

For the best experience, view the presentation in canva:

<https://canva.link/t0kv61x40g7ehk5>

大象、老鼠、森林： 基於機器學習的雙時間尺度負載平衡器

A Two-Timescale Load Balancer Guided by ML

專題生：石浚邑、石維廉

指導教授：張燕光



Cited by: 363

DRILL: Micro Load Balancing for Low-latency Data Center Networks

Soudeh Ghorbani*

University of Wisconsin-Madison

Zibin Yang

University of Illinois at
Urbana-Champaign

P. Brighten Godfrey

University of Illinois at
Urbana-Champaign

Yashar Ganjali

University of Toronto

Amin Firoozshahian

Intel

ABSTRACT

The trend towards simple datacenter network fabric strips most network functionality, including load balancing, out of the network core and pushes it to the edge. This slows reaction to microbursts, the main culprit of packet loss in datacenters. We investigate the opposite direction: could slightly smarter fabric significantly improve load balancing? This paper presents DRILL, a datacenter fabric for Clos networks which performs *micro load balancing* to distribute load as evenly as possible on microsecond timescales. DRILL employs per-packet decisions at each switch based on local queue occupancies and randomized algorithms to distribute load. Our design addresses the resulting key challenges of packet reordering and topological asymmetry. In simulations with a detailed switch hardware model and realistic workloads, DRILL outperforms recent edge-based load balancers, particularly under heavy load. Under 80% load, for example, it achieves 1.3-1.4 $\times$ lower mean flow completion time than recent proposals, primarily due to shorter upstream queues. To test hardware feasibility, we implement DRILL in Verilog and estimate

1 INTRODUCTION

Modern datacenter topologies such as Clos networks (Figure 1) are overwhelmingly constructed with large path diversity [4, 17, 19, 41, 42, 50, 51, 57]. A critical issue is the efficient balancing of load among available paths. While ECMP is extensively used in practice [42, 48, 64], it is known to be far from optimal [19, 27, 42]. A study of 10 datacenters, for instance, indicates a significant fraction of core links regularly experience congestion even while there is spare capacity elsewhere [23].

Many recent proposals address this need. Aligned with the recent trend of moving functionality out of the network fabric [28], many proposals strive to delegate load balancing to centralized controllers [18, 25, 32, 59], to the network edge [19], or even to end-hosts [27, 42]. These entities serve as convenient locations for collecting global or end-to-end information about congestion. A notable example is CONGA [19], in which switches at the edge (leafs in Clos networks) gather and analyze congestion feedback from remote leafs and spines to inform forwarding decisions. Planck [61], MicroTE

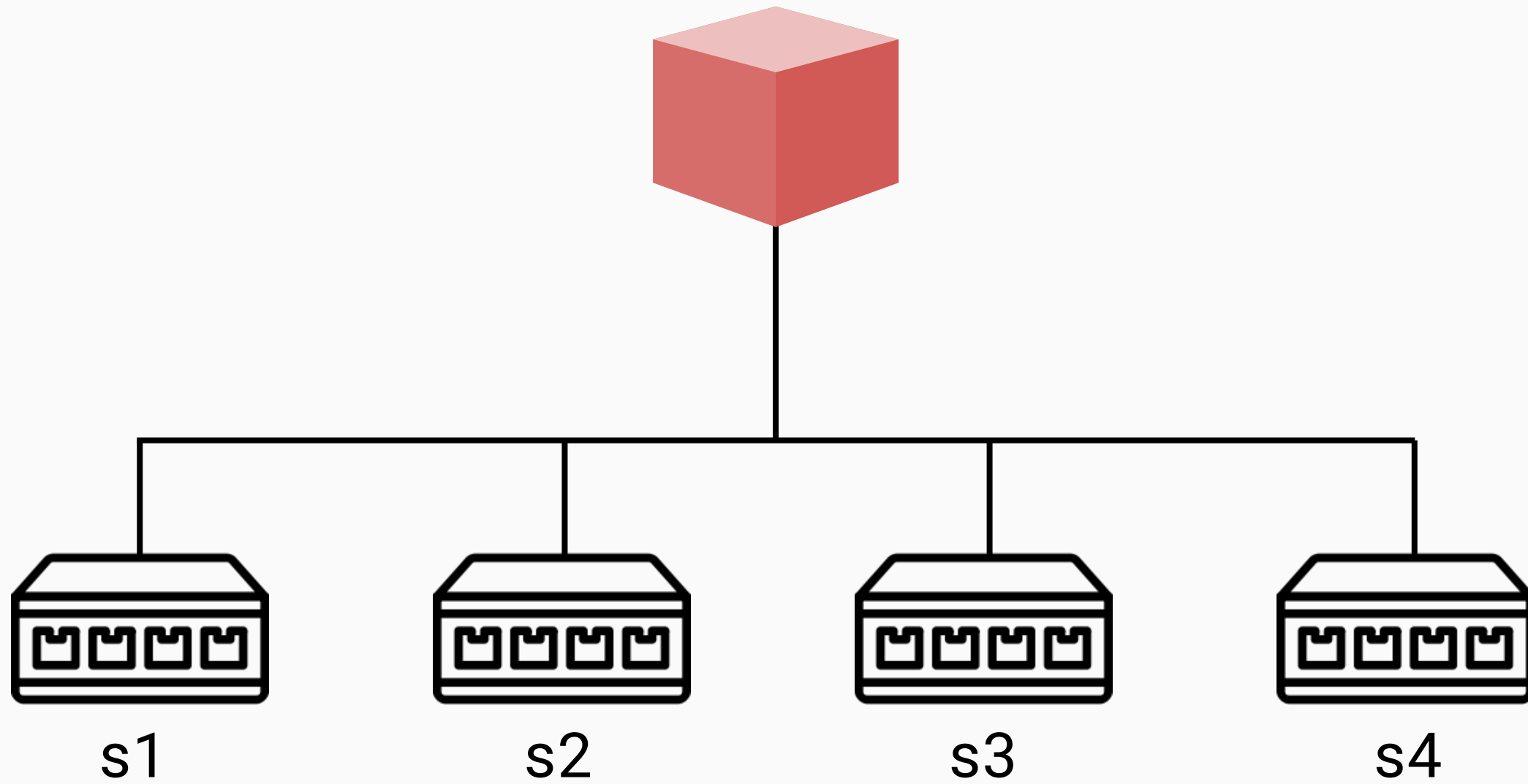
RESULTS:

演算法	Lat p50	Lat p95	Lat p99	Throughput	Util σ
ECMP	108.66 ms	443.84 ms	1185.99 ms	2.53 mbps	0.1180
W-ECMP	79.64 ms	1025.95 ms	2485.76 ms	2.41 mbps	0.0870
Naive DRILL	18.97 ms	159.50 ms	187.78 ms	2.87 mbps	0.1220
ML Hybrid	20.27 ms	141.16 ms	172.24 ms	2.86 mbps	0.0430

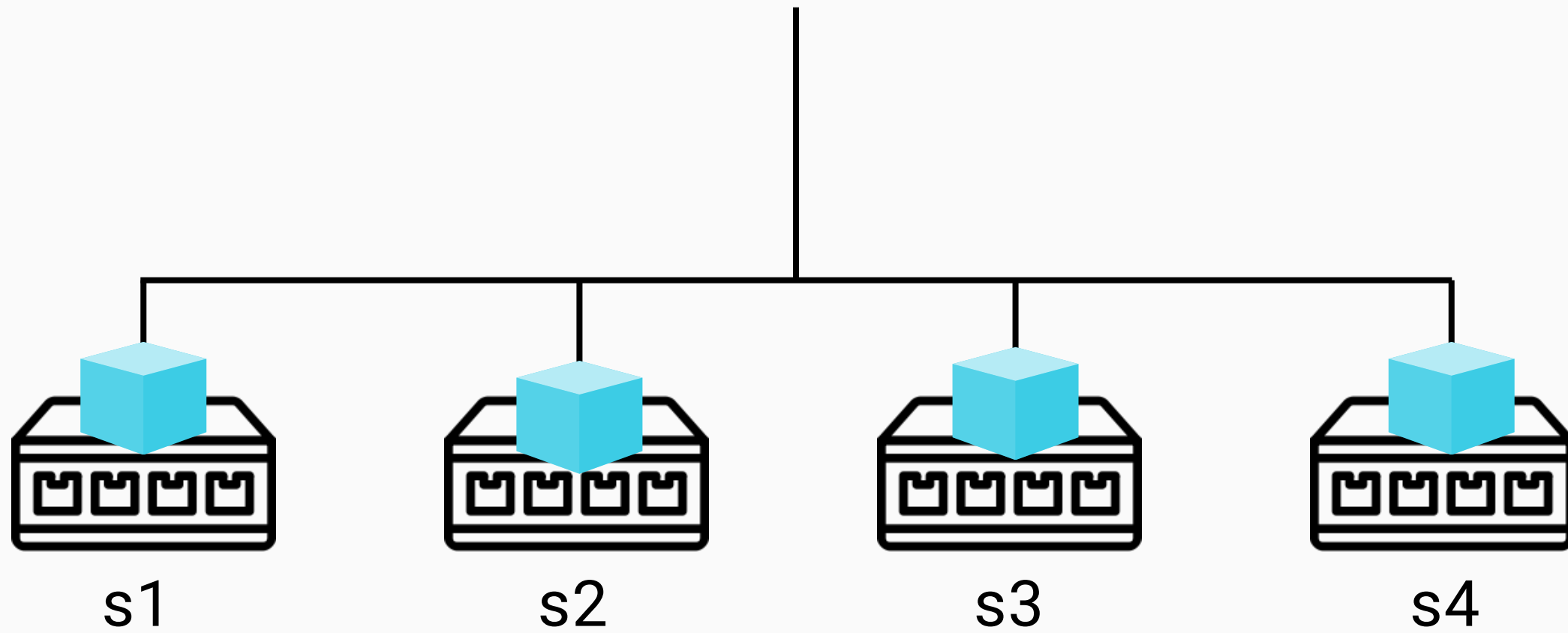
相對DRILL演算法，我們的演算法降成功降低 P99 與 P95 延遲並達到最有效的負載平衡。

Algorithms

Equal Cost Multi-Path (ECMP)



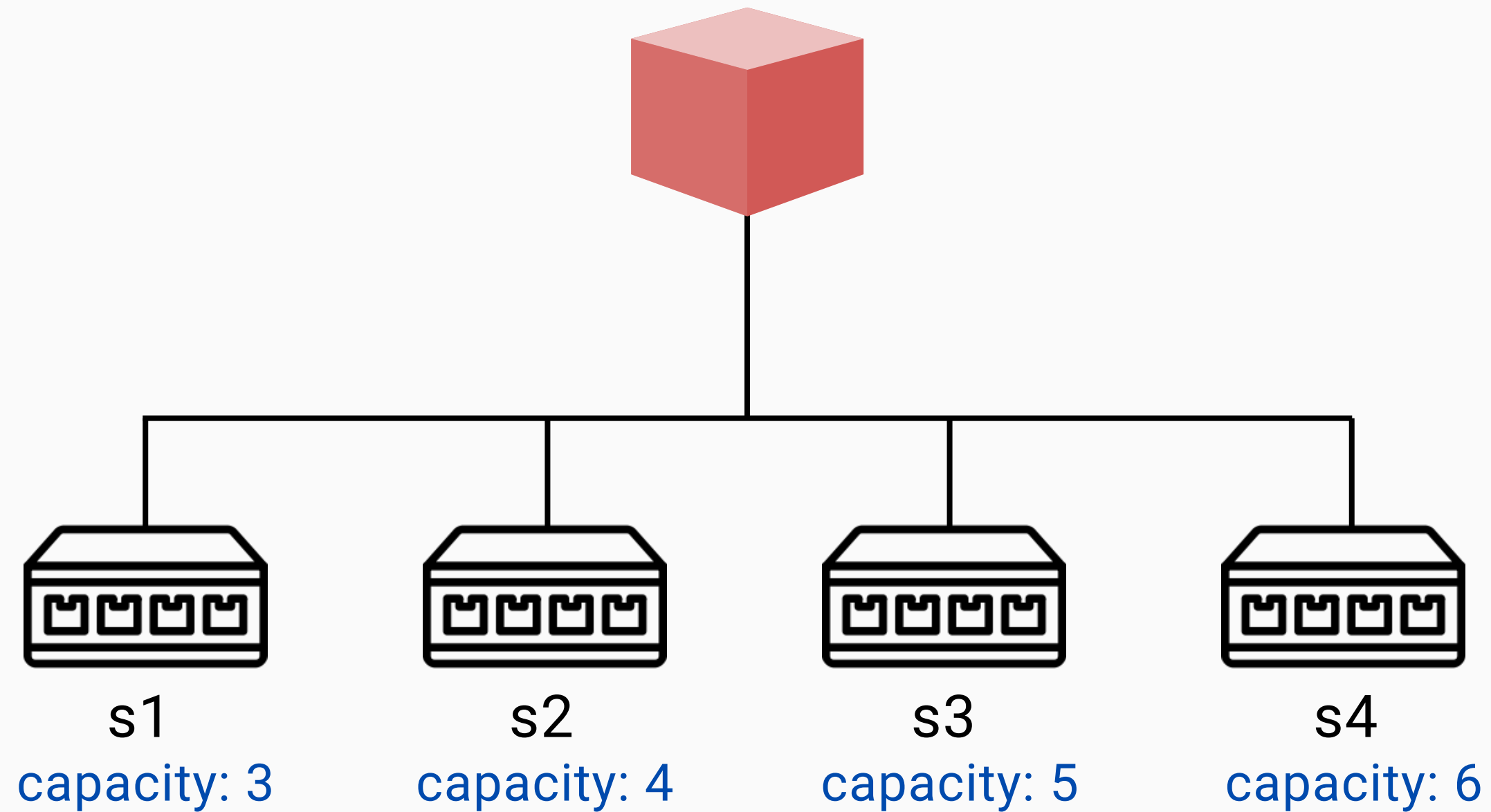
Equal Cost Multi-Path (ECMP)



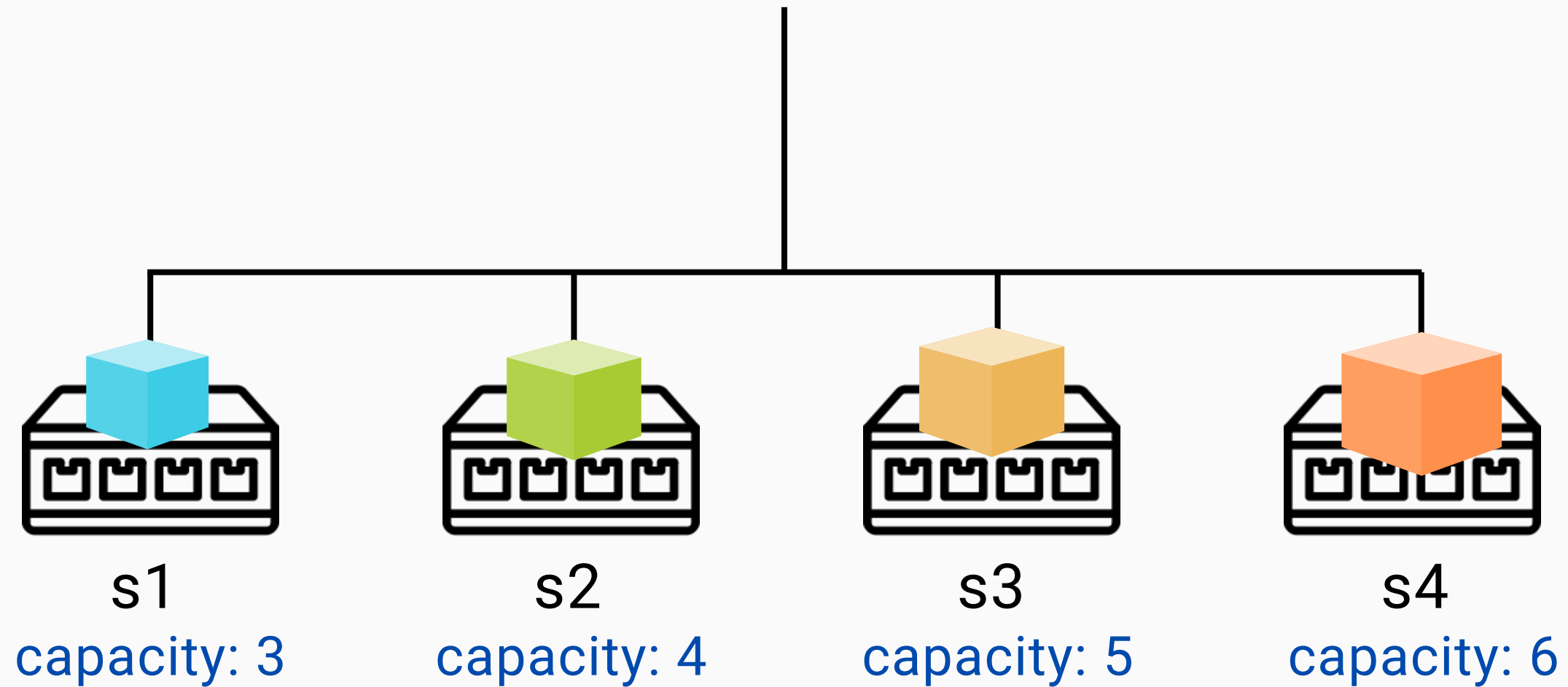


**What if switches have different
capacities?**

Weighted Equal Cost Multi-Path (W-ECMP)



Weighted Equal Cost Multi-Path (W-ECMP)

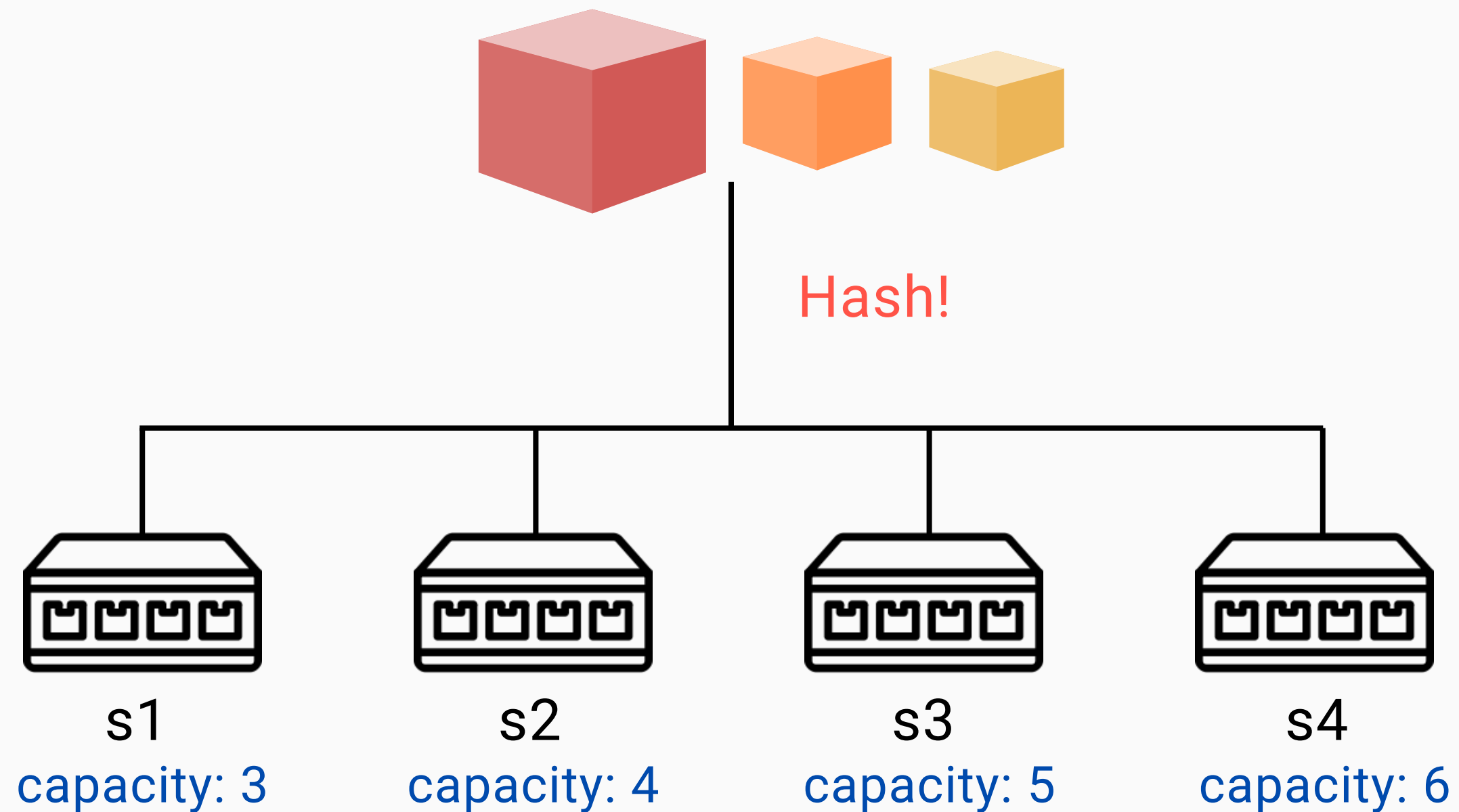




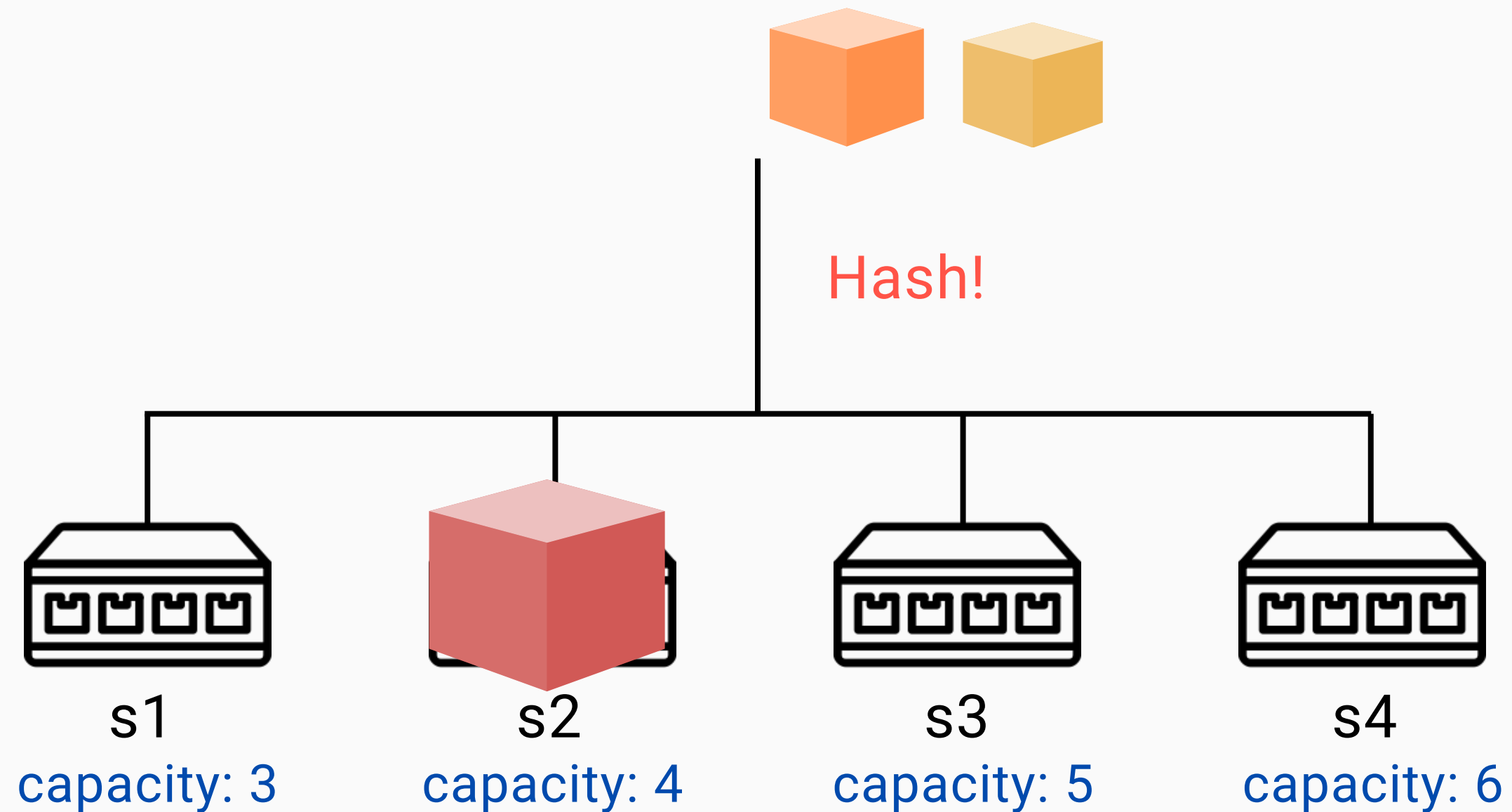
Looks great!

But imagine elephant & mice flows...

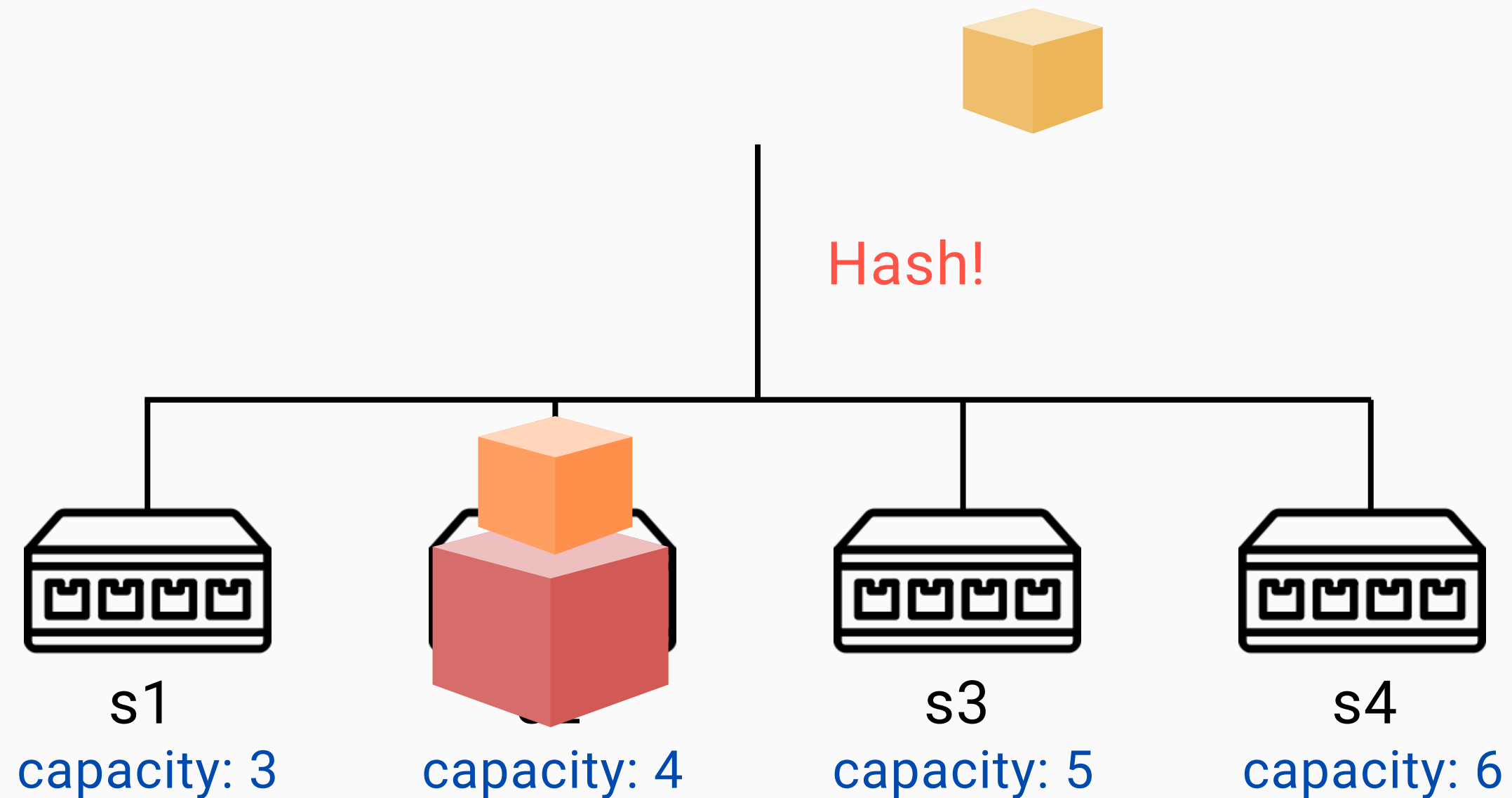
Weighted Equal Cost Multi-Path (W-ECMP)



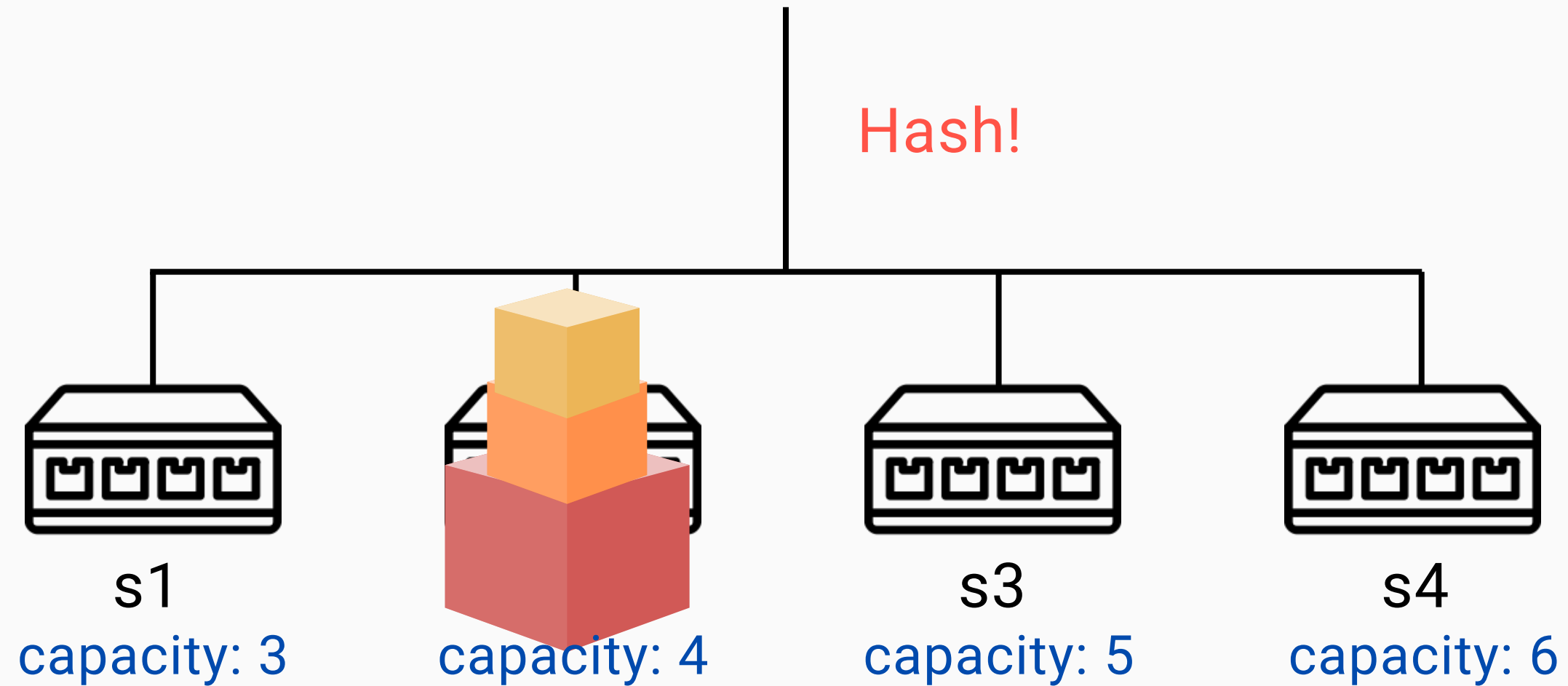
Weighted Equal Cost Multi-Path (W-ECMP)



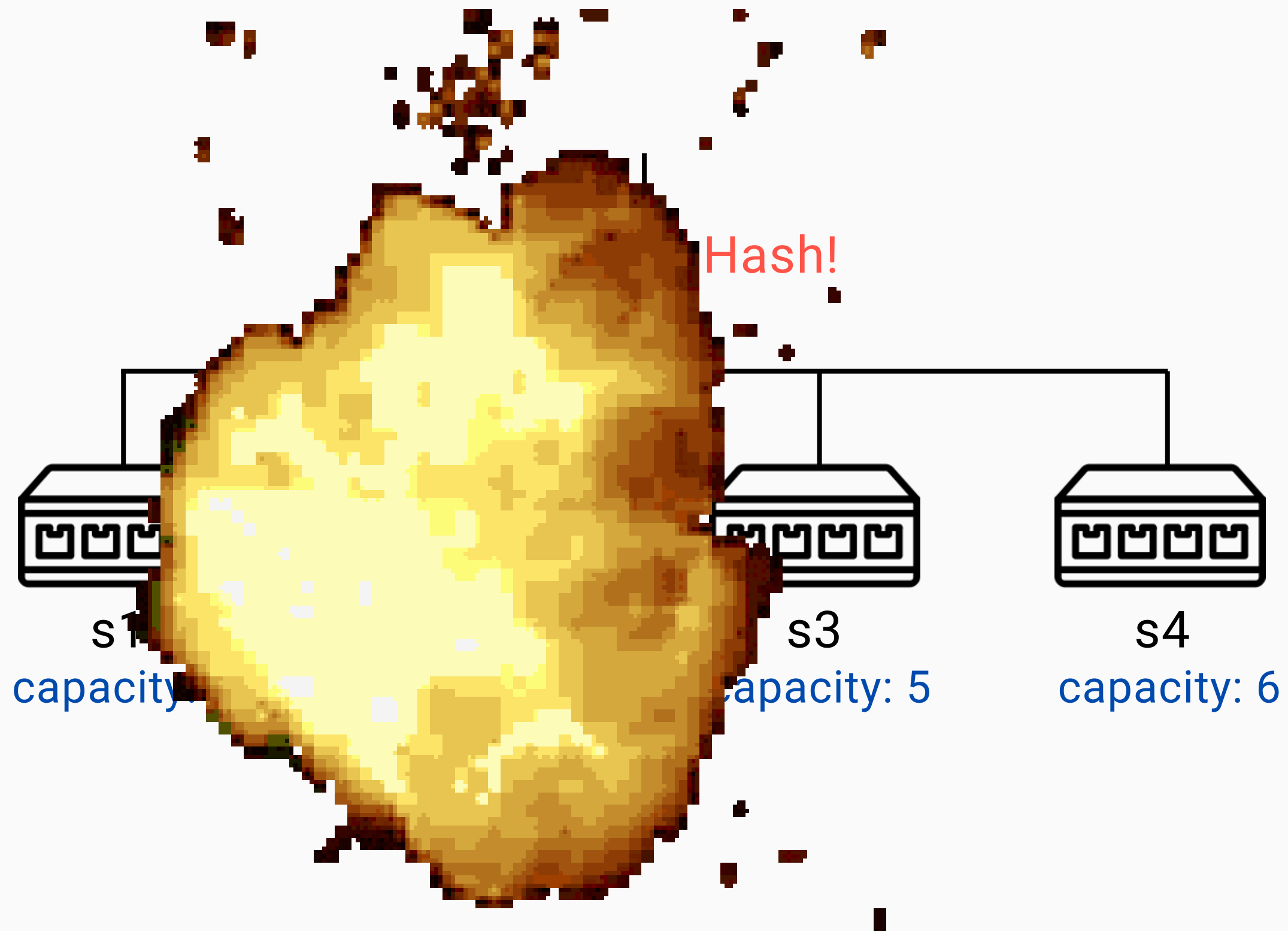
Weighted Equal Cost Multi-Path (W-ECMP)



Weighted Equal Cost Multi-Path (W-ECMP)



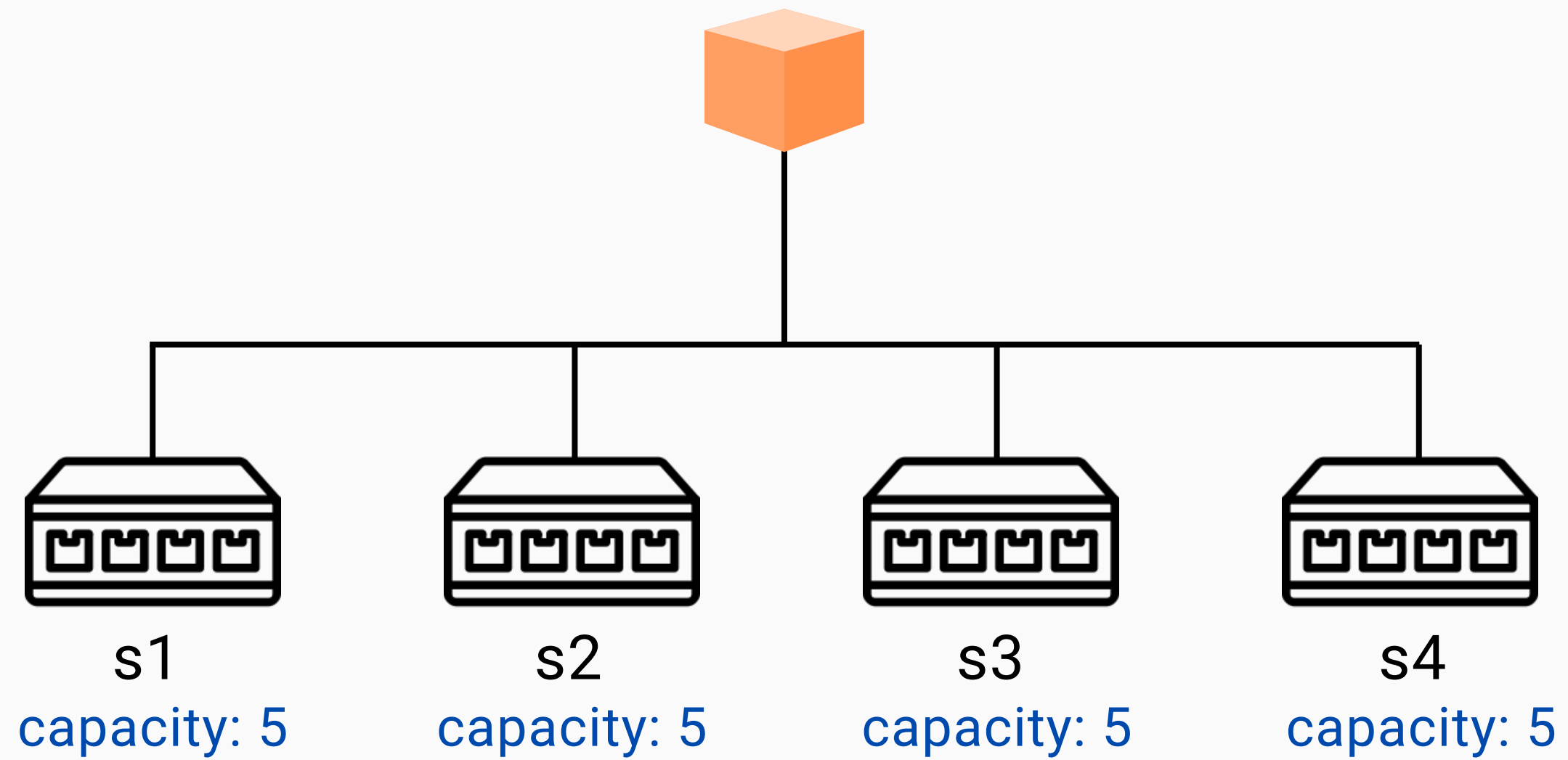
Weighted Equal Cost Multi-Path (W-ECMP)



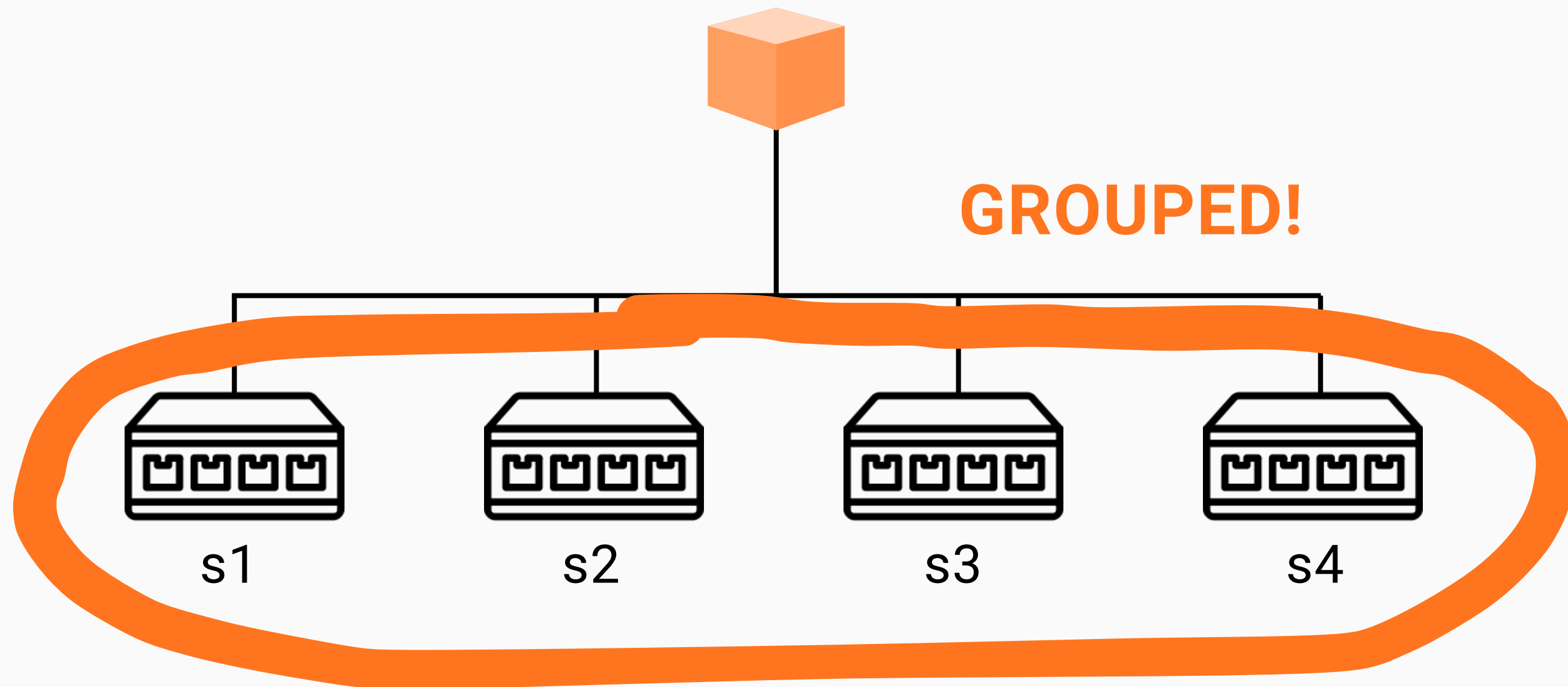


Distributed Randomized In-network Localized Load-balancing (DRILL)

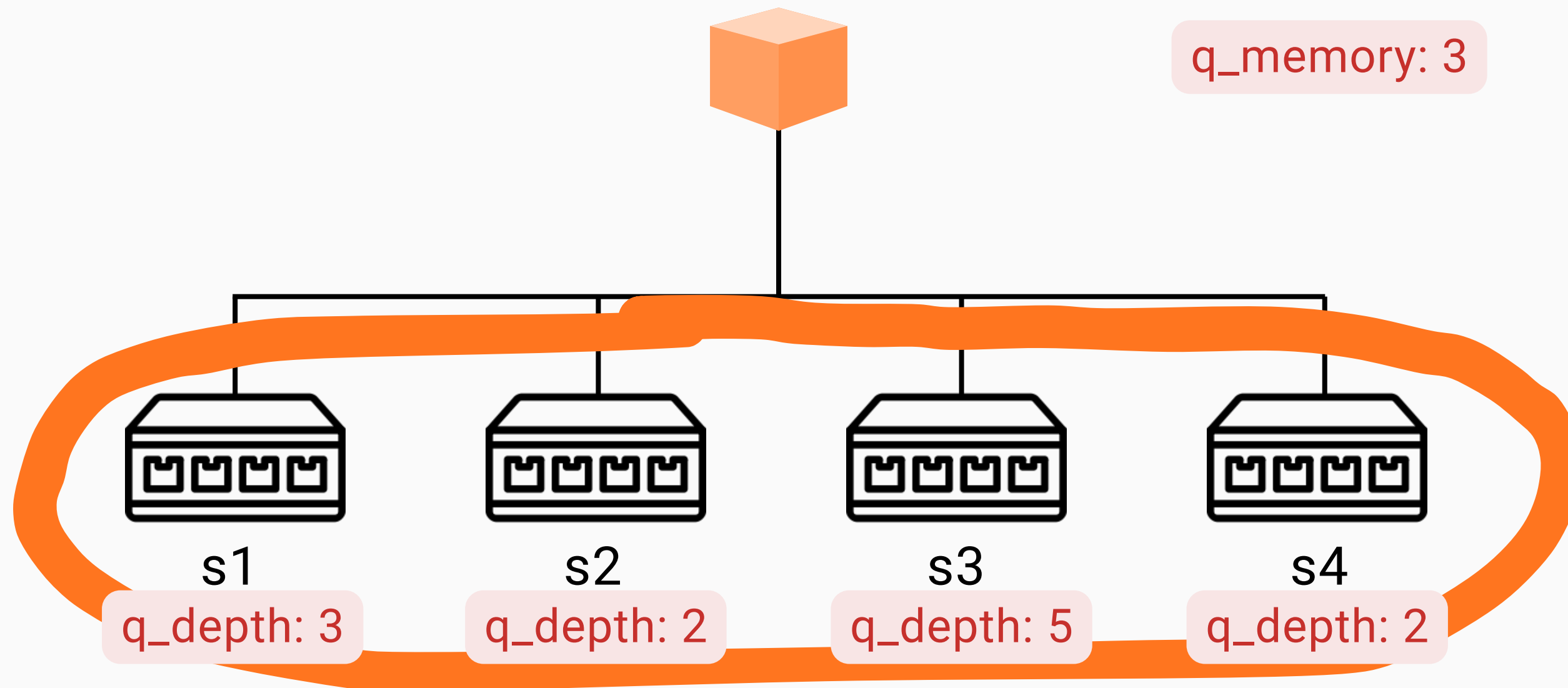
DRILL



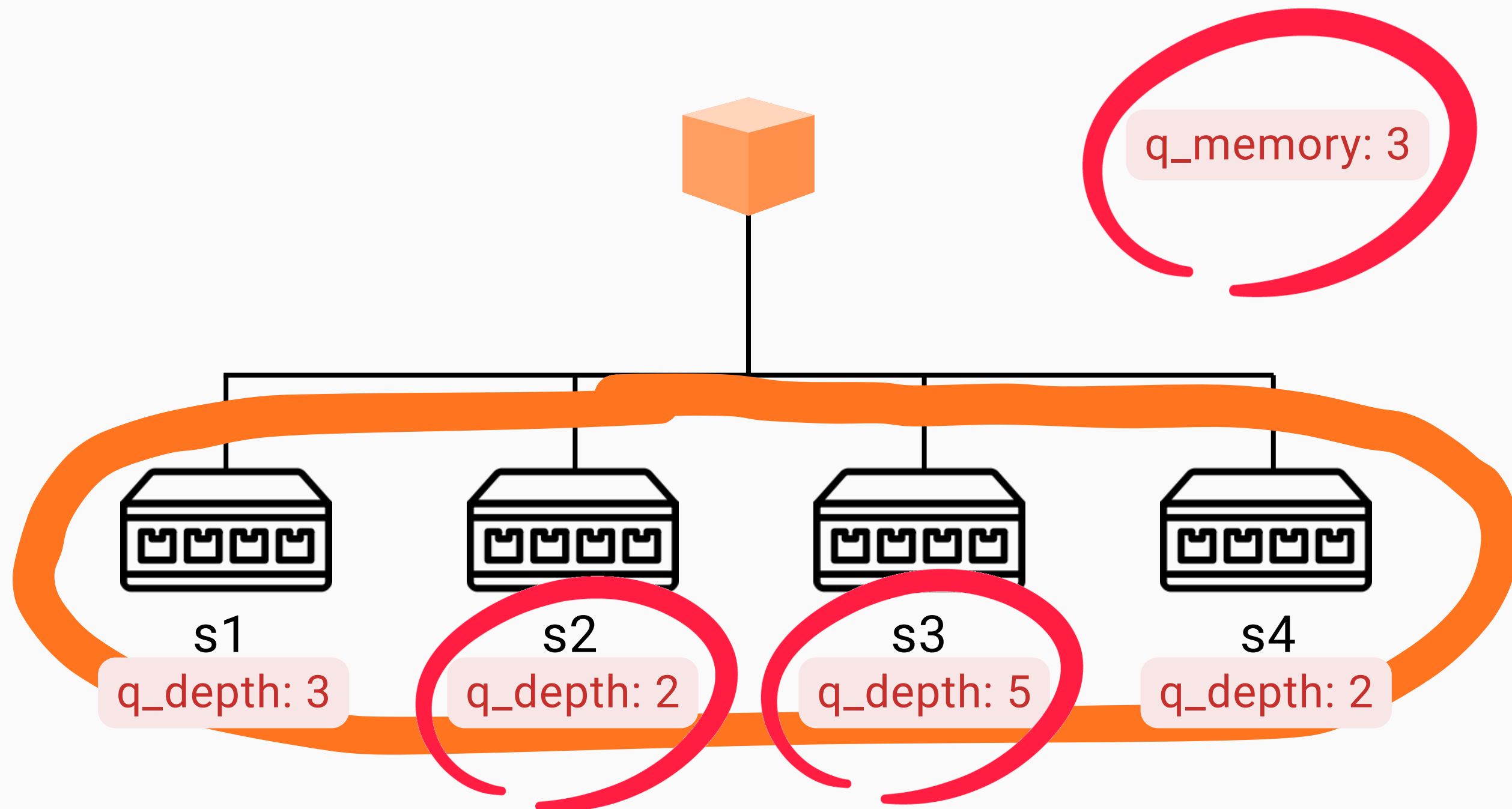
DRILL



DRILL

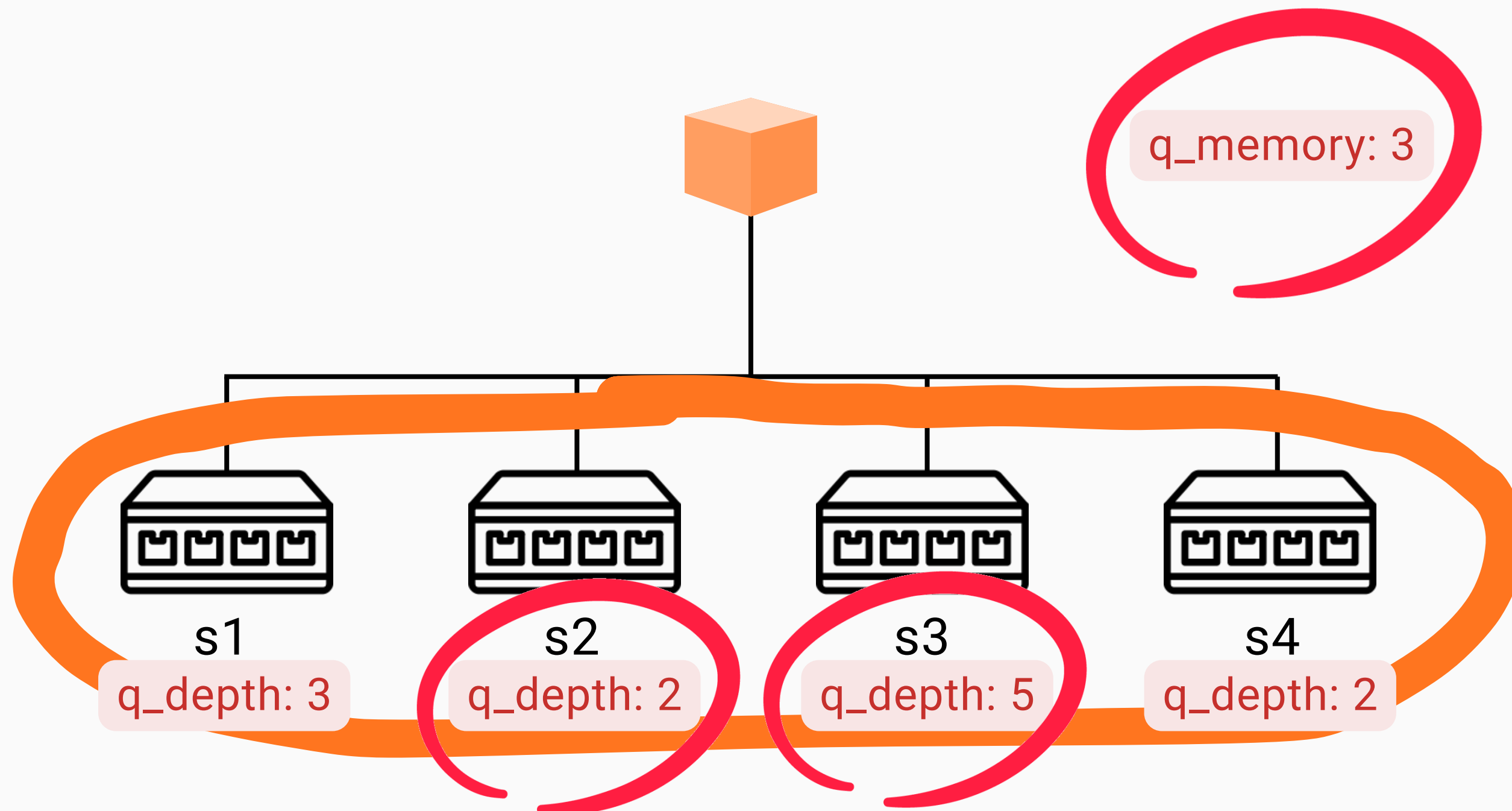


DRILL

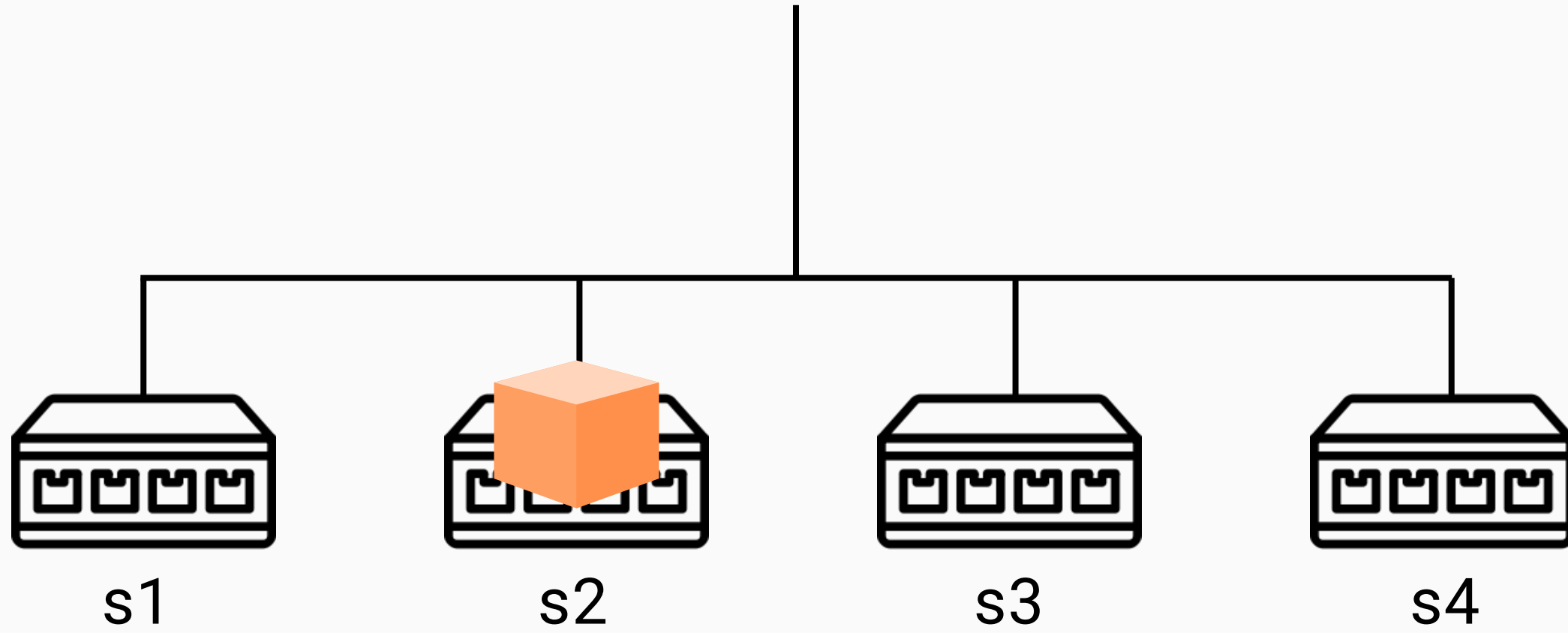


DRILL

- Randomly pick 2 switches → s2 & s3
- $\min(q_memory, s2_q_depth, s3_q_depth)$
= $s2_q_depth = 2$



DRILL



Problem?

Problem?

- 1. DRILL collapses into W-ECMP when the topology is asymmetric.**

Problem?

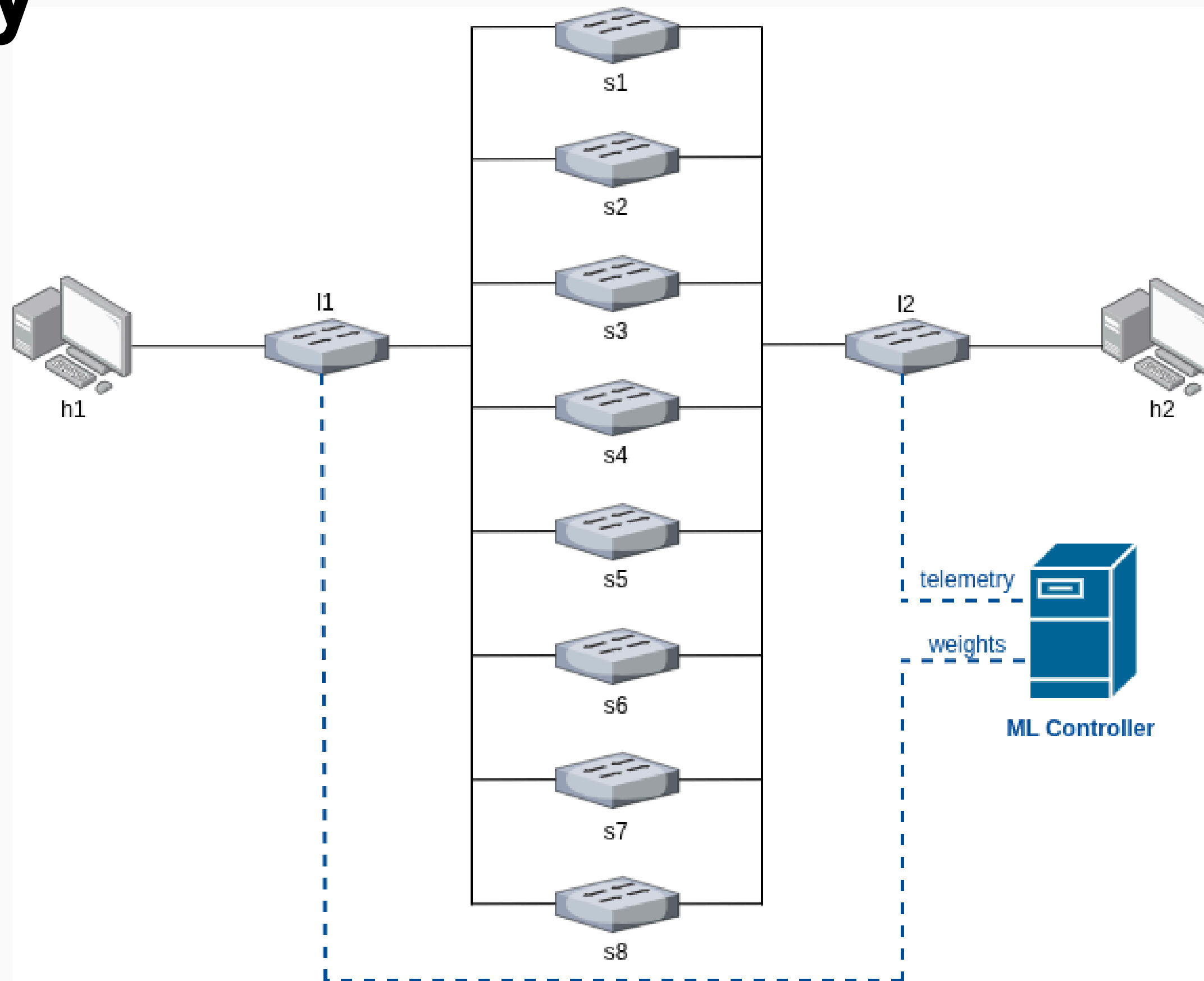
- 1. DRILL collapses into W-ECMP when the topology is asymmetric.**
- 2. If forced into groups, doesn't take into account switch capacities.**

Our Solution:



ML + W-ECMP + DRILL
ML Hybrid

Topology



ML Hybrid Grouping Rule

- Sort spine switches by capacity.

$$s_1 < s_2 < \dots < s_8$$

- Group switches with similar capacity.

$$(s_1, s_2), (s_3, s_4), (s_5, s_6), (s_7, s_8)$$

ML Hybrid Grouping Rule

- Each group becomes one W-ECMP component, group weight is based on total group capacity.

$$\text{base}_g \propto \sum p \in \text{gcapacity}_p$$

- Example base weights:

$$w_g = \text{round} \left(\frac{C_g}{\min_h C_h} \cdot w_{\min} \right) \quad [13 : 17 : 21 : 25] \Rightarrow [3, 4, 5, 6]$$

Machine Learning!

What is Random Forrest?

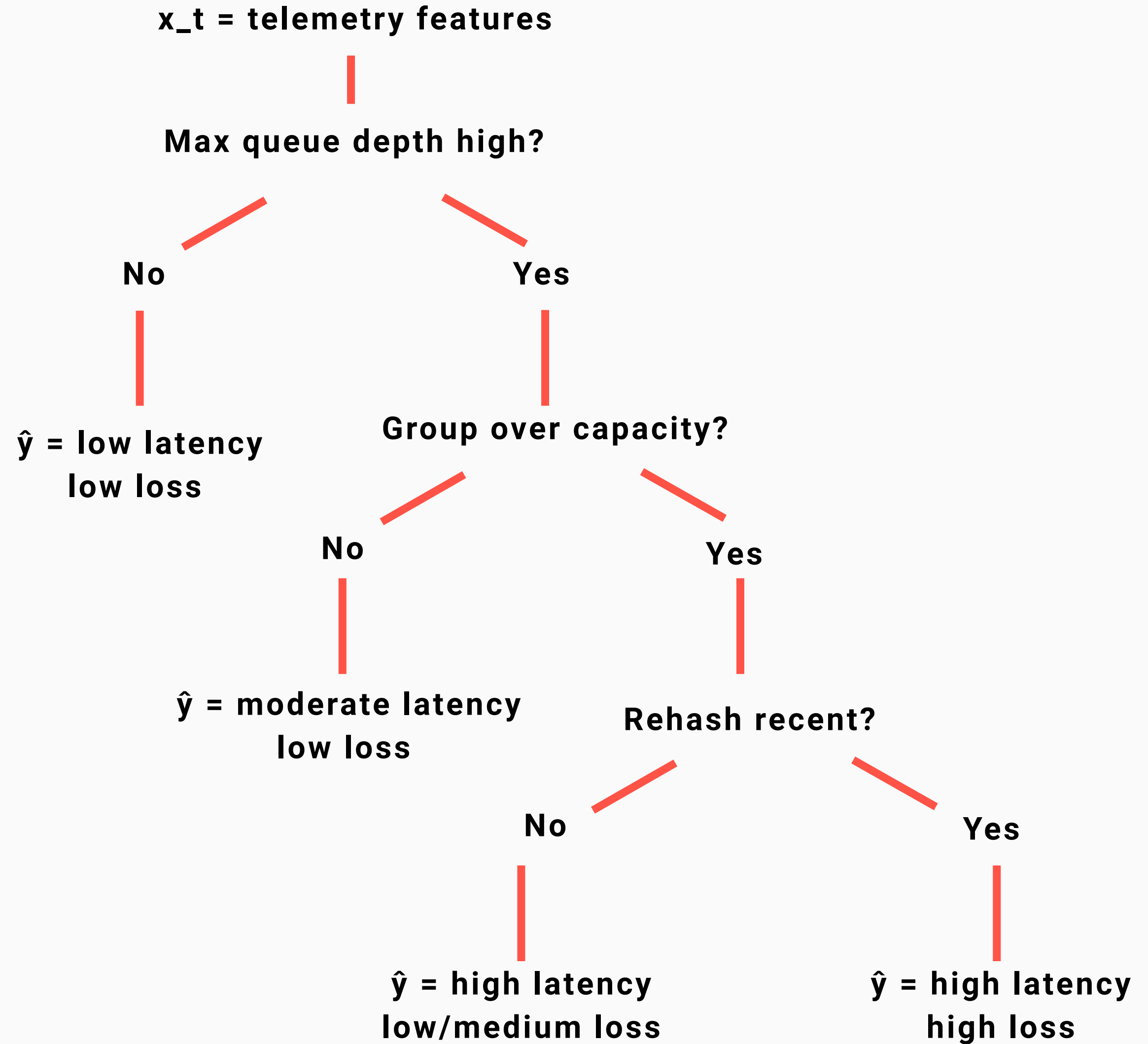
A forrest of Decision trees.

Takes inputs and spits out a prediction.

$$\hat{y}_{t+1} = \frac{1}{N} \sum_{i=1}^N \hat{y}_{i,t+1} = (\widehat{\text{latency}}_{t+1}, \widehat{\text{loss}}_{t+1})$$

Each tree's leaf give an output y.

Decision Tree



Output:

$$\hat{y}_{t+1}(\widehat{\text{latency}}, \widehat{\text{loss}})$$

How was the model trained?

Training

- **Data:** ~31k snapshots, each = 1 second of all 8 spine ports (queue depth, throughput, drop rate, weights)
- **Labels:** real hardware ground truth from P4 registers (INT queue-delay + packet counters)
- **Features:** 39 topology-independent features
 - 18 global aggregates (total utilization, imbalance, over-capacity, queue danger)
 - Top-3 most-congested ports × 7 metrics each → learns "the hottest port," not "port 5"
- **Tuned for a noisy 1-second signal, 100 trees, depth 15, validated with 5-fold cross-validation**

Training

- **5-fold cross-validation**
 - **train 5 temporary models to estimate performance**
 - **80% data, 20% validation**
- **final training**
 - **train 1 final model on all data for actual use**
 - **100% data, validated with an avg of 5 temp models**

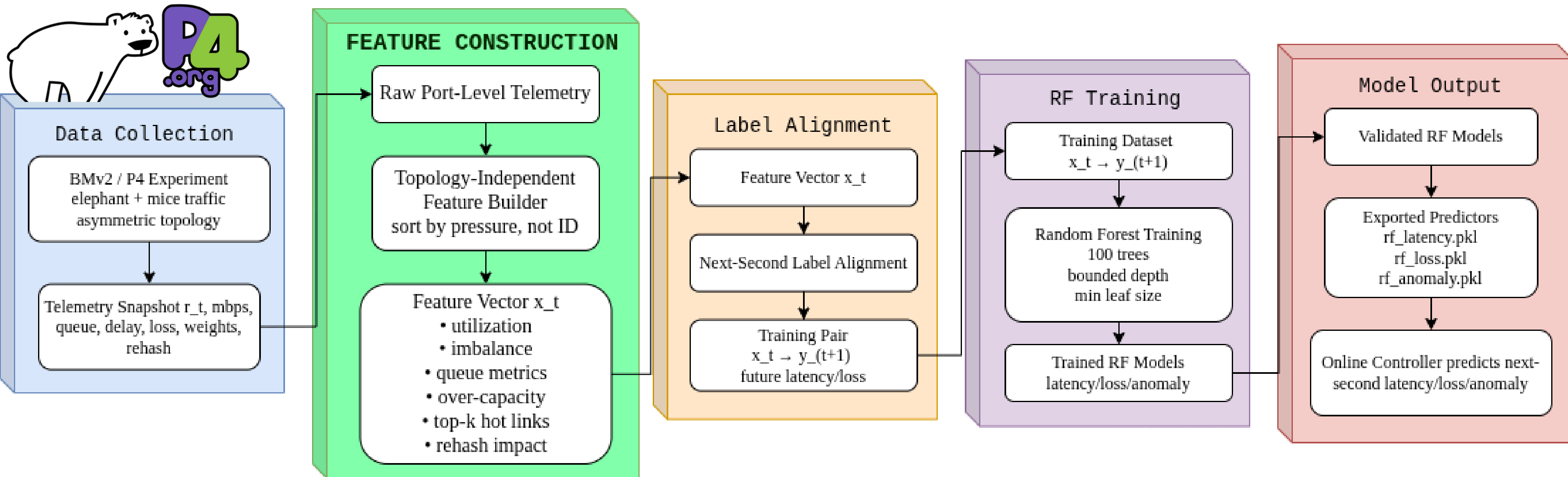
39 Features? 🤯

#	Feature	Meaning	18 global aggregates		
1	Is_Rehash_Event	did weights just change this tick?	#	Feature	Meaning
2	Time_Since_Last_Rehash_s	seconds since last weight change	10	Over_Capacity_Sum	total Mbps exceeding capacity (drives loss)
3	Rehash_Impact	$e^{-\text{time since rehash}}$	11	Max_Q_Ratio	deepest queue ÷ 64 (fullness)
4	Total_Util_Sum	summed utilization across all ports	12	Q_Danger_Flag	1 if any queue > 40
5	Max_Util_Diff	max-min utilization (imbalance)	13	Q_Danger_Count	how many ports have queue > 40
6	Group_Imbalance	load/weight gap between the two port halves	14	Total_QDepth_Trend	is total queue growing?
7	Max_QDepth	deepest queue anywhere	15	Total_Actual_Mbps	total throughput
8	Total_QDepth	summed queue depth	16	Expected_Over_Capacity_Sum	projected over-capacity given weights
9	QDepth_Imbalance	max-min queue depth	17	Overflow_Intensity	$\text{Over_Capacity_Sum} \times \text{Max_Q_Ratio}$
			18	Queue_Full_And_Over_Cap	$\text{Over_Capacity_Sum} \times \text{Q_Danger_Flag}$

21 Top-K features (3 hottest ports × 7 metrics)

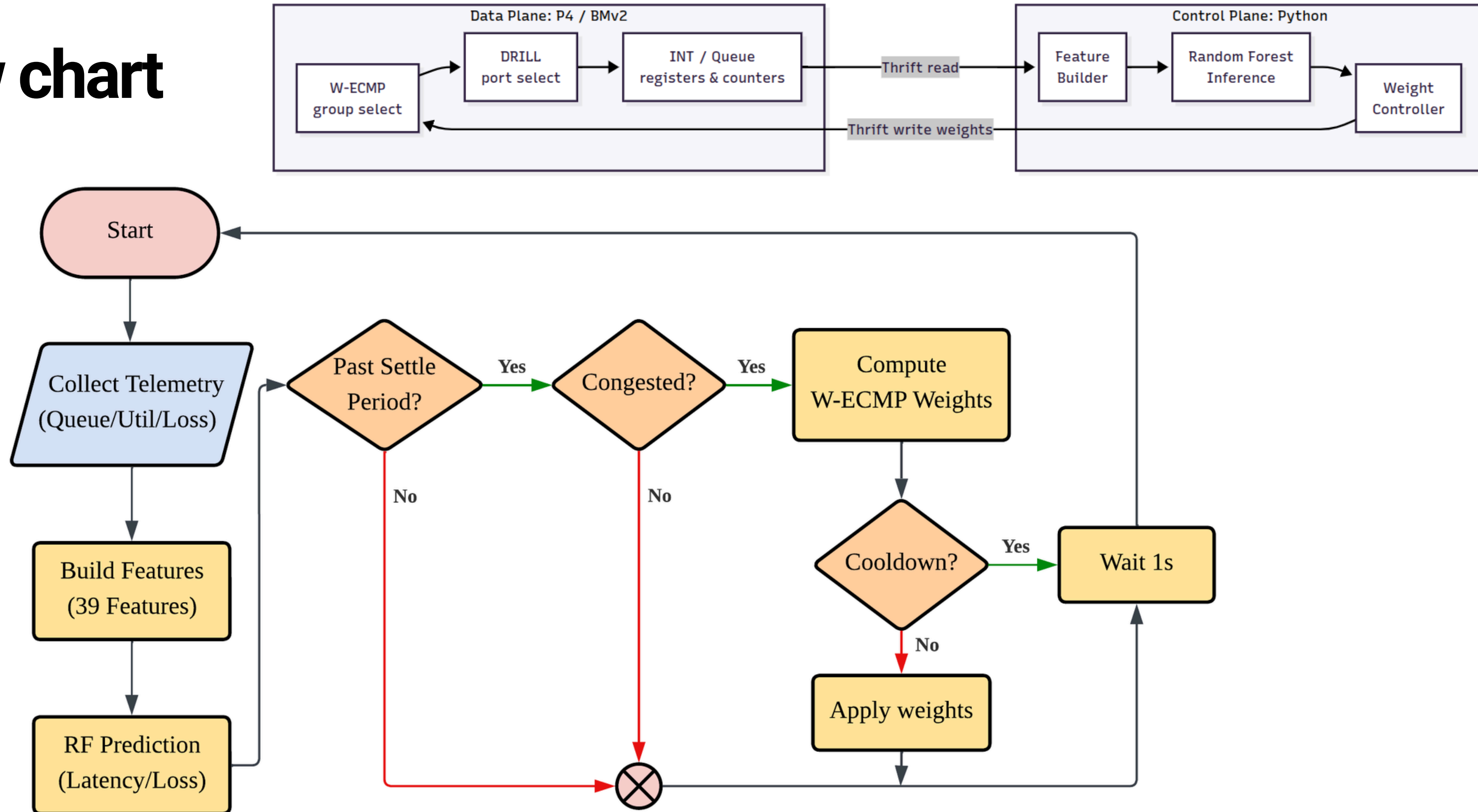
#	Feature	#	Feature	#	Feature
19	top1_qdepth	26	top2_qdepth	33	top3_qdepth
20	top1_mbps	27	top2_mbps	34	top3_mbps
21	top1_weight	28	top2_weight	35	top3_weight
22	top1_norm_load	29	top2_norm_load	36	top3_norm_load
23	top1_expected_util	30	top2_expected_util	37	top3_expected_util
24	top1_qdepth_trend	31	top2_qdepth_trend	38	top3_qdepth_trend
25	top1_mbps_trend	32	top2_mbps_trend	39	top3_mbps_trend

Procedure



**ML predicts, weight
changes correspondingly**

Flow chart



What is P4?

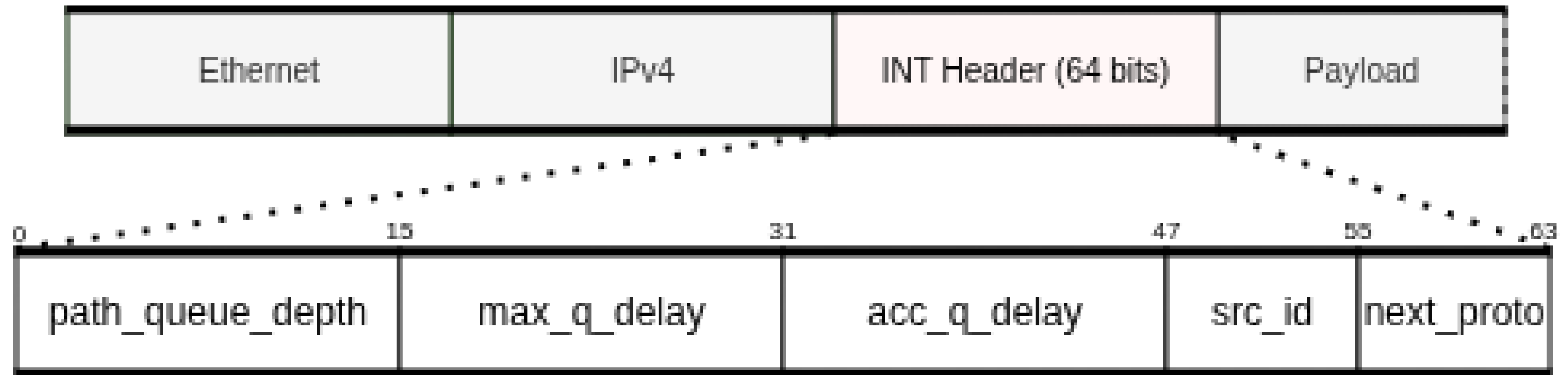


P4

- **P4 (Programming Protocol-independent Packet Processors)** is a language for programming the data plane of a switch
- **It defines:**
 - Parser → reads raw bytes off the wire into structured headers (Ethernet, IPv4, and the **custom** int_hdr).
 - MyIngress → routing, W-ECMP, DRILL, INT insertion
 - MyEgress → records real queue depth/delay
 - Deparser → serializes headers back onto the wire.

P4 Collecting Telemetry

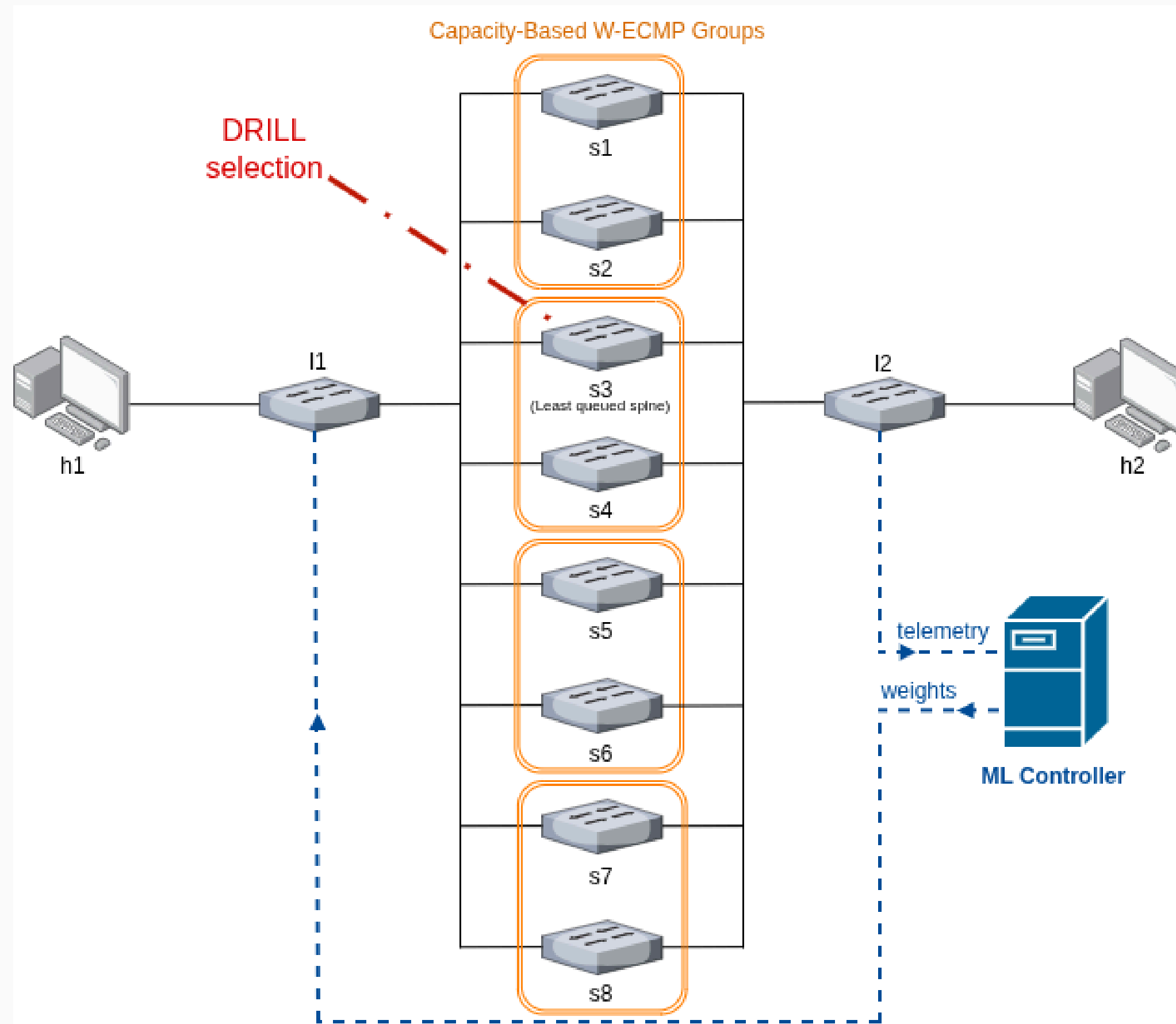
- Instead of polling each switch separately, we let real packets carry a logbook that gets stamped at every hop



- true end-to-end delay that trains our model

ML Hybrid in action!

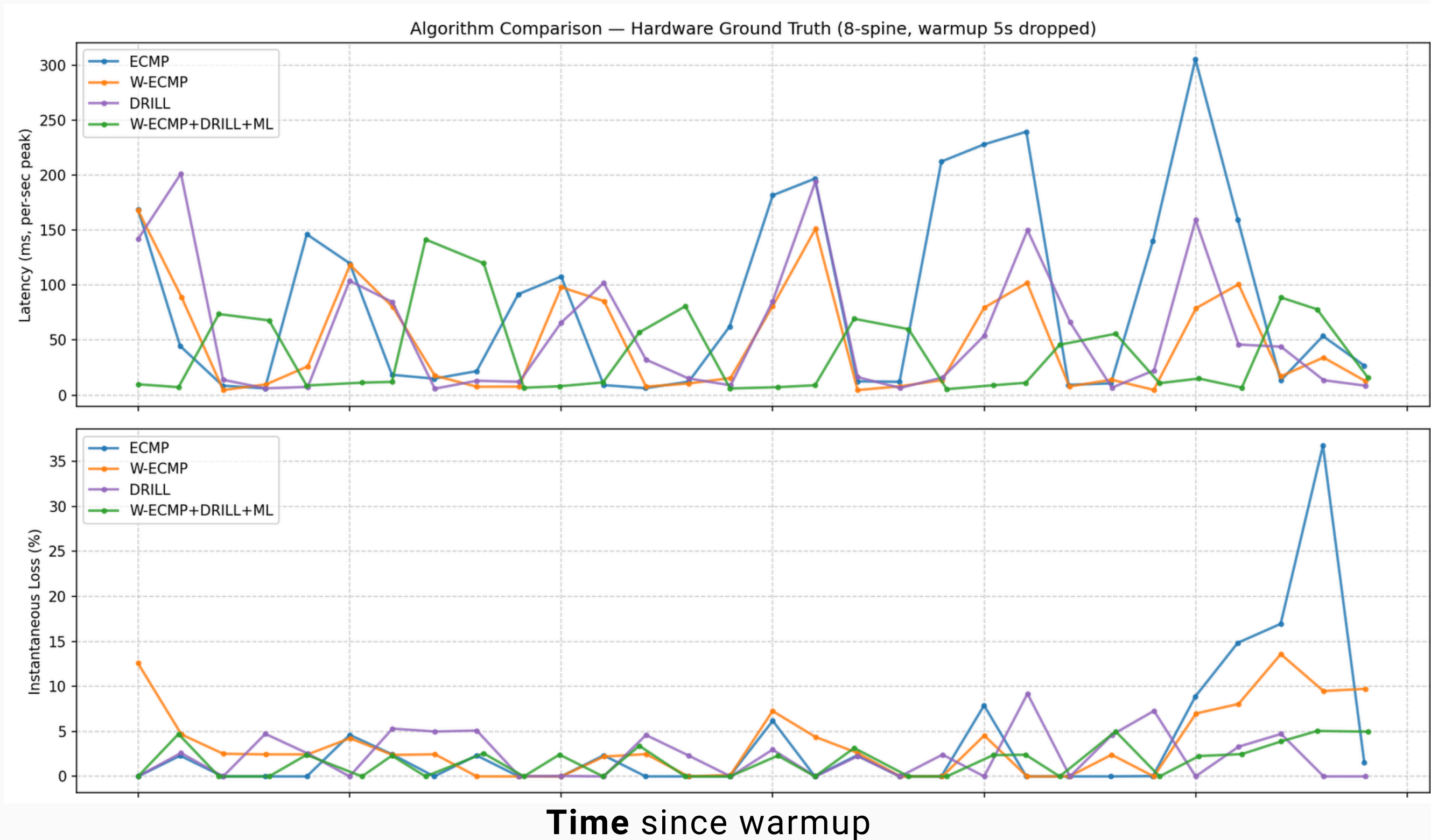
ML Hybrid in action!



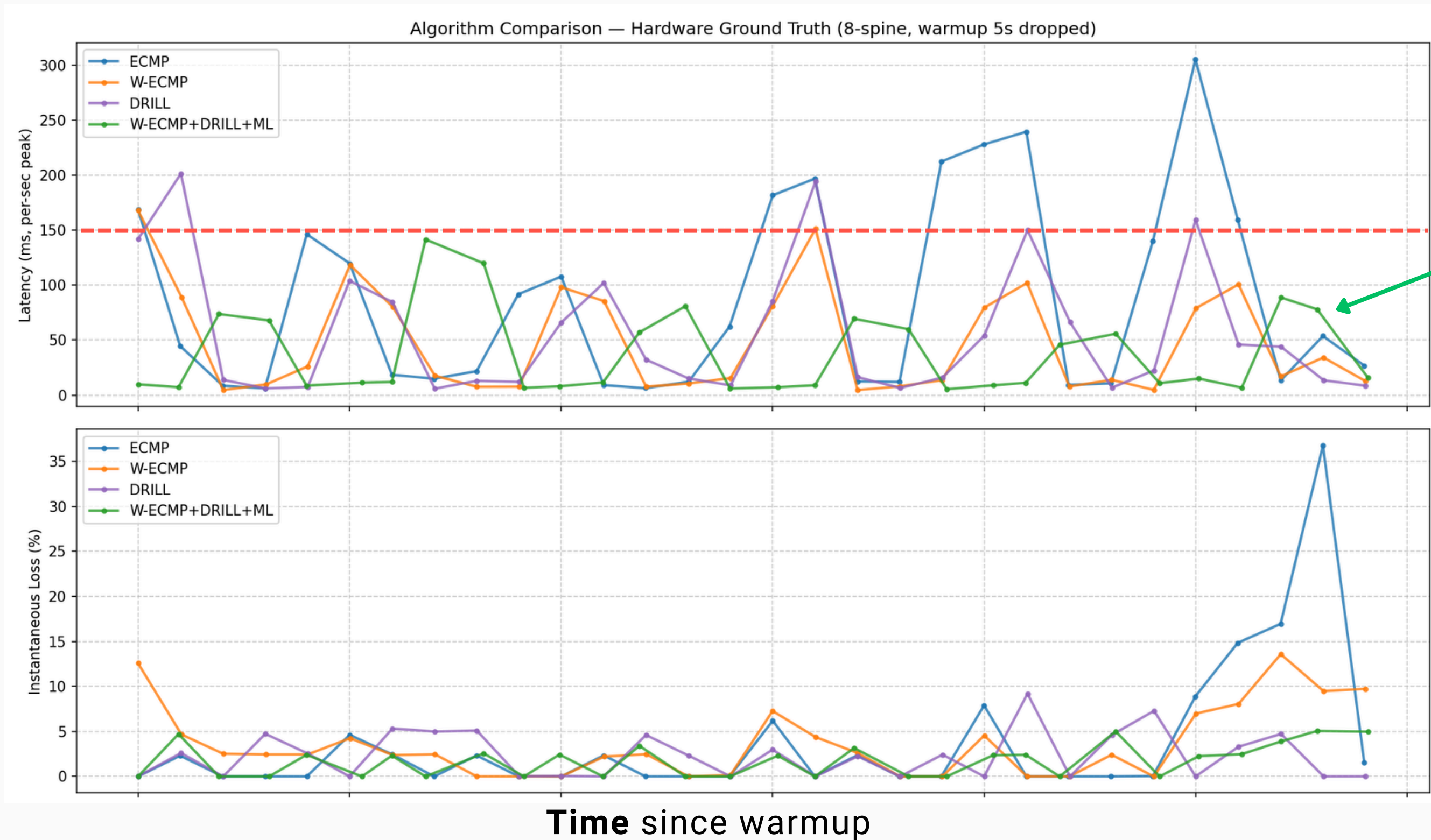
Analyzing results



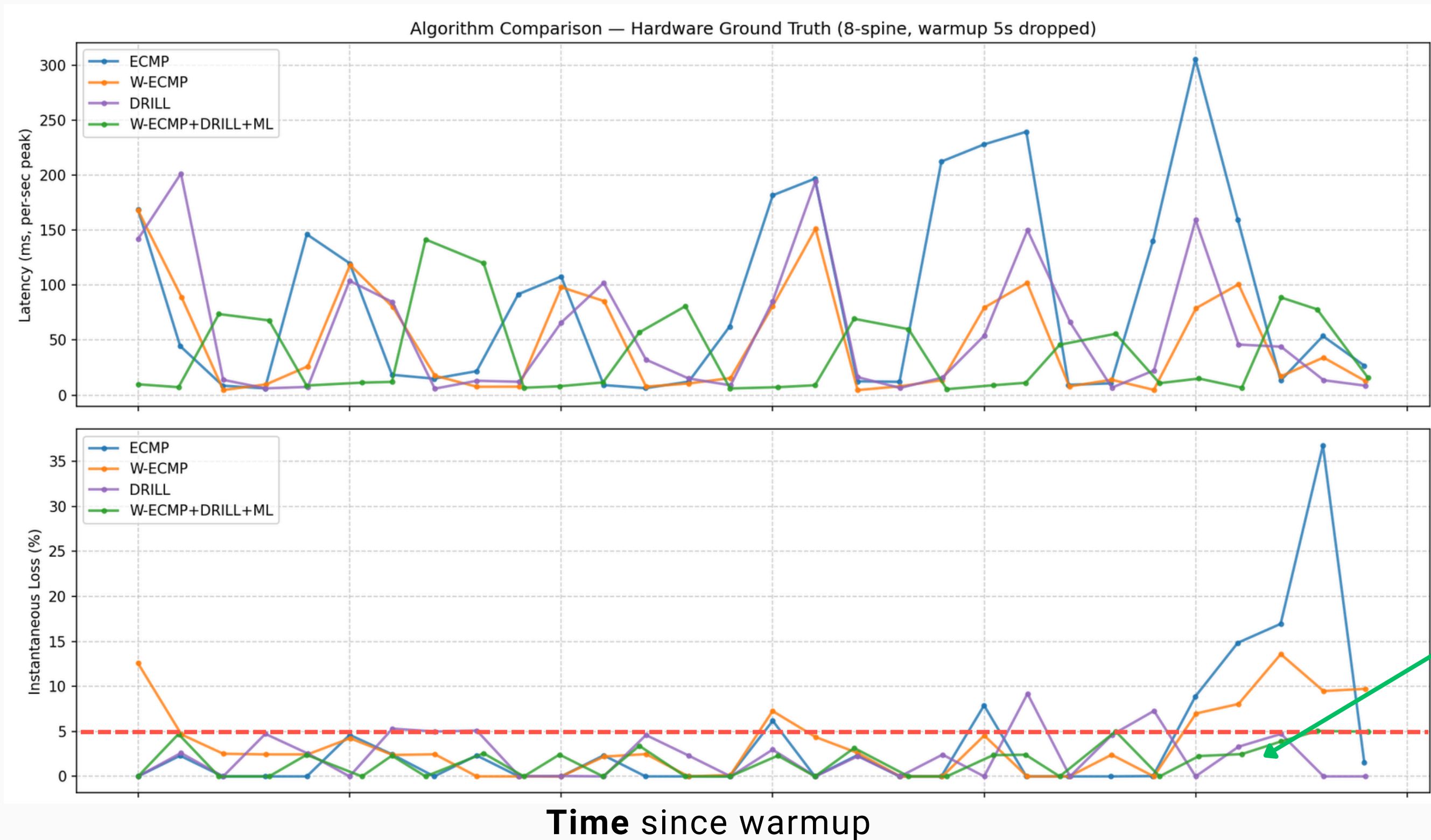
Result (Latency / Loss of 30 sec)



Result (Latency / Loss of 30 sec)

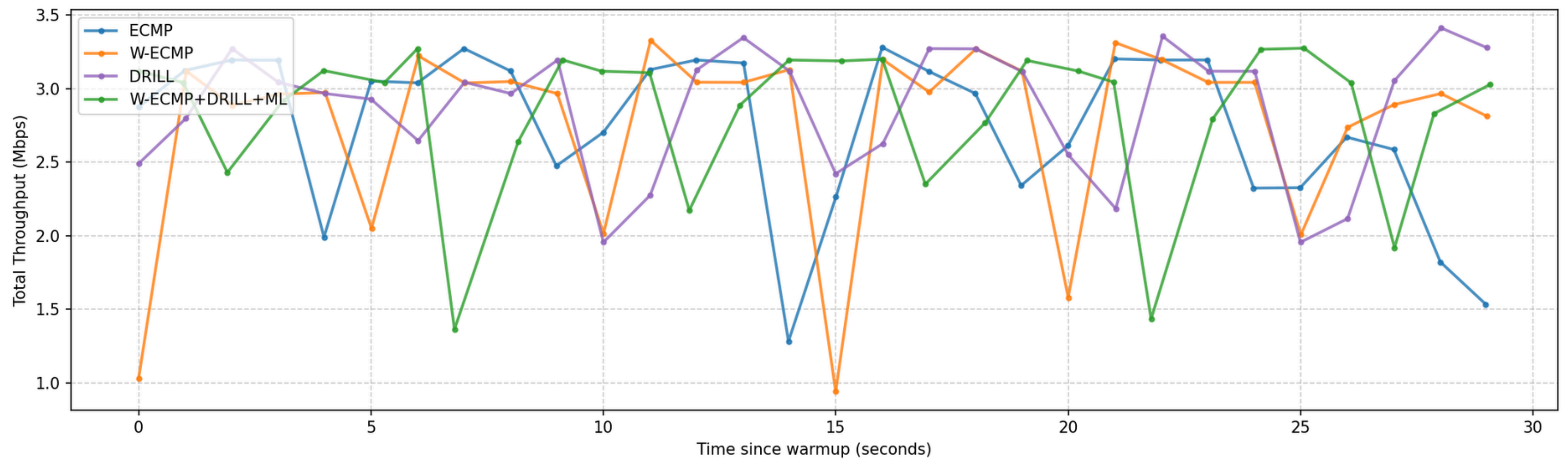


Result (Latency / Loss of 30 sec)

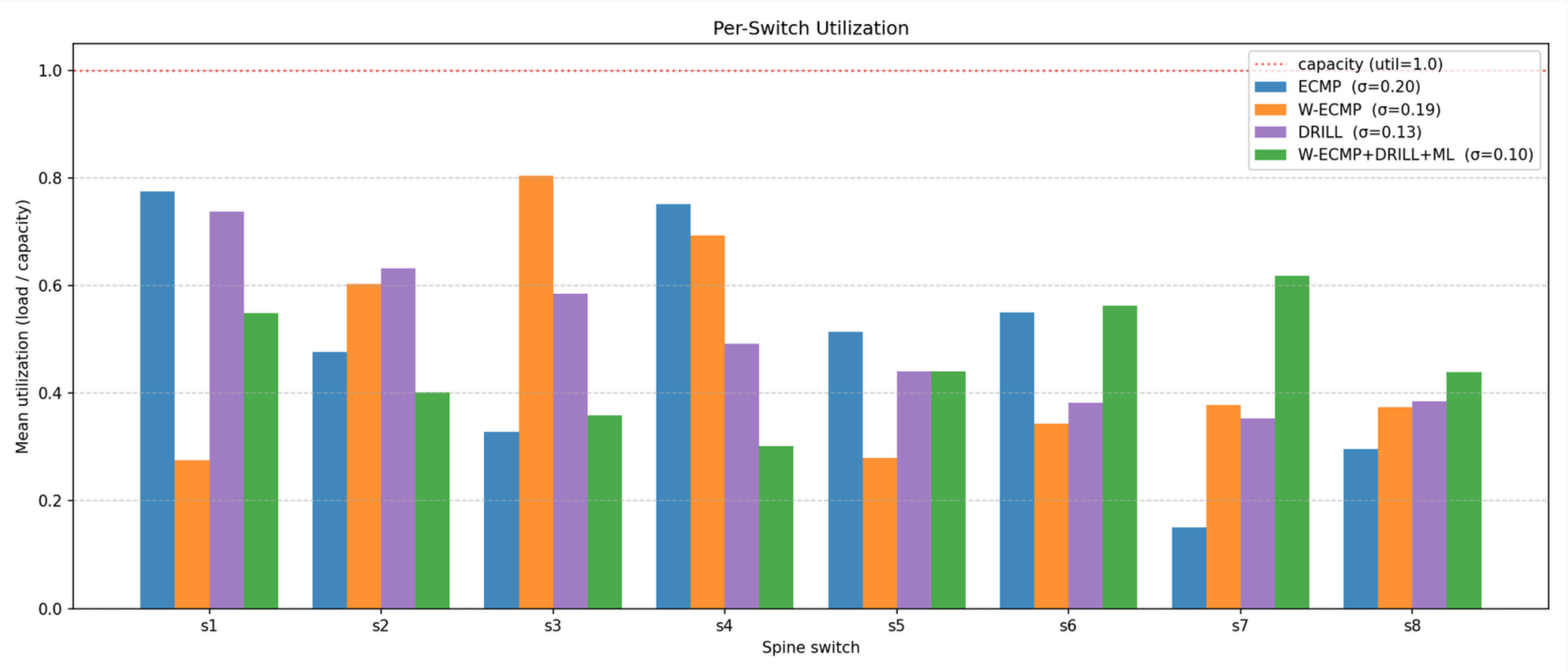


ML Hybrid :
< 5 %

Result (Throughput of 30 sec)



Result (Per Switch Utilization of 30 sec)



演算法	Util σ
ECMP	0.20
W-ECMP	0.19
Naive DRILL	0.13
ML Hybrid	0.10

Result (300 sec)



演算法	Lat p50	Lat p95	Lat p99	Throughput	Util σ
ECMP	108.66 ms	443.84 ms	1185.99 ms	2.53 mbps	0.1180
W-ECMP	79.64 ms	1025.95 ms	2485.76 ms	2.41 mbps	0.0870
Naive DRILL	18.97 ms	159.50 ms	187.78 ms	2.87 mbps	0.1220
ML Hybrid	20.27 ms	141.16 ms	172.24 ms	2.86 mbps	0.0430

Result (300 sec)

演算法	Lat p50	Lat p95	Lat p99	Throughput	Util σ
ECMP	108.66 ms	443.84 ms	1185.99 ms	2.53 mbps	0.1180
W-ECMP	79.64 ms	1025.95 ms	2485.76 ms	2.41 mbps	0.0870
Naive DRILL	18.97 ms	159.50 ms	187.78 ms	2.87 mbps	0.1220
ML Hybrid	20.27 ms	141.16 ms	172.24 ms	2.86 mbps	0.0430

5.2x faster
than ECMP

3.1x faster
than ECMP

6.9x faster
than ECMP

Result (300 sec)

演算法	Lat p50	Lat p95	Lat p99	Throughput	Util σ
ECMP	108.66 ms	443.84 ms	1185.99 ms	2.53 mbps	0.1180
W-ECMP	79.64 ms	1025.95 ms	2485.76 ms	2.41 mbps	0.0870
Naive DRILL	18.97 ms	159.50 ms	187.78 ms	2.87 mbps	0.1220
ML Hybrid	20.27 ms	141.16 ms	172.24 ms	2.86 mbps	0.0430

2.74x better
than ECMP

Result (300 sec)

演算法	Lat p50	Lat p95	Lat p99	Throughput	Util σ
ECMP	108.66 ms	443.84 ms	1185.99 ms	2.53 mbps	0.1180
W-ECMP	79.64 ms	1025.95 ms	2485.76 ms	2.41 mbps	0.0870
Naive DRILL	18.97 ms	159.50 ms	187.78 ms	2.87 mbps	0.1220
ML Hybrid	20.27 ms	141.16 ms	172.24 ms	2.86 mbps	0.0430

11.5% faster
than DRILL

9% faster
than DRILL

Result (300 sec)

演算法	Lat p50	Lat p95	Lat p99	Throughput	Util σ
ECMP	108.66 ms	443.84 ms	1185.99 ms	2.53 mbps	0.1180
W-ECMP	79.64 ms	1025.95 ms	2485.76 ms	2.41 mbps	0.0870
Naive DRILL	18.97 ms	159.50 ms	187.78 ms	2.87 mbps	0.1220
ML Hybrid	20.27 ms	141.16 ms	172.24 ms	2.86 mbps	0.0430

2.83x better
than DRILL

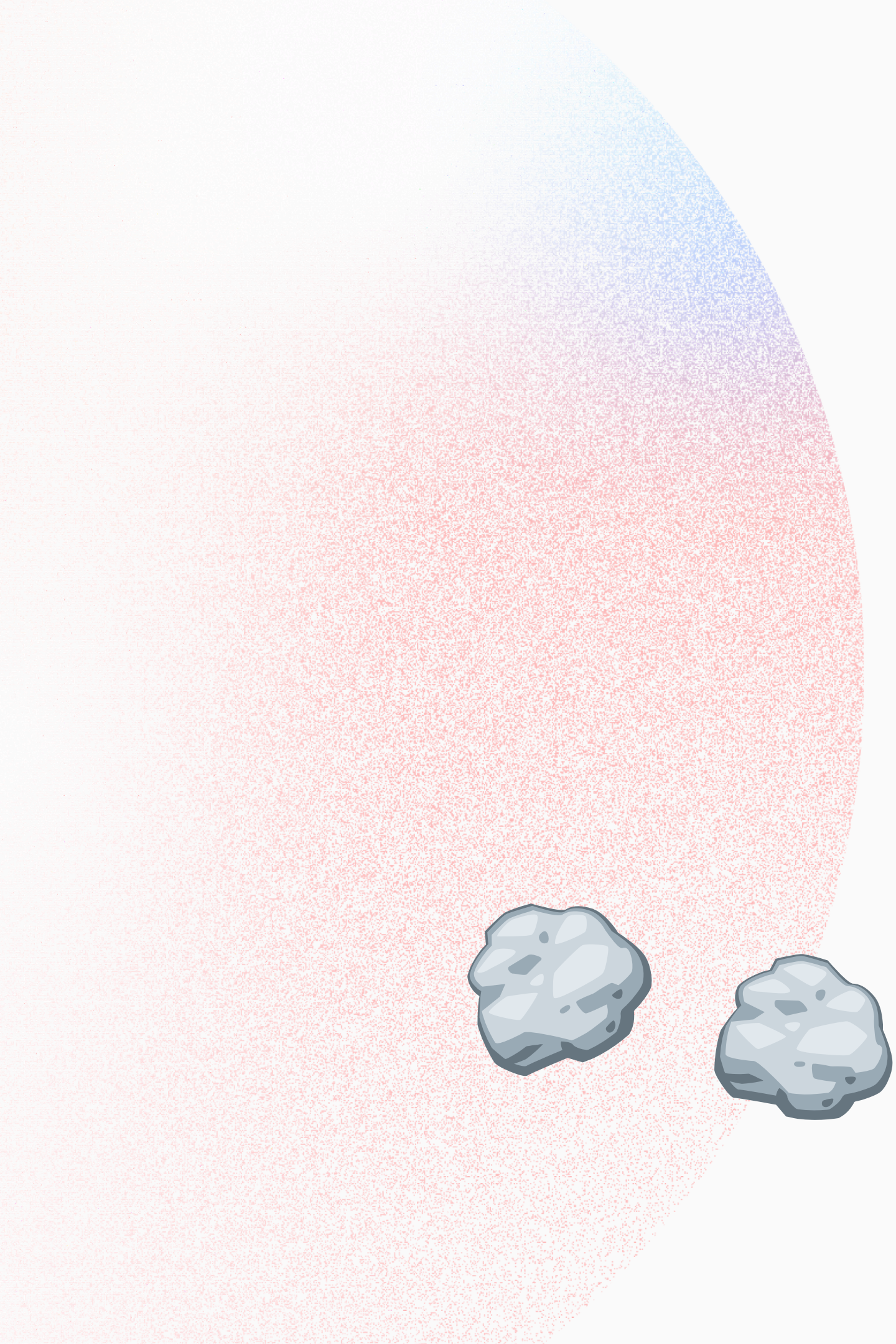
Achievements

1. Two-timescale design
2. Predictive, not reactive
3. Grouping rule for asymmetric topologies
4. Designed around real switch costs

Take away...

Take away...

Machine learning can move data center network control from reactive response to proactive prevention.



The End

TYSM!

資訊116：石浚邑、石維廉

謝謝 張燕光 教授

也特別感謝碩二 李昆翰 學長兩學期來的幫助！