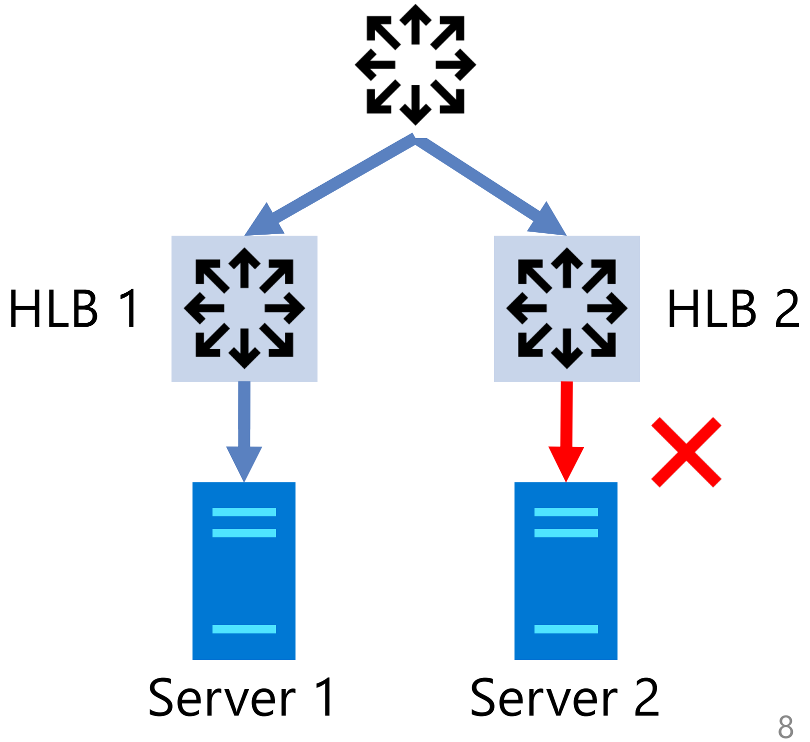
Existing Solution – ECMP

Border router distribute traffic to HLBs using ECMP

Limitation: Potential PCC violation

- Per-Connection Consistency (PCC): a flow should be served by only one backend during its liveness
- ECMP reshuffle directs flow to another HLB, e.g., add a new HLB node



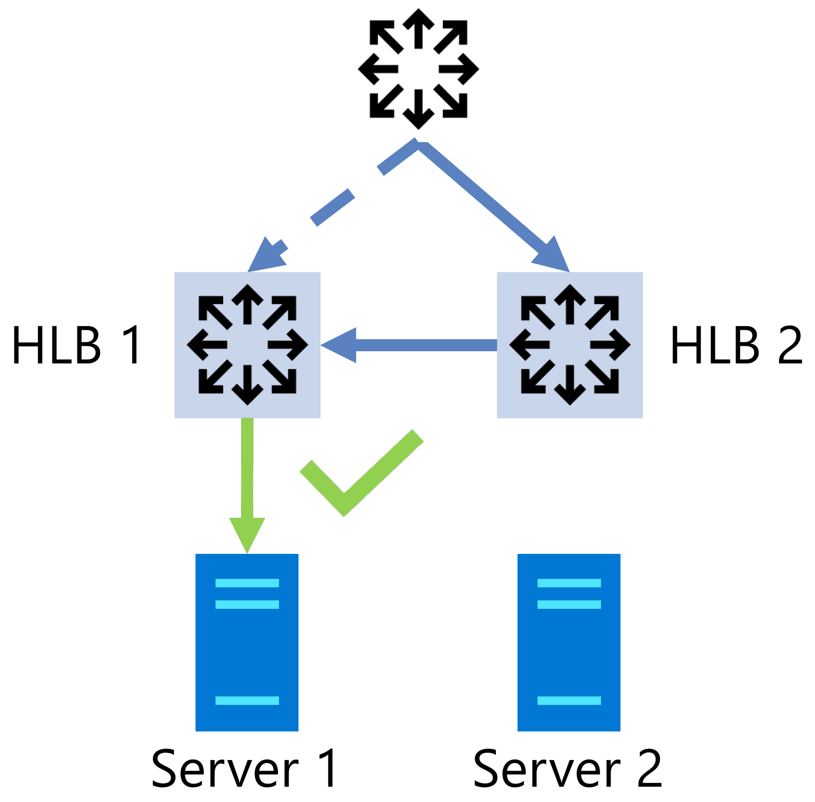


Our Solution – SlimeMold

Key idea: HLBs work *collaboratively* to take *consistent* actions

SlimeMold: Decouple HLB roles logically

- Forwarders: entry points that can always map a flow to the HLB who has its ConnTable entry
- State Owners: store part of ConnTable



Flow	State Owner
5-tuple	HLB 1
...	...

Our Solution – SlimeMold

Flow to State Owner table is as big as ConnTable

- Grouping flows as segments to reduce size

Simple flow to segment mapping

Want loads between segments evenly

- Flow hash (e.g., CRC32)

Segment as the unit of load distribution between State Owners

- Number of segment should be large enough to allow dynamic scaling, e.g., 10x number of State Owners

Flow	State Owner
5-tuple	State Owner 3
…	…

$\Rightarrow$

Flow Hash	State Owner
0x13B	State Owner 3
…	…

Splitting State Owner Table

Hash to State Owner table is too big

- Consumes unaffordable Forwarder table resource

Introduce Secondary Lookup to split the table into 2-level

$$\begin{tikzcd}\text{SRC} \arrow[r] & \text{Forwarder} \arrow[r] & \text{State Owner} \arrow[r] & \text{DIP}\end{tikzcd}$$

Hash	State Owner
0x13B	State Owner 3
…	…

Flow	DIP
5-tuple	DIP 10
…	…

$$\begin{tikzcd}\text{SRC} \arrow[r] & \text{Forwarder} \arrow[r] & \text{Secondary Lookup} \arrow[r] & \text{State Owner} \arrow[r] & \text{DIP}\end{tikzcd}$$

Hash	Sec. Lookup
0x13_	Sec. Lookup 1
…	…

Hash	State Owner
0x13B	State Owner 3
…	…

Flow	DIP
5-tuple	DIP 10
…	…

SlimeMold Overview

Forwarder

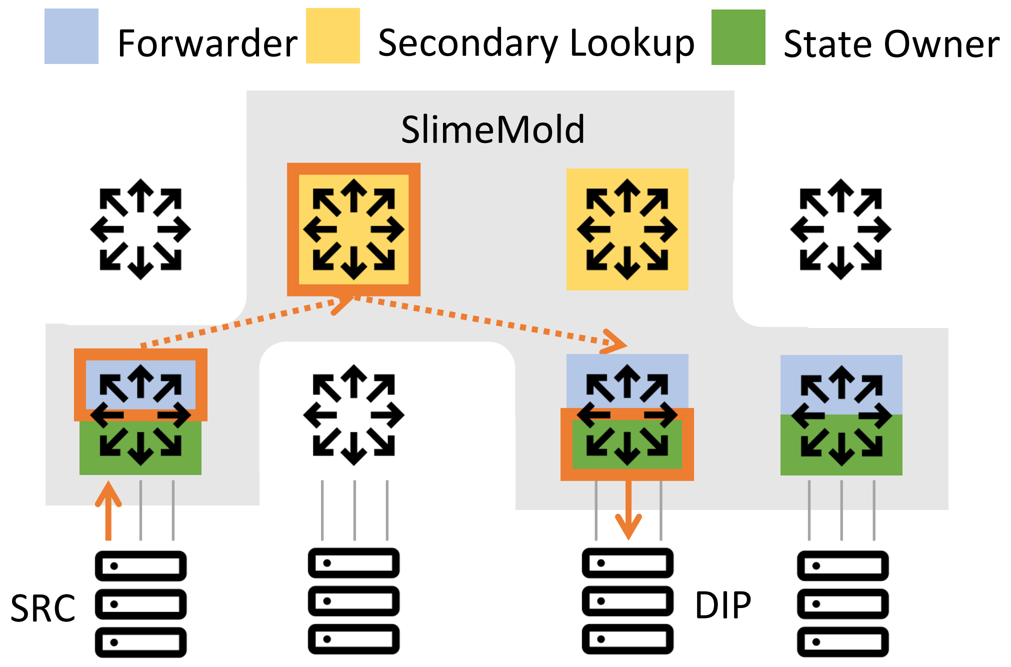
- Announces VIPs as entry point
- Routes packets to Secondary Lookup

Secondary Lookup

- Routes packets to State Owner

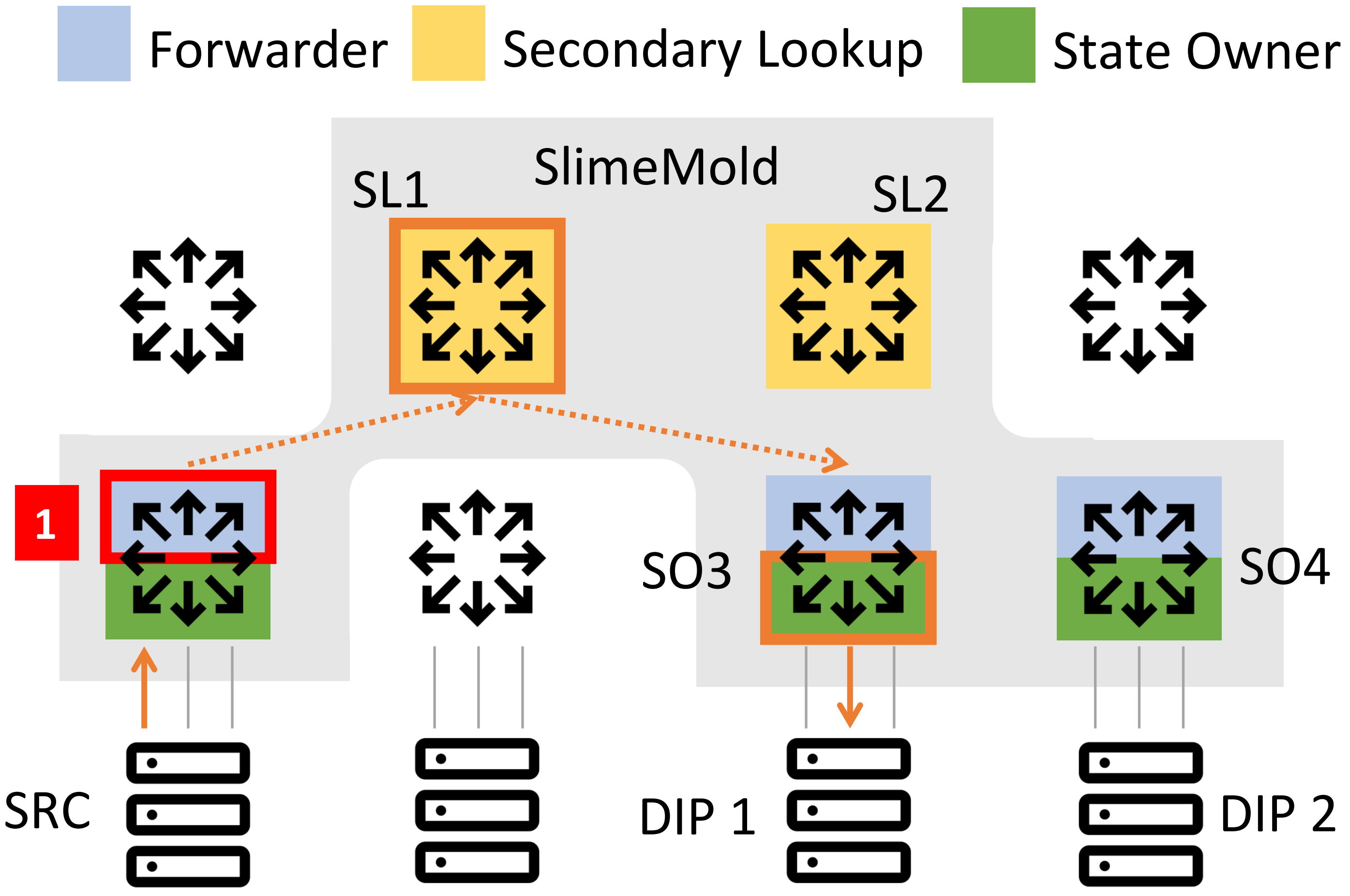
State Owner

- Exclusively owns part of flow states
- Forwards packet to DIP



Note: multiple roles may locate on a same physical node

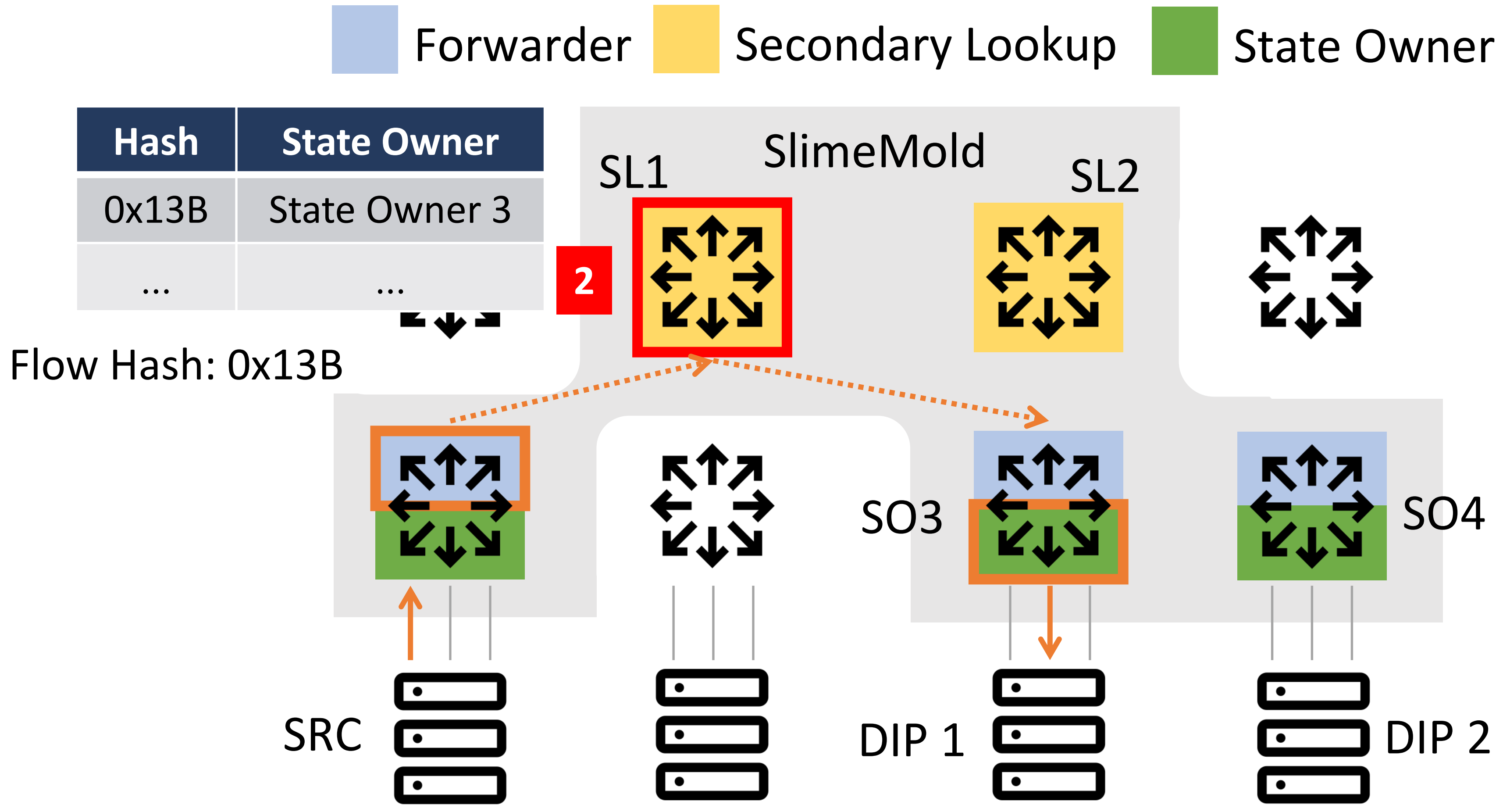
SlimeMold Workflow



Hash	Sec. Lookup
0x13_	Sec. Lookup 1
…	…

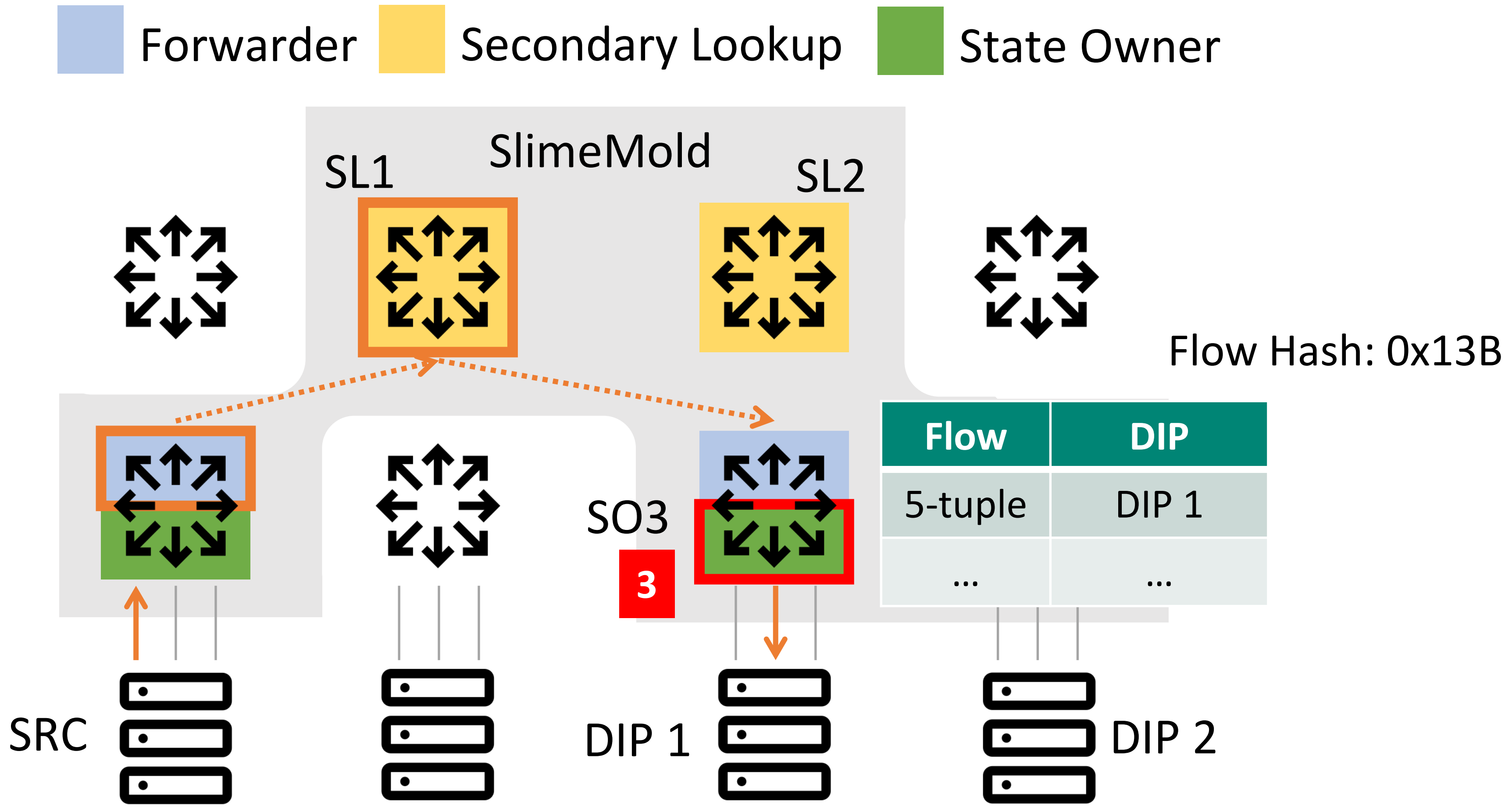
Flow Hash: 0x13B

SlimeMold Workflow



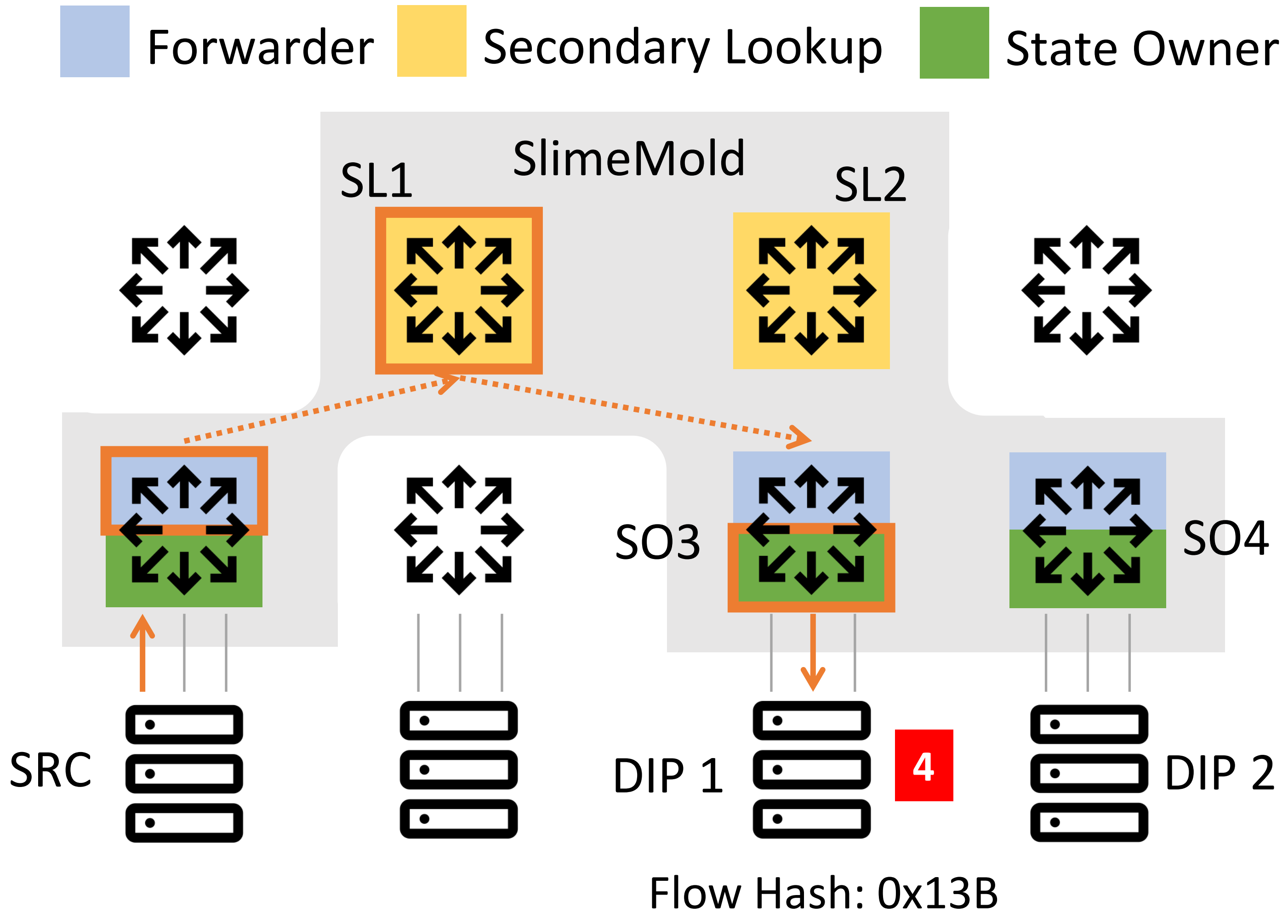
Hash	State Owner
0x13B	State Owner 3
...	...

SlimeMold Workflow



Flow	DIP
5-tuple	DIP 1
…	…

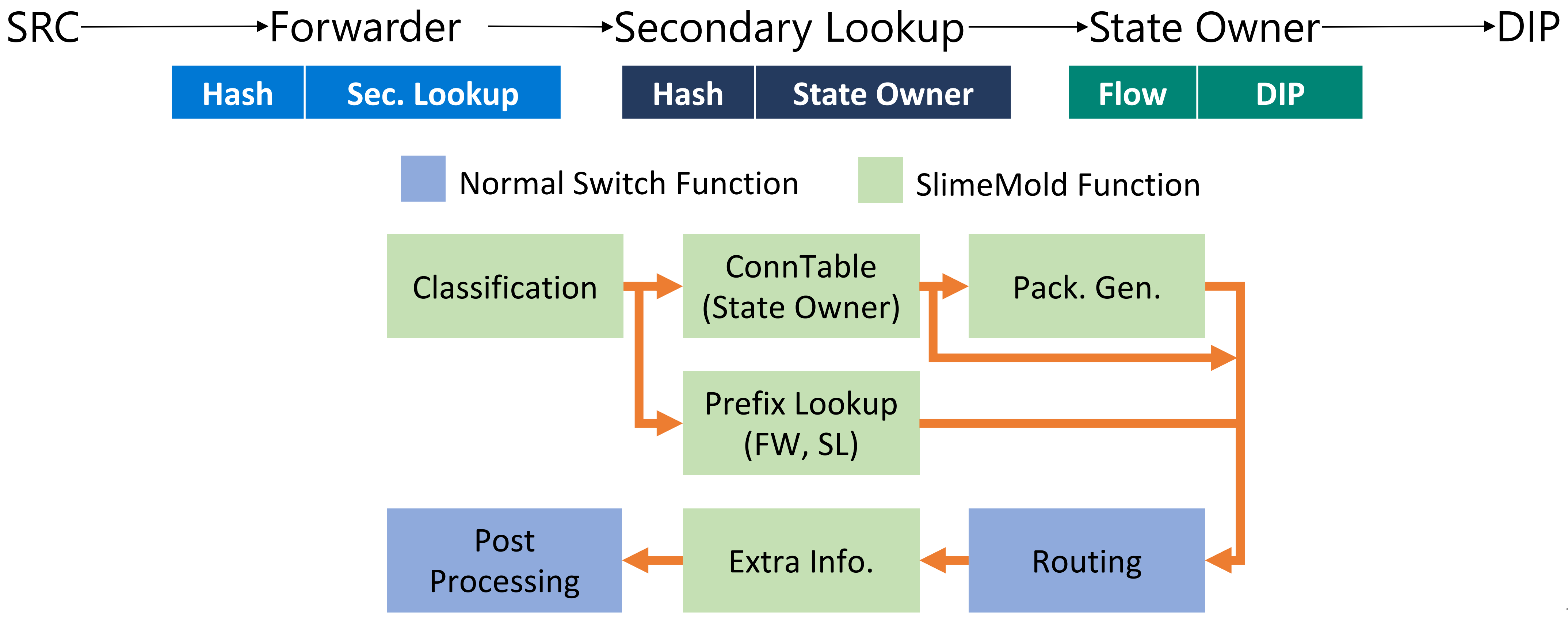
SlimeMold Workflow



SlimeMold Building Block

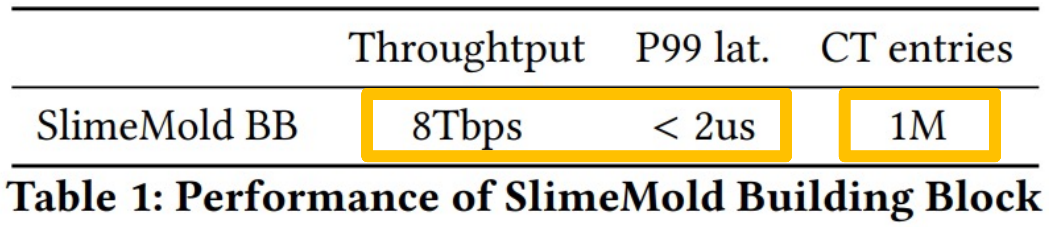
Building block: a switch that support full set of SlimeMold roles

- Can be configured as any combination of SlimeMold roles



Evaluation – Building Block Performance

We build a prototype using **Ragile programmable switch** equipped with **Broadcom BCM56788 SmartToR** chip



	Throughtput	P99 lat.	CT entries
SlimeMold BB	8Tbps	< 2us	1M

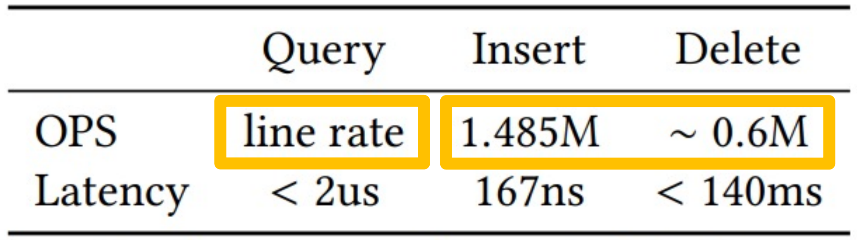
Table 1: Performance of SlimeMold Building Block

Line rate with low latency

1 M ConnTable entries

Evaluation – ConnTable Performance

We build a prototype using **Ragile programmable switch** equipped with **Broadcom BCM56788 SmartToR** chip



	Query	Insert	Delete
OPS	line rate	1.485M	~ 0.6M
Latency	< 2us	167ns	< 140ms

Table 2: Performance of ConnTable Operations

Line rate ConnTable lookup

Near 1.5 MOPS insertion and ~0.6 MOPS deletion

- hardware-learning based insertion is extremely faster than existing control-plane based solution

Large Scale Simulation

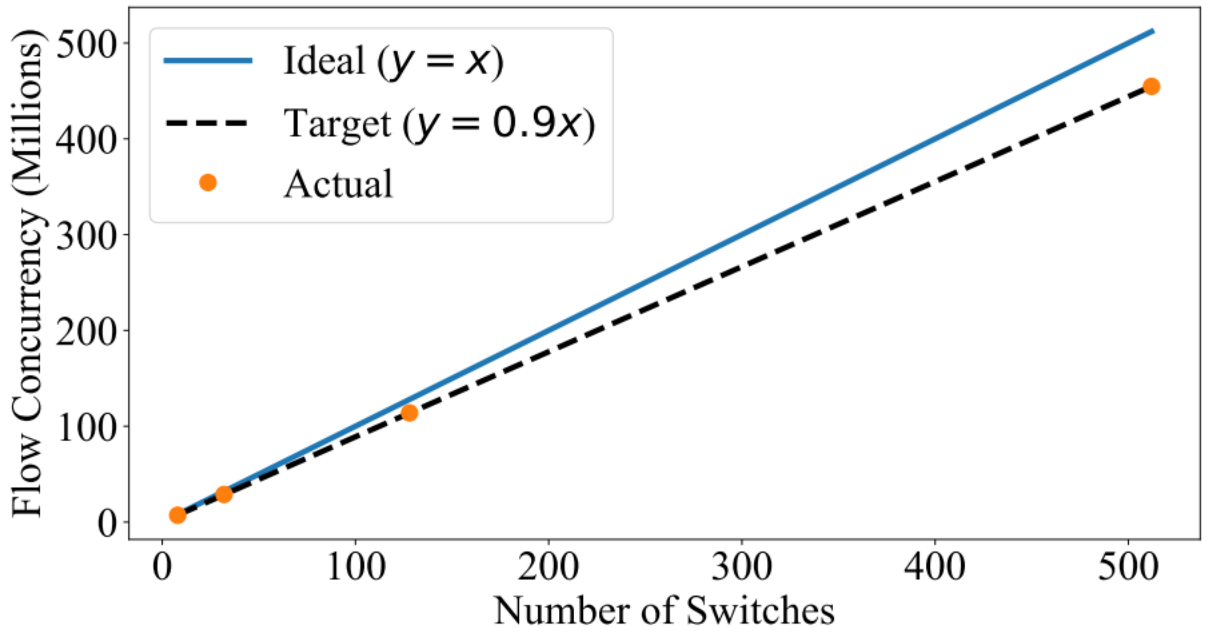


Figure 4: Scalability

Highly efficient scale out

Linear scalability

Conclusion

SlimeMold: a collaborative scalable hardware load balancer for data centers

- High performance building block prototype
- Linear scalability and high efficiency

Q&A

Backup

DIP Decision

A separate service out of SlimeMold

Interactions between SlimeMold

- Direct to the service when a State Owner should but does not have the ConnTable entry
- Handle all following packets within SlimeMold itself

Only needs to handle first several packets of a flow

Free to use any LB algorithm

Allow to make *inconsistent* decision

- Arbitrate by State Owner

Detour in SlimeMold

Multiple optimizations can be leveraged

- Secondary Lookup placement policy: to reduce detour between Forwarder and State Owner
- ConnTable cache on Forwarder

Segment to State Owner Table

Almost static to avoid frequent synchronization overhead

- A flow will change ConnTable, but not segment to State Owner table