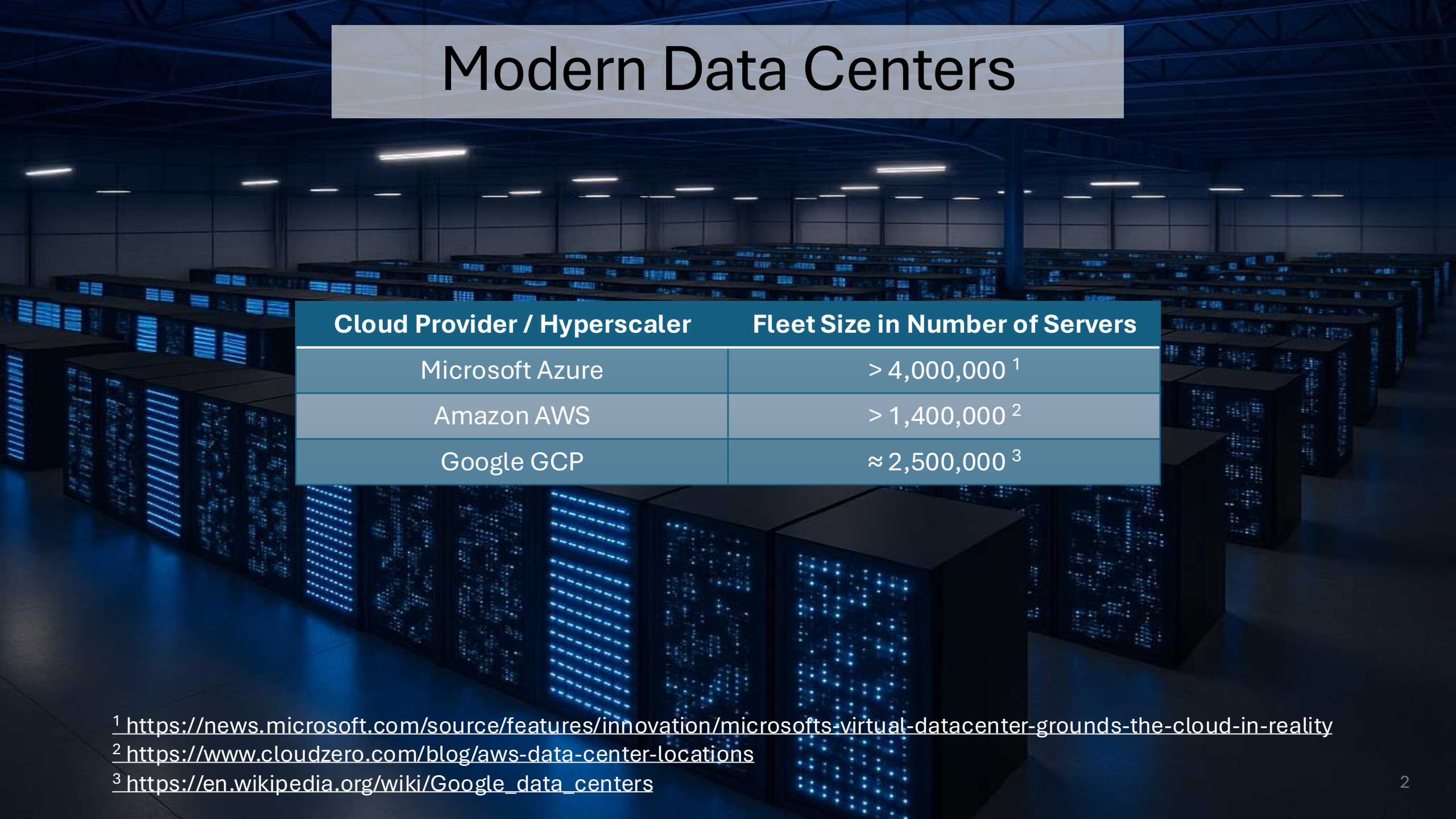
Rearchitecting the End Host Network for the Terabit Per Second Era

Annus Zulfiqar

Ph.D. Dissertation Defense

University of Michigan 2026

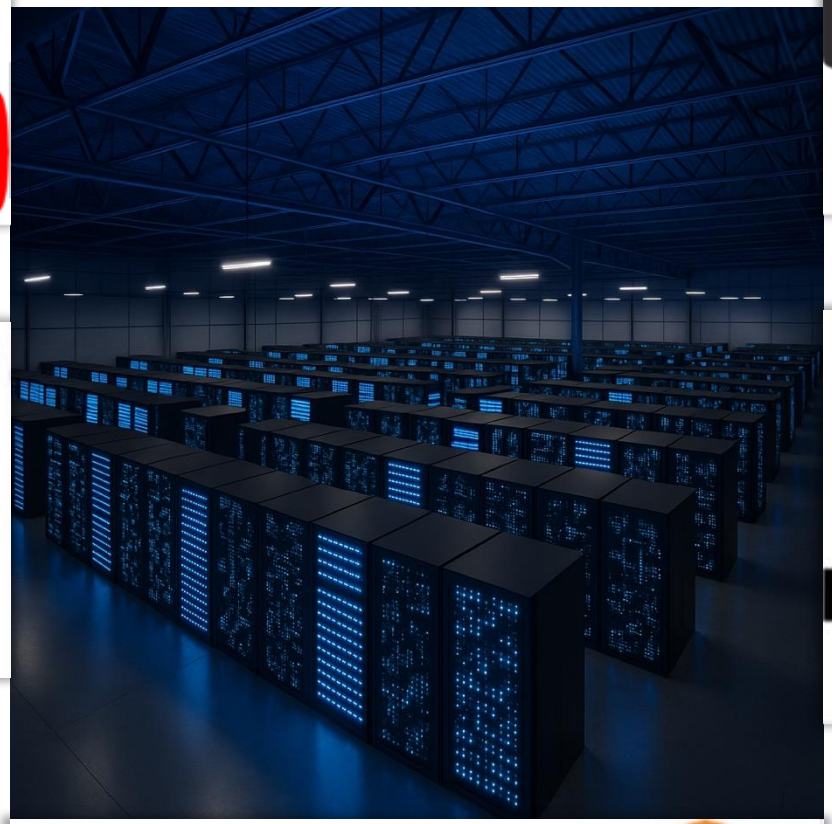
Modern Data Centers



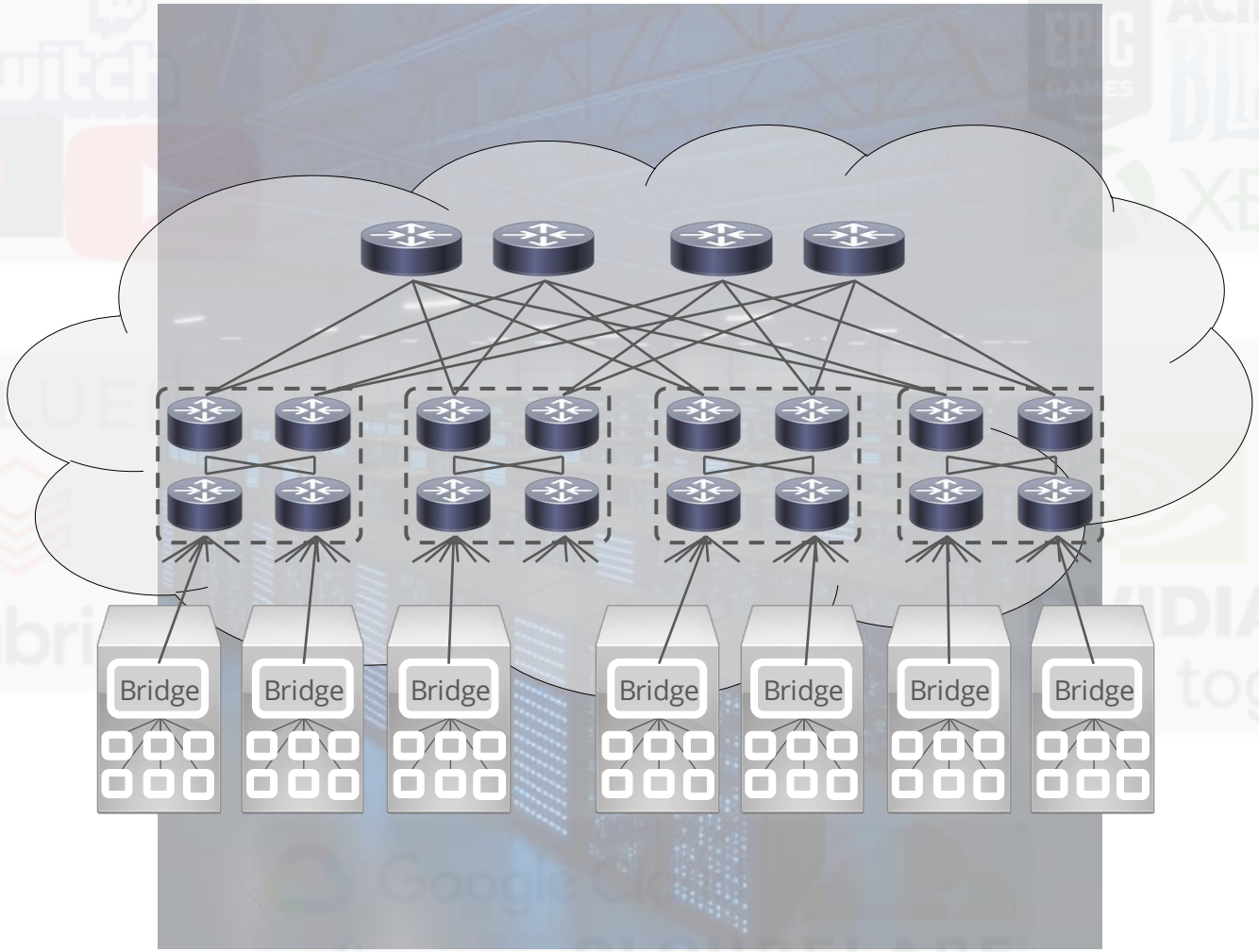
Cloud Provider / Hyperscaler	Fleet Size in Number of Servers
Microsoft Azure	> 4,000,000 [1]
Amazon AWS	> 1,400,000 [2]
Google GCP	≈ 2,500,000 [3]

[1] https://news.microsoft.com/source/features/innovation/microsofts-virtual-datacenter-grounds-the-cloud-in-reality
[2] https://www.cloudzero.com/blog/aws-data-center-locations
[3] https://en.wikipedia.org/wiki/Google_data_centers

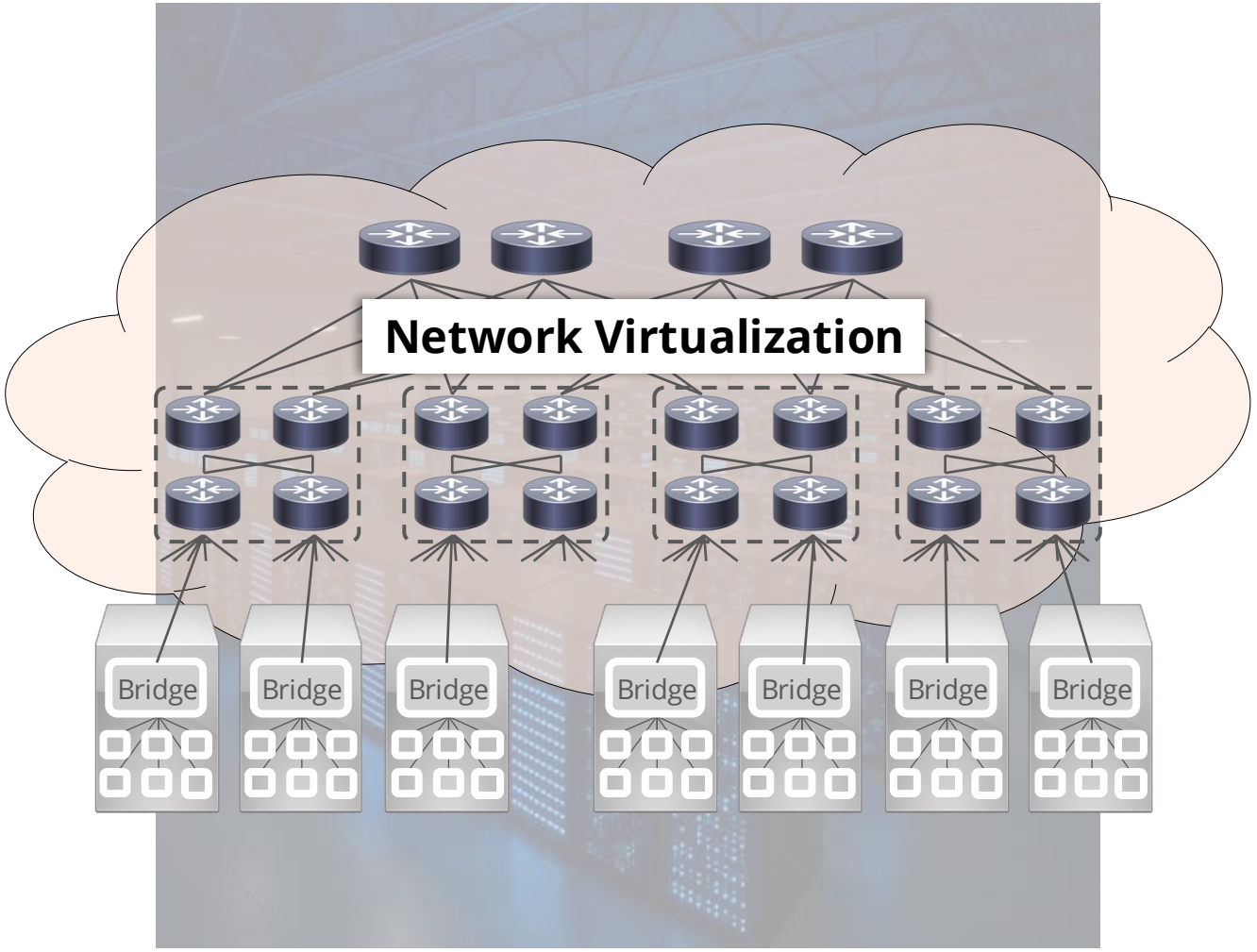
Modern Data Centers



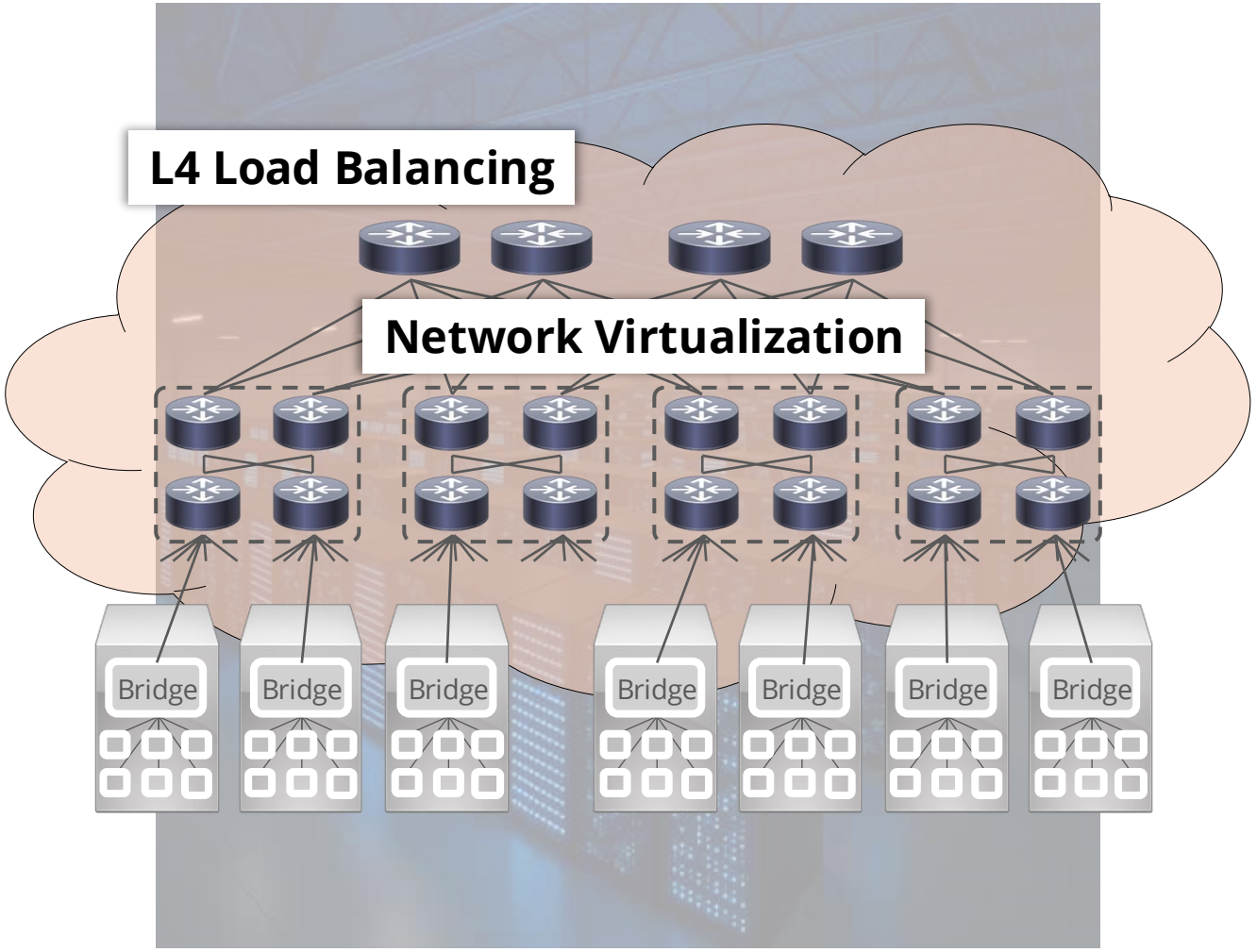
Modern Data Centers



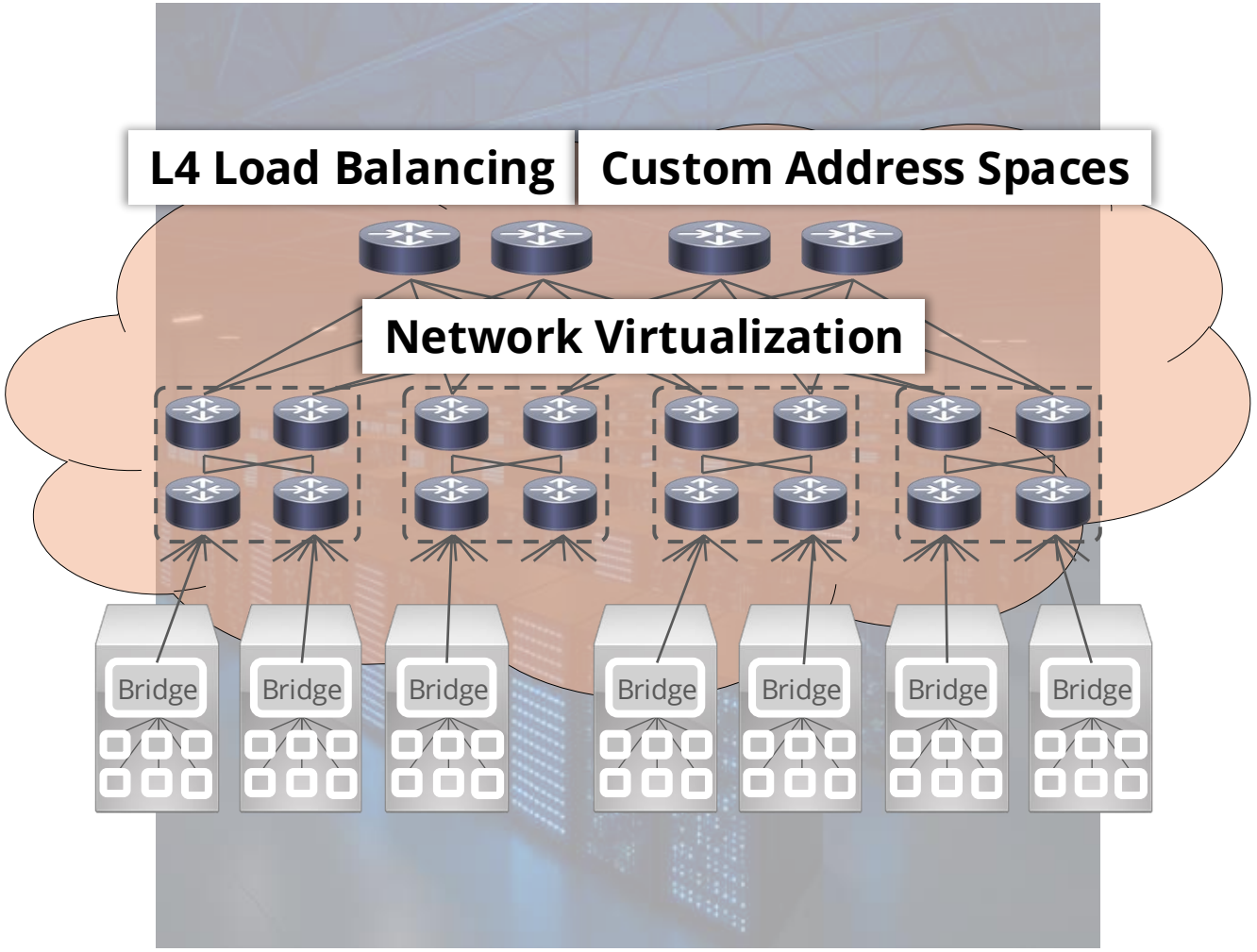
Modern Data Centers



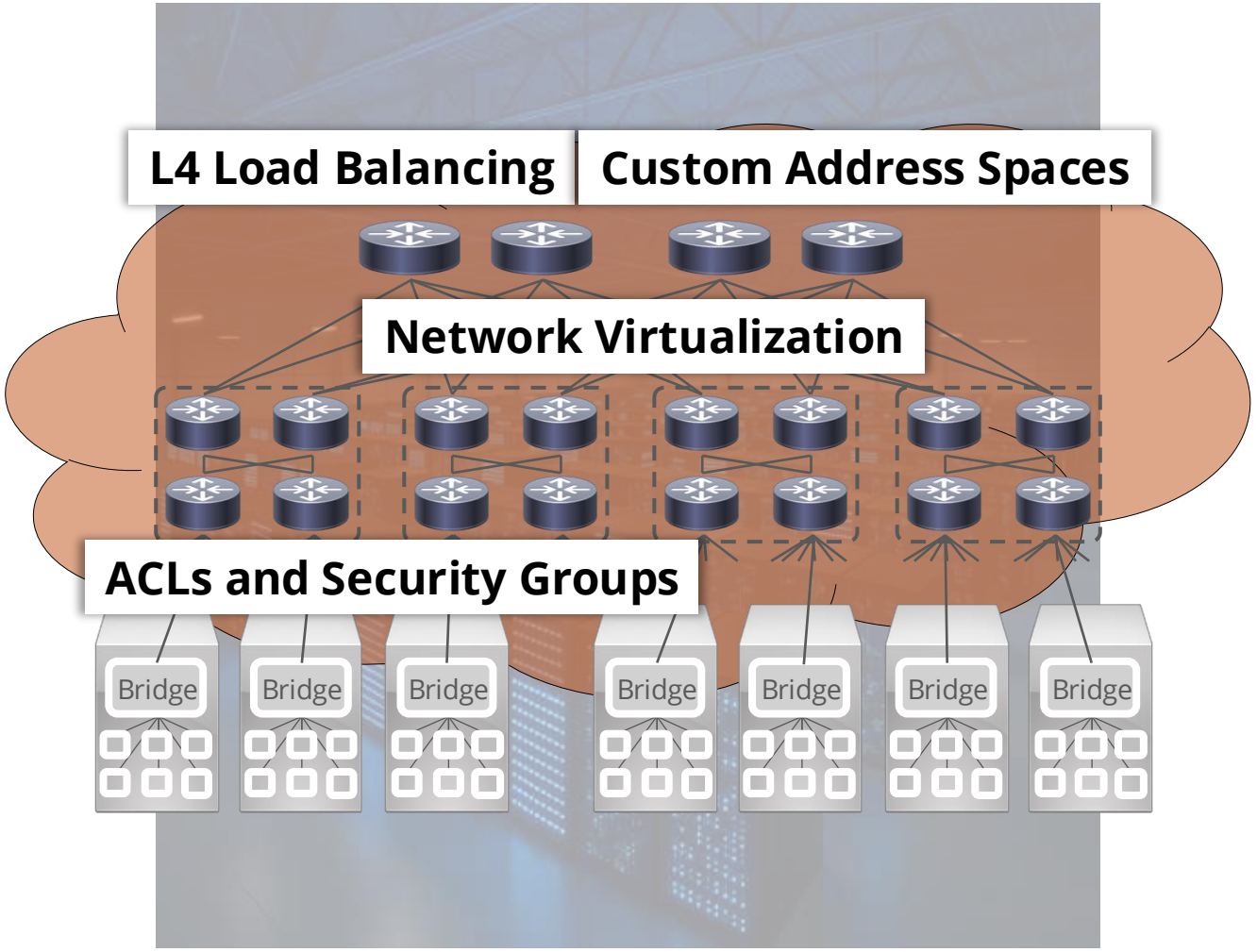
Modern Data Centers



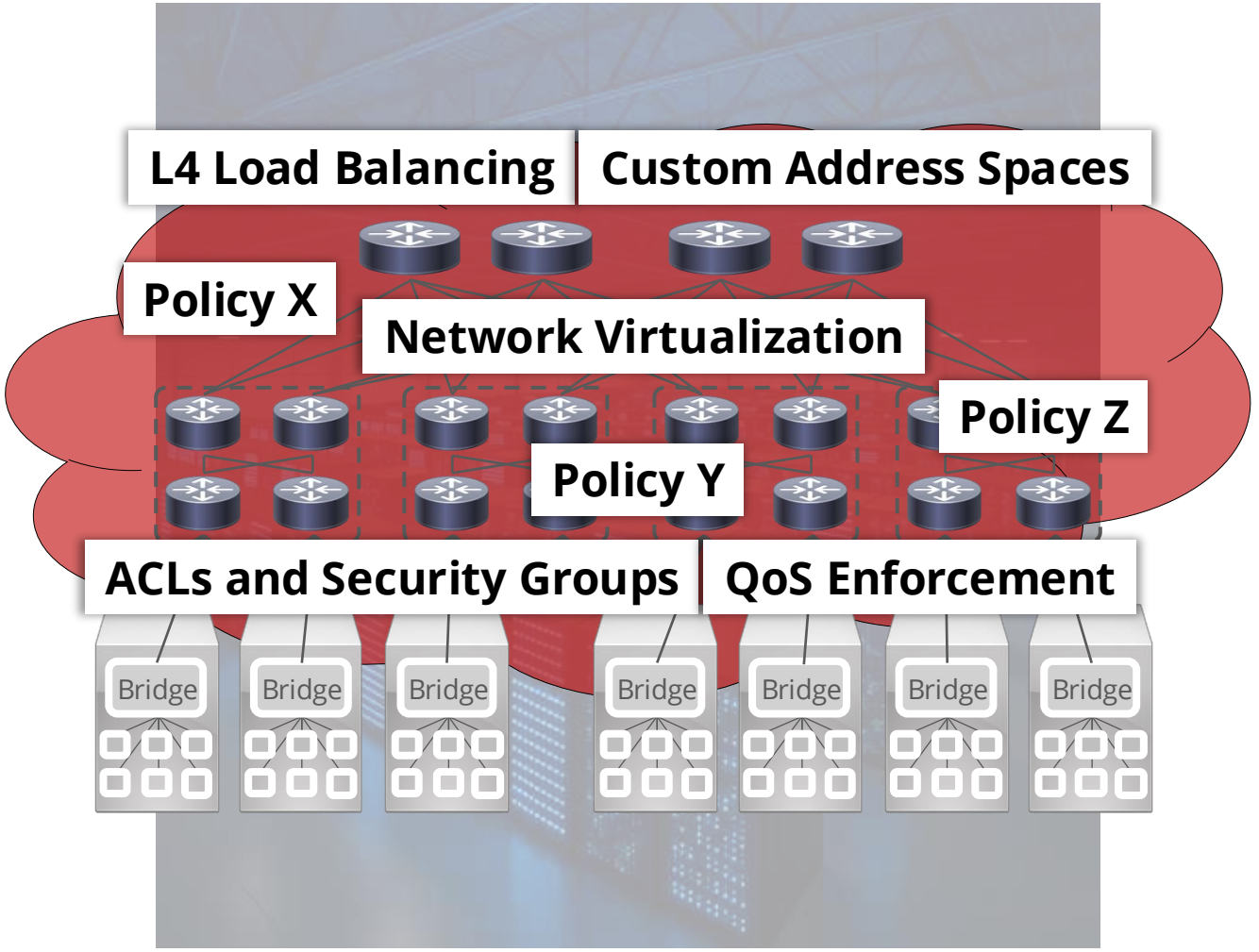
Modern Data Centers



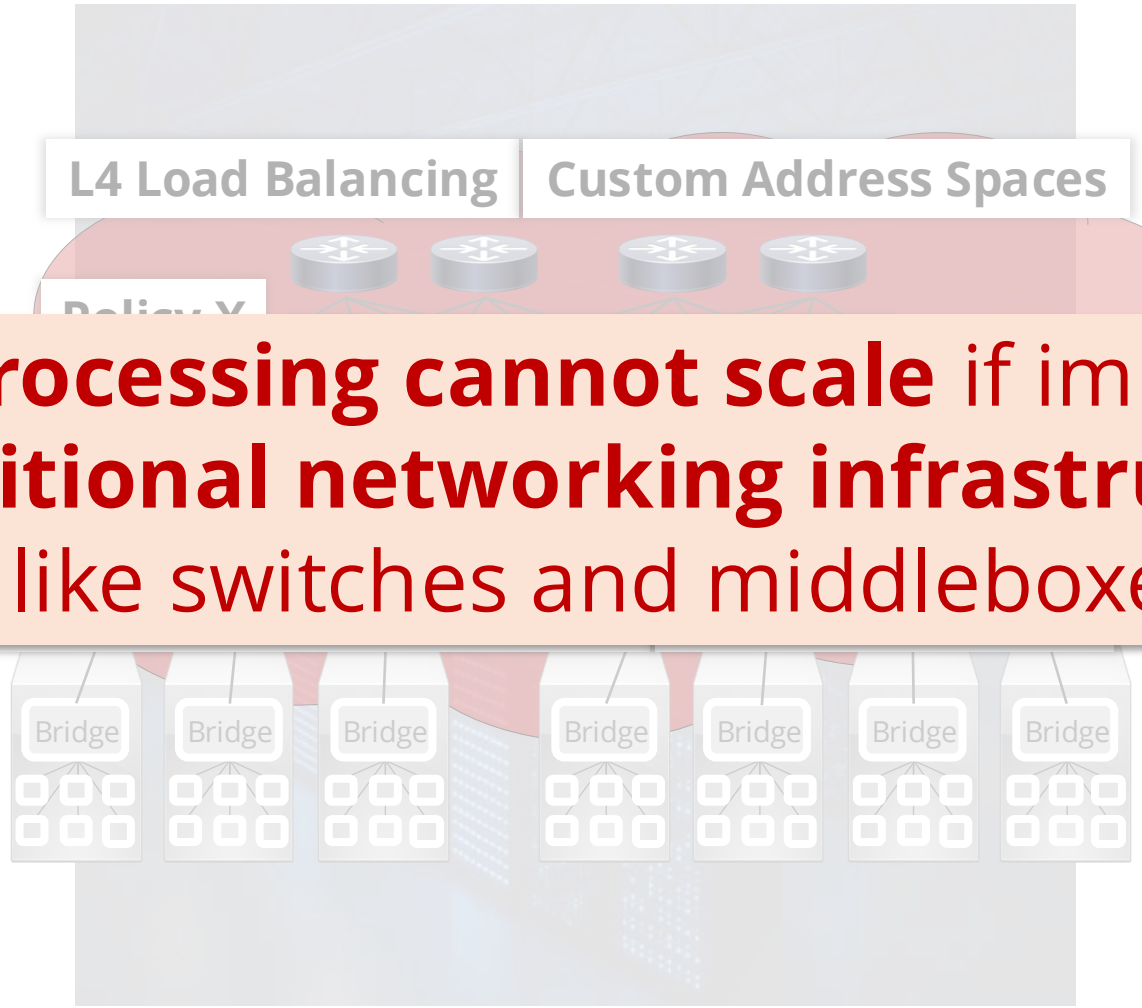
Modern Data Centers



Modern Data Centers

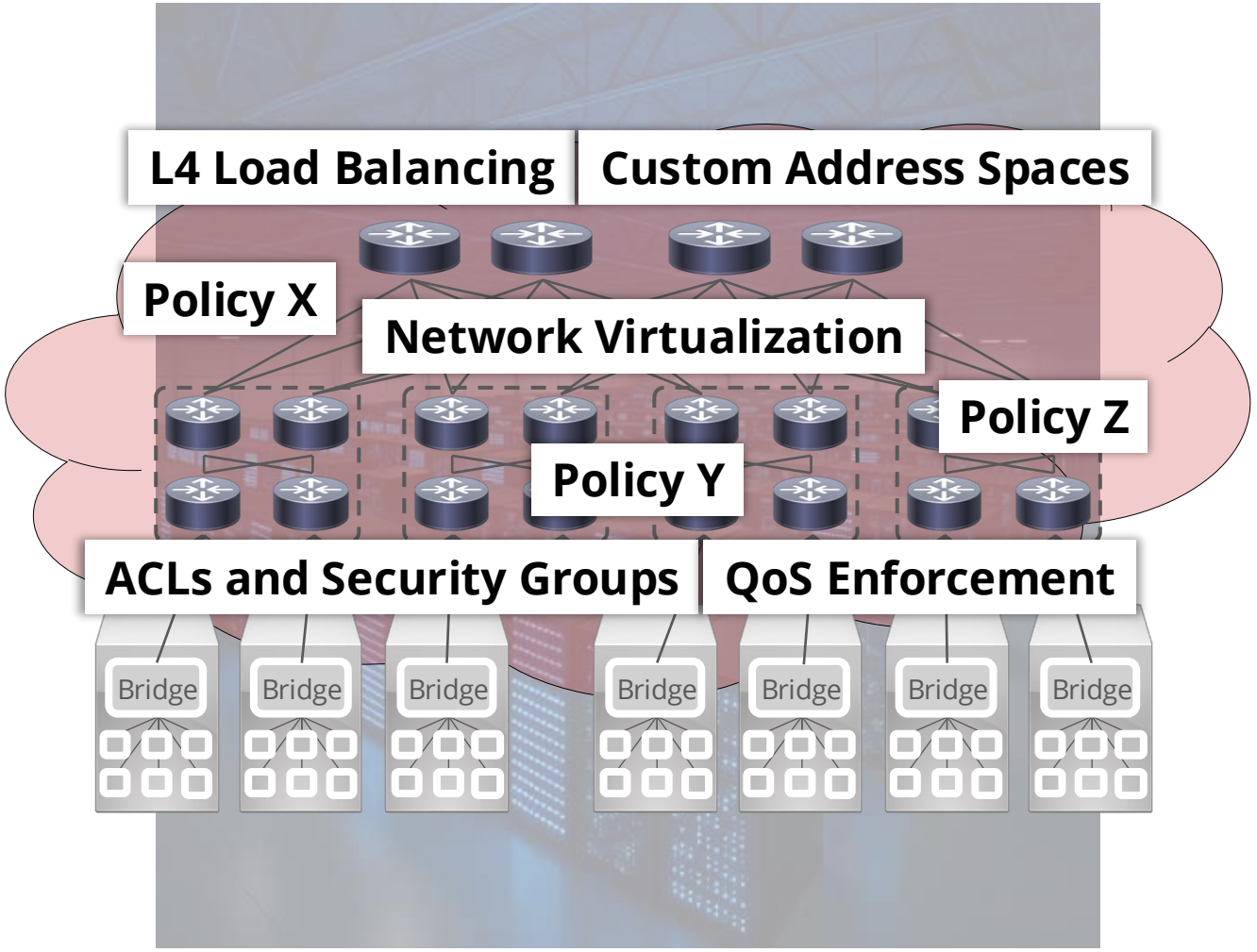


Modern Data Centers

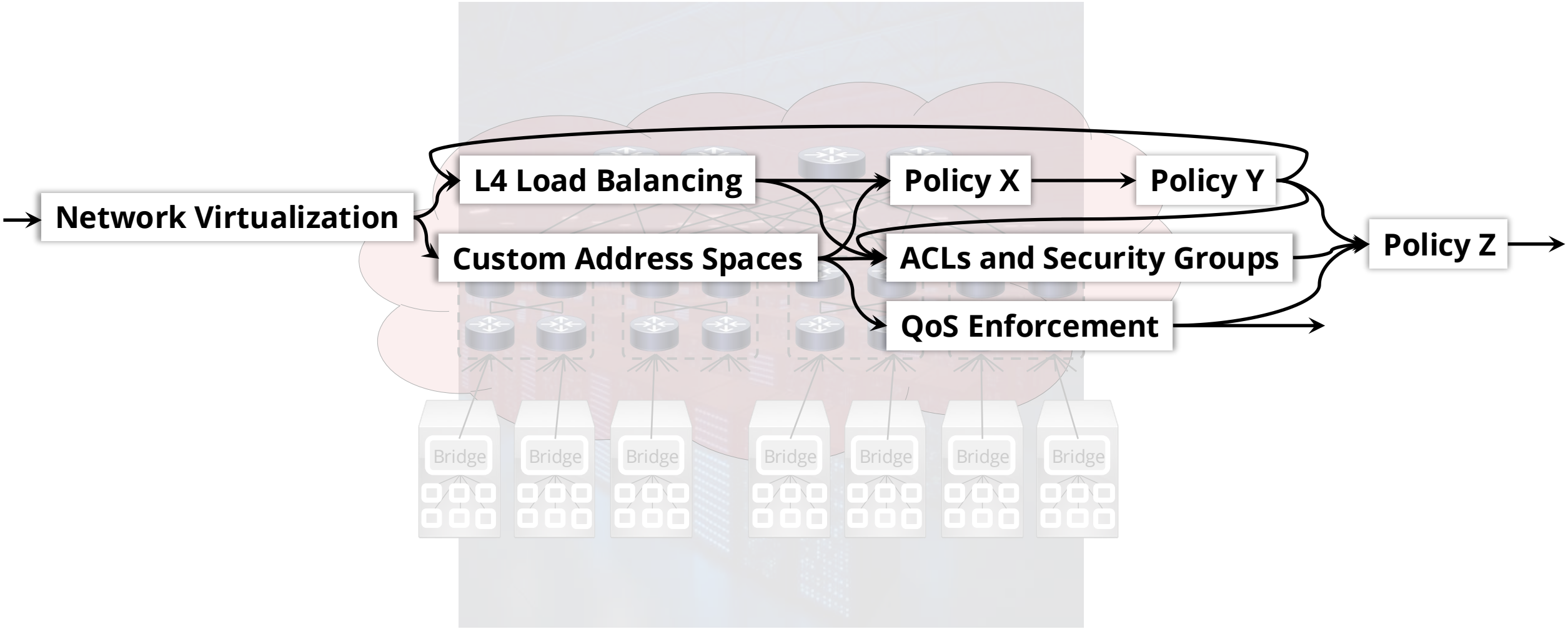


Network processing cannot scale if implemented on **traditional networking infrastructure**
like switches and middleboxes

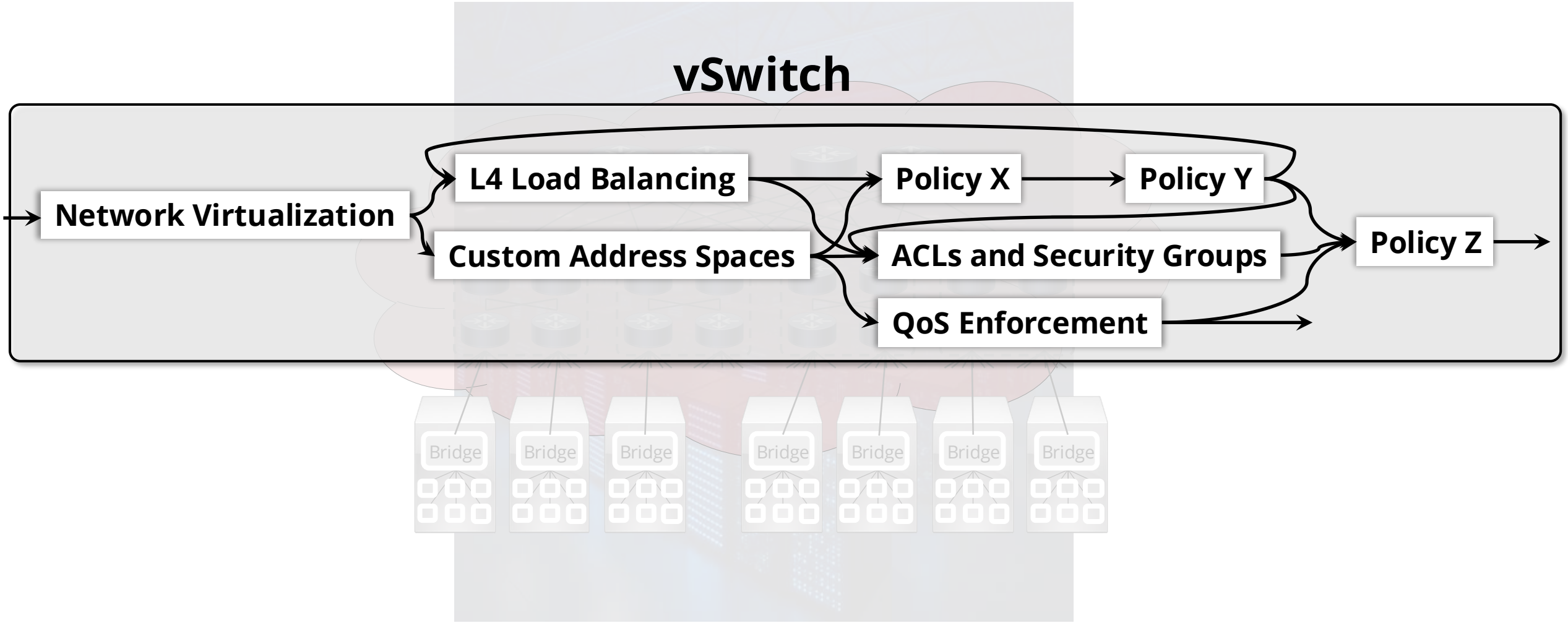
Modern Data Centers



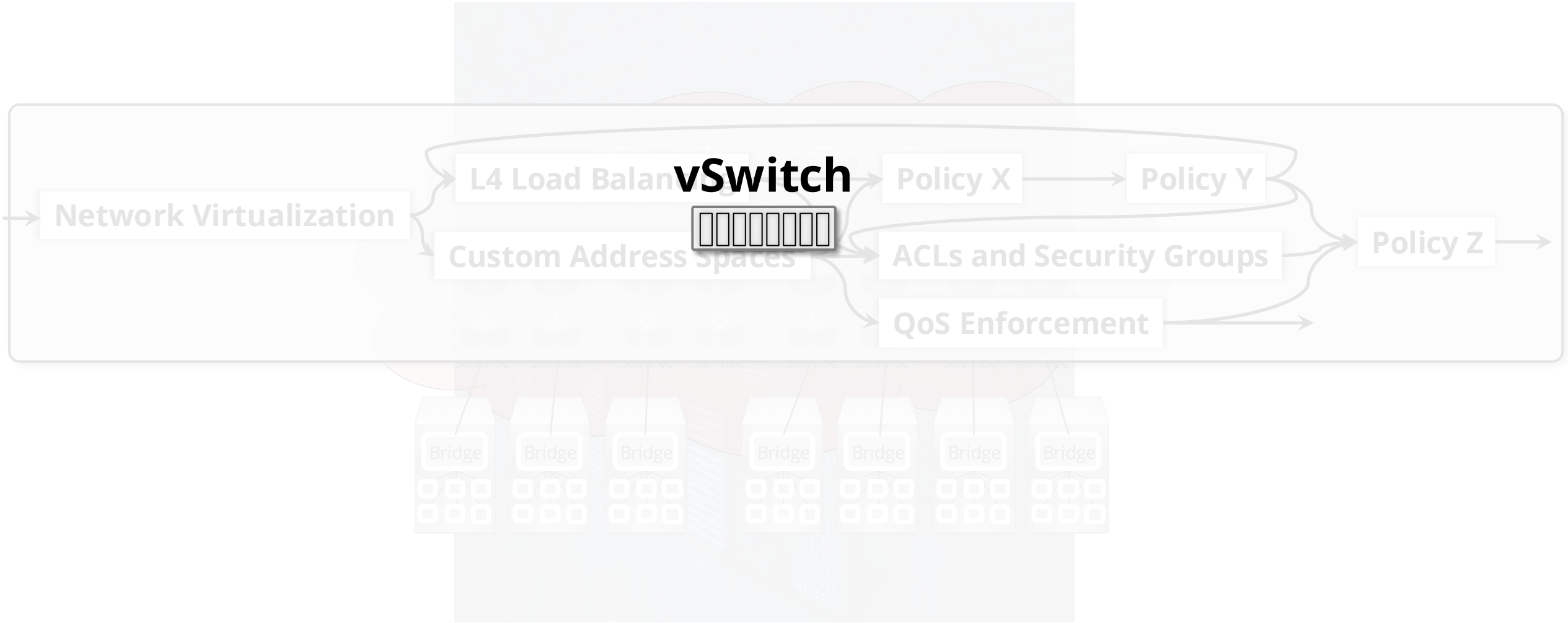
Virtualize the Switch



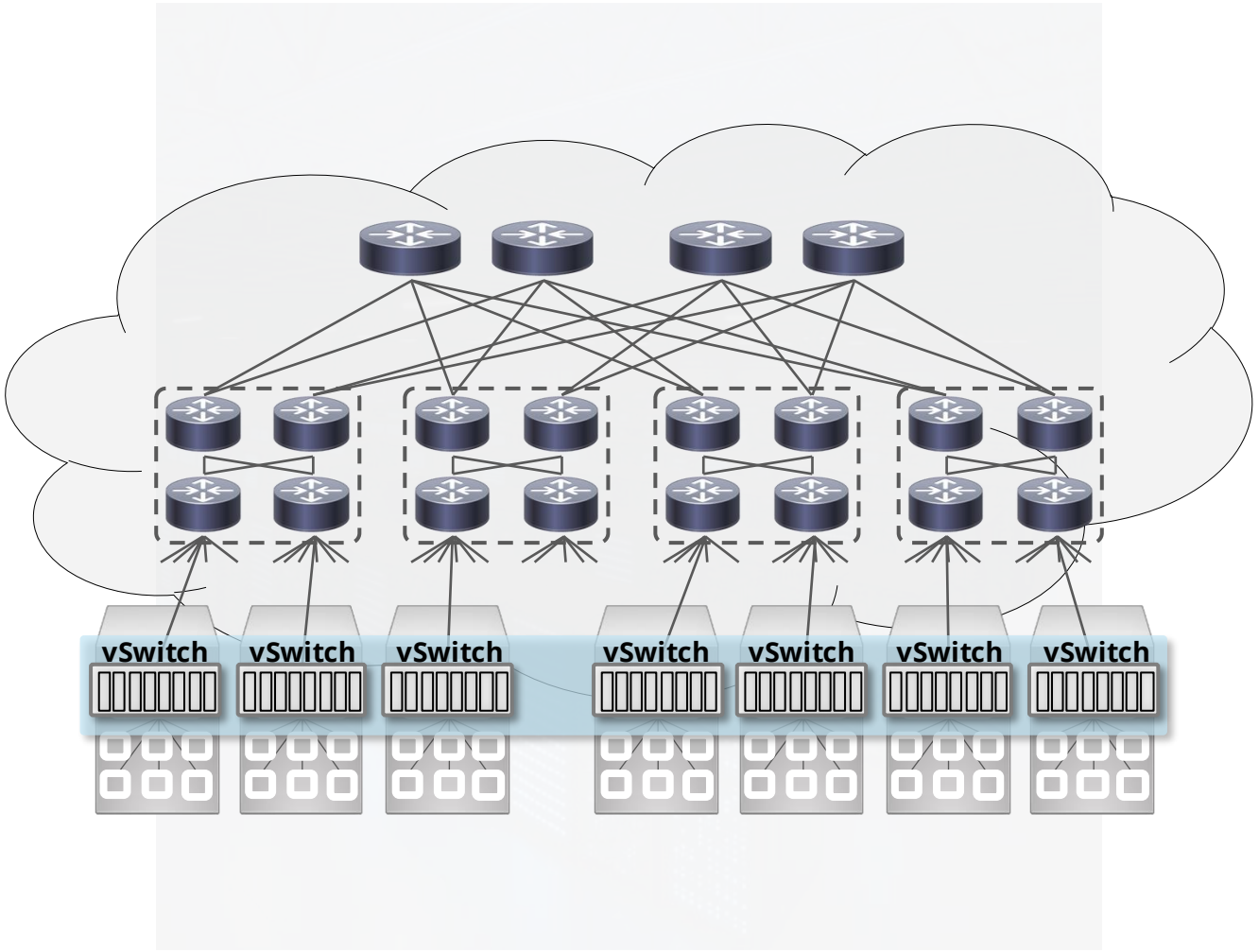
Virtualize the Switch



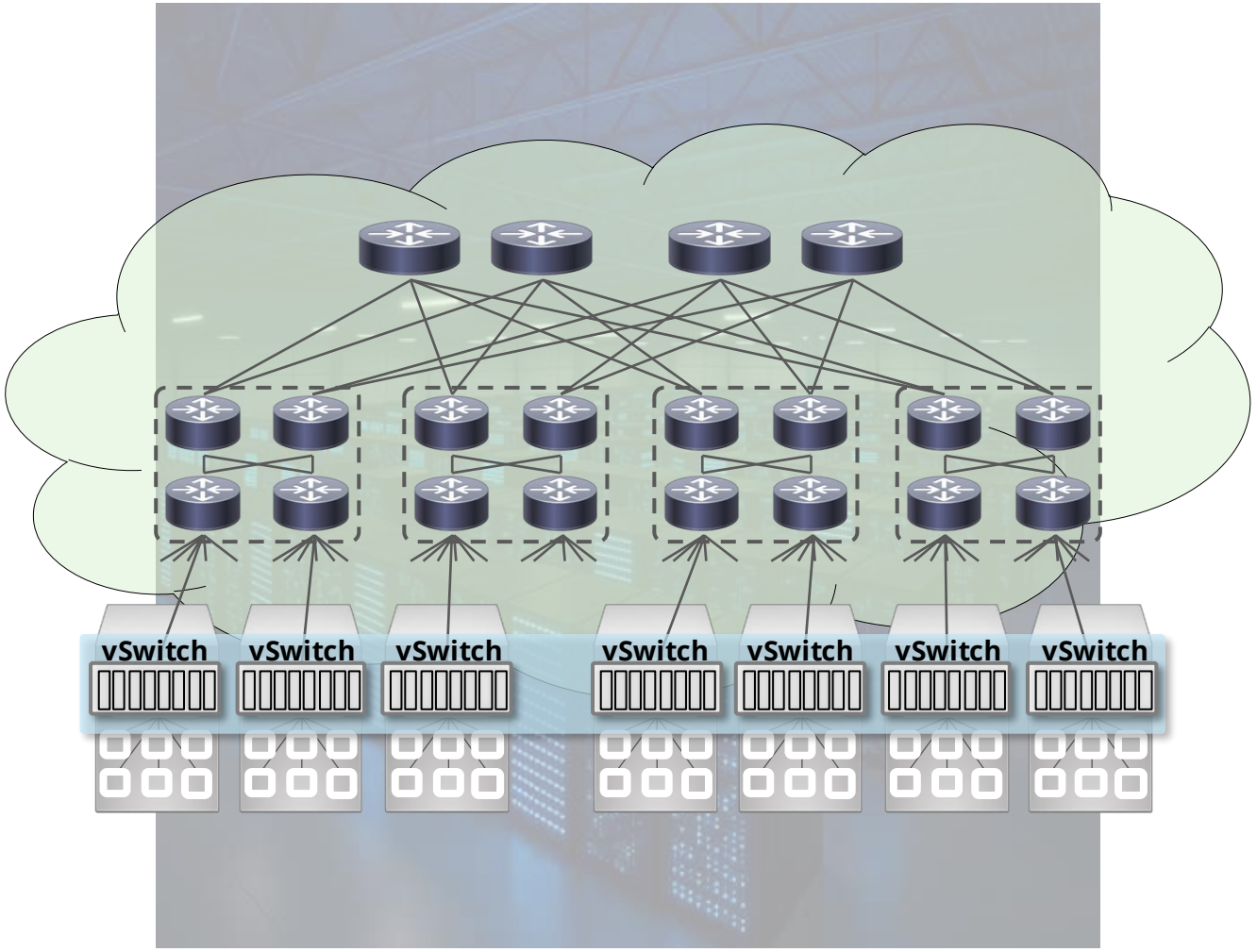
Virtualize the Switch



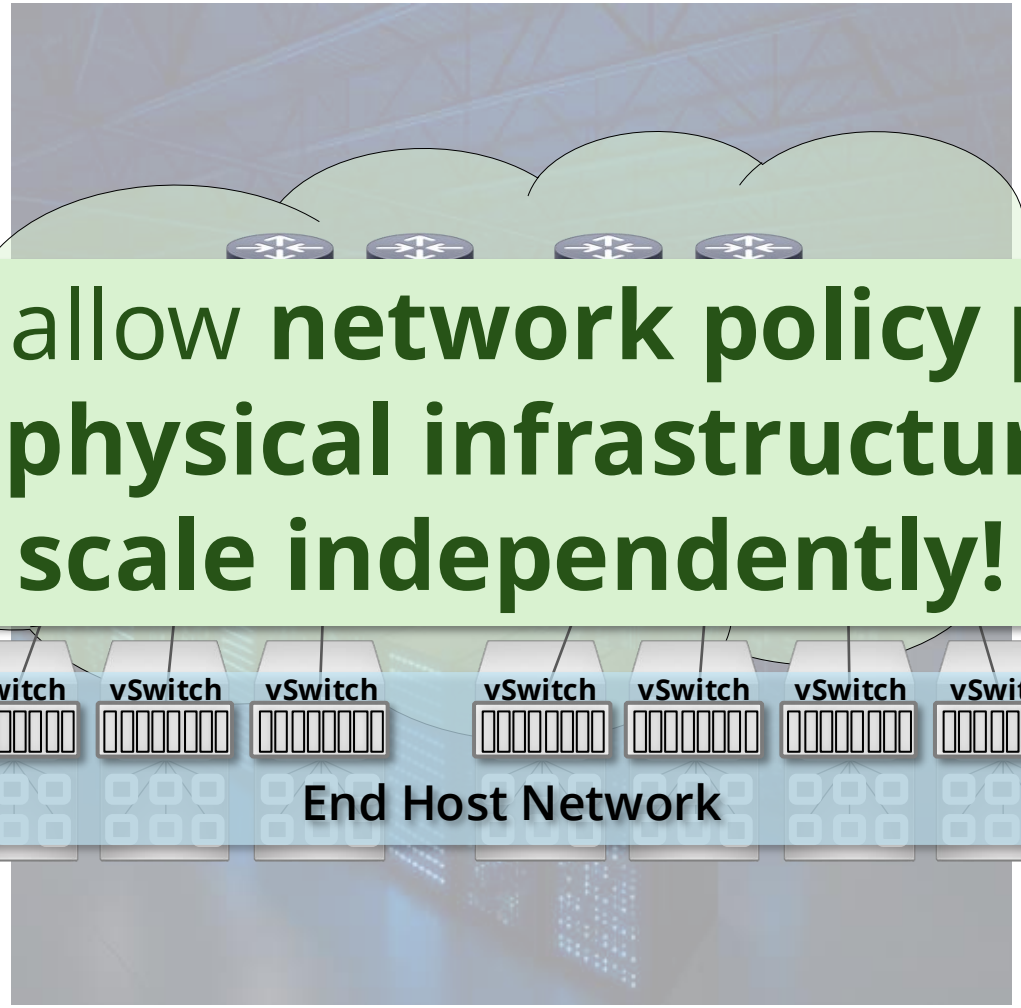
End Host Networks



End Host Networks

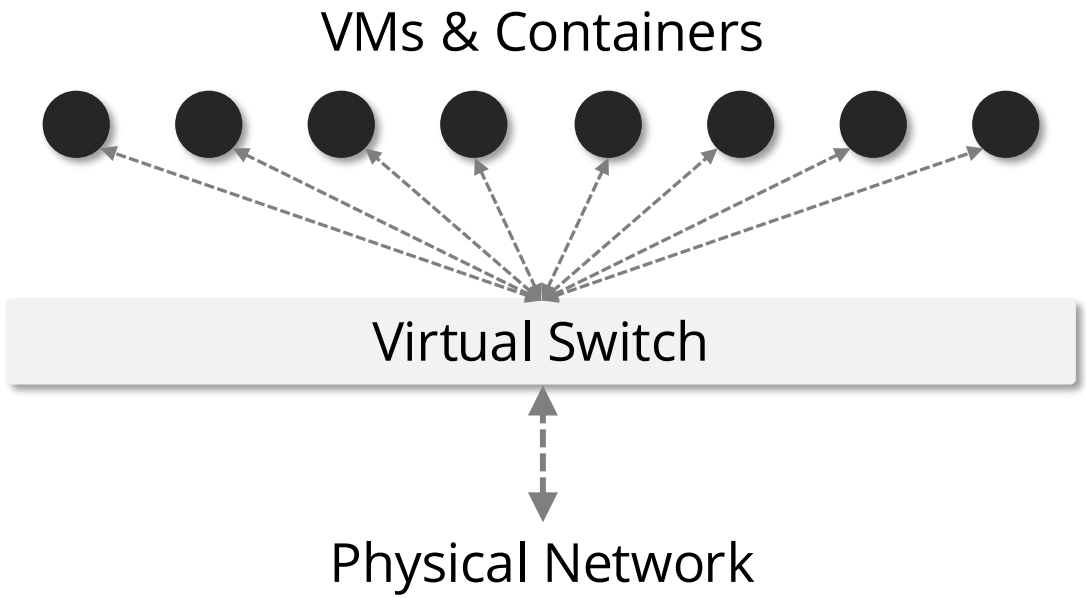


End Host Networks

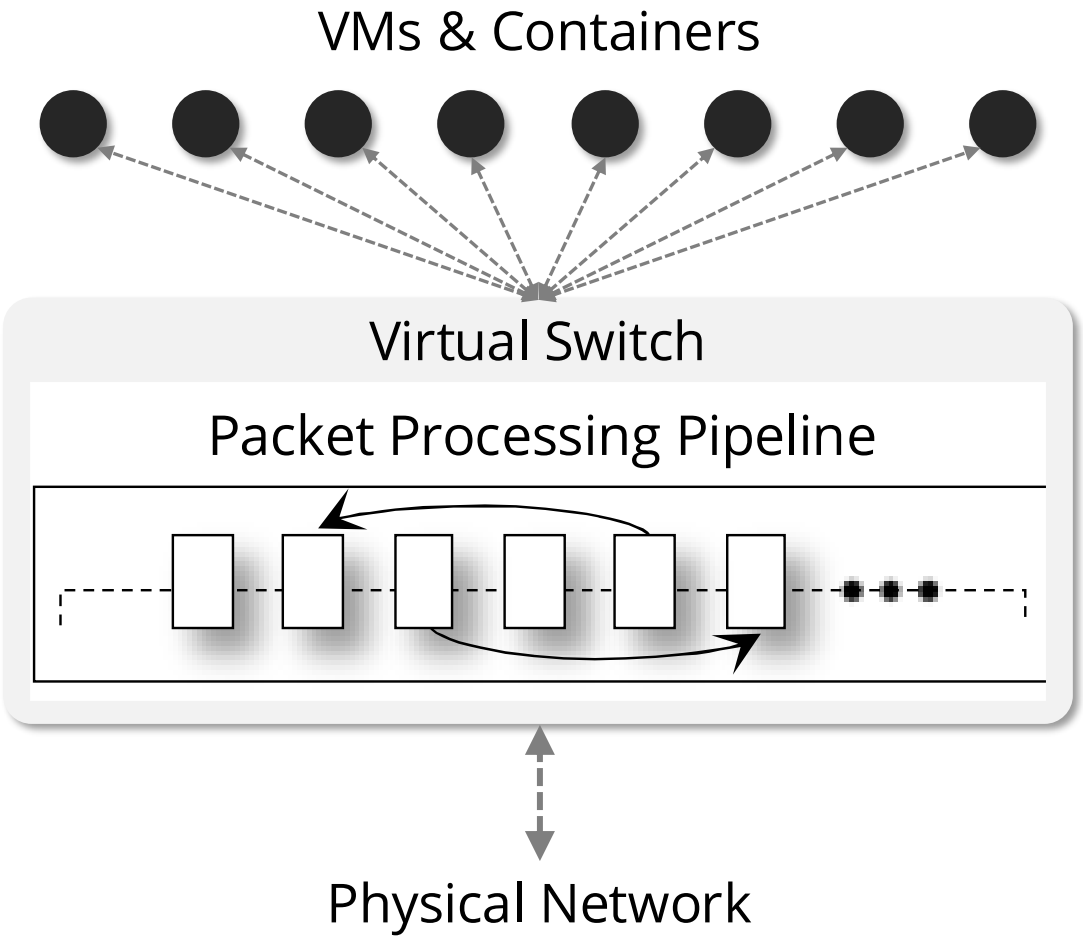


vSwitches allow **network policy processing** and **physical infrastructure** to **scale independently!**

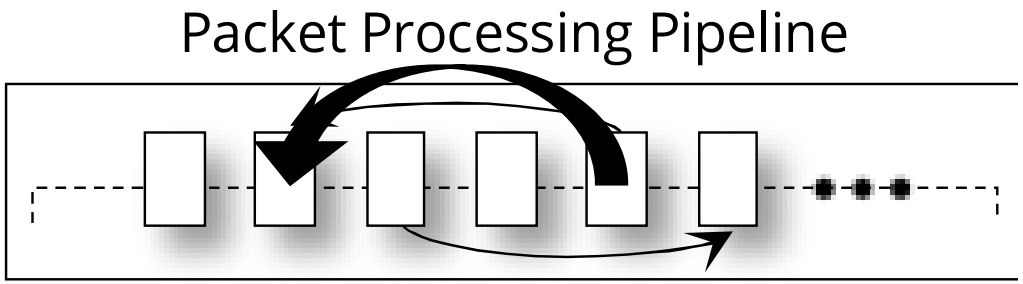
Virtual Switches in the Data Center



Virtual Switches in the Data Center

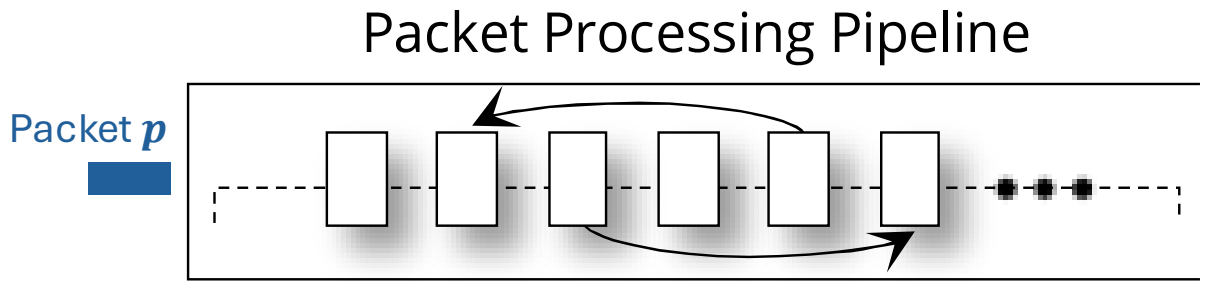


Virtual Switches in the Data Center

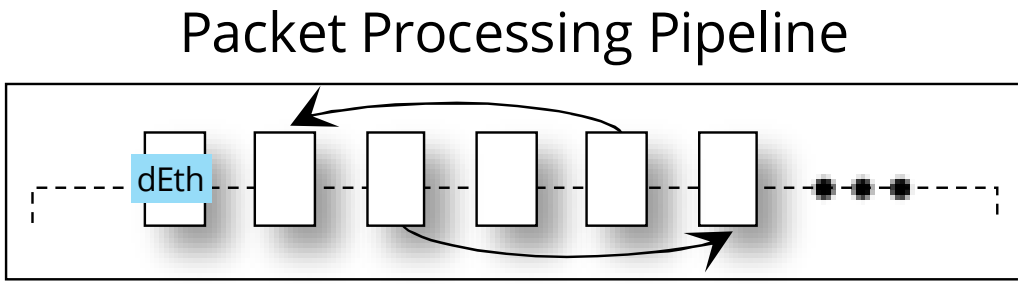


Some networking policies require **recursive application of rules**, e.g., encapsulation and decapsulation actions

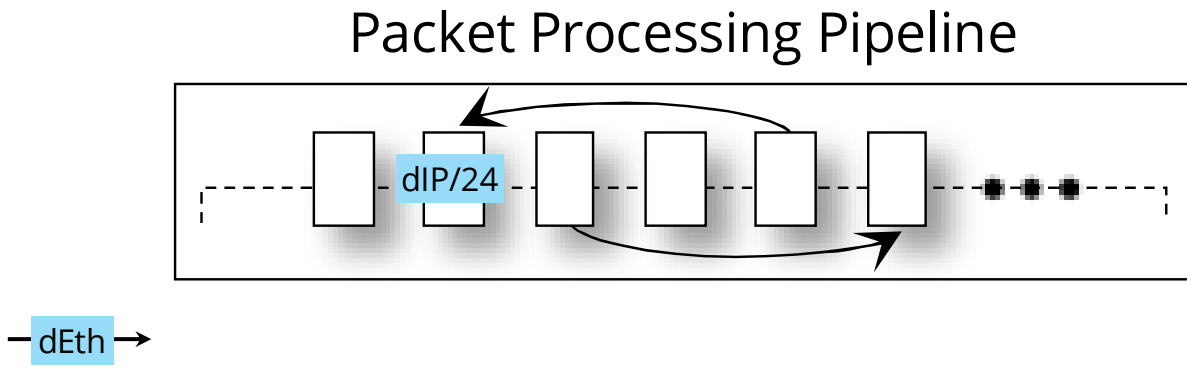
Virtual Switches in the Data Center



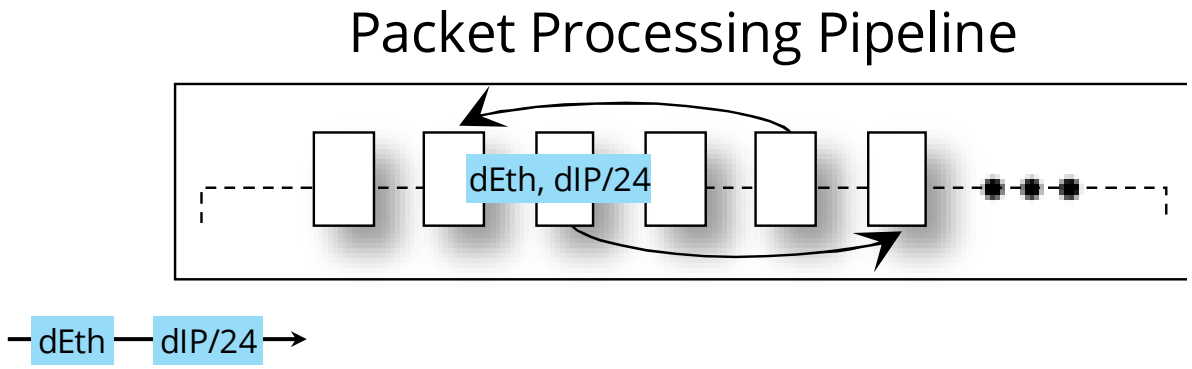
Virtual Switches in the Data Center



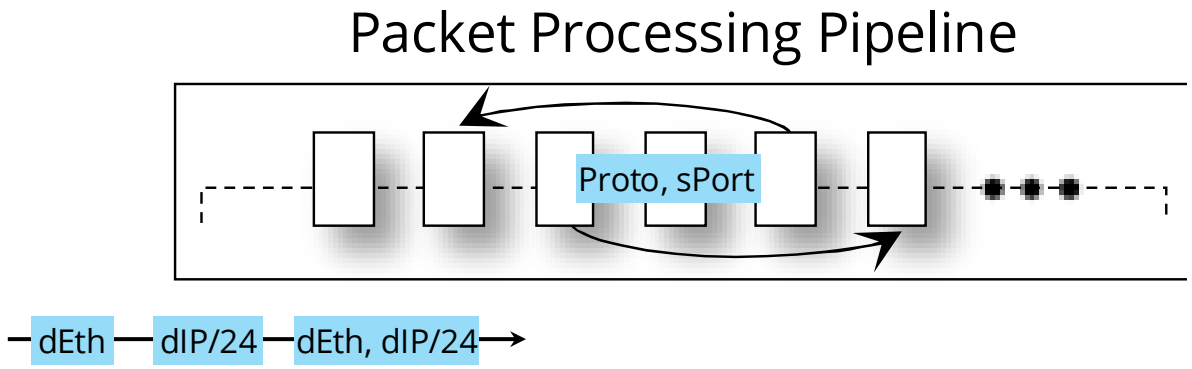
Virtual Switches in the Data Center



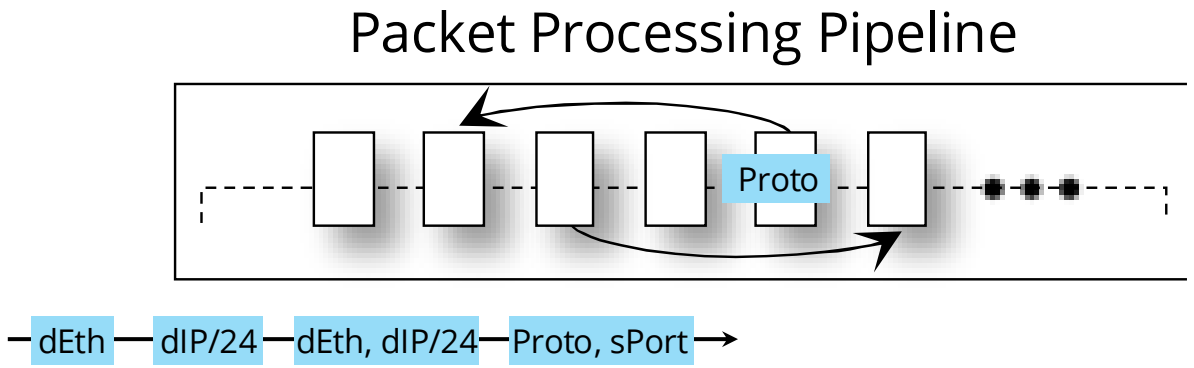
Virtual Switches in the Data Center



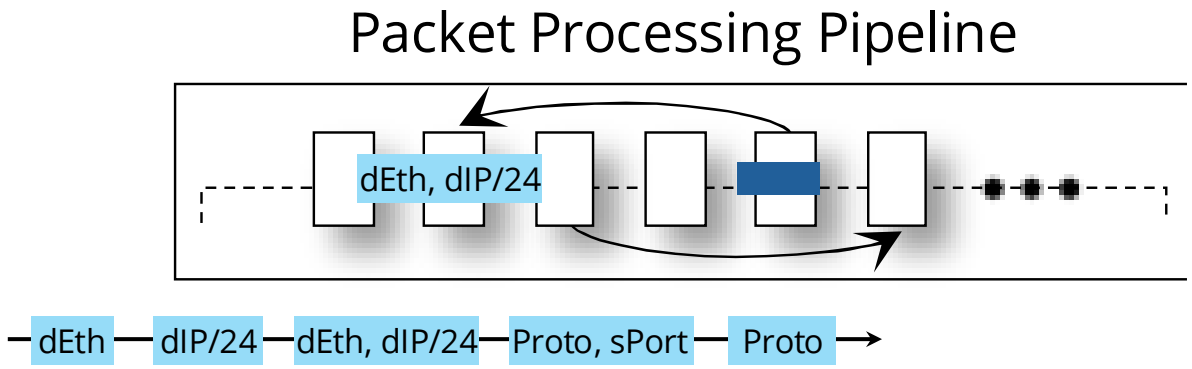
Virtual Switches in the Data Center



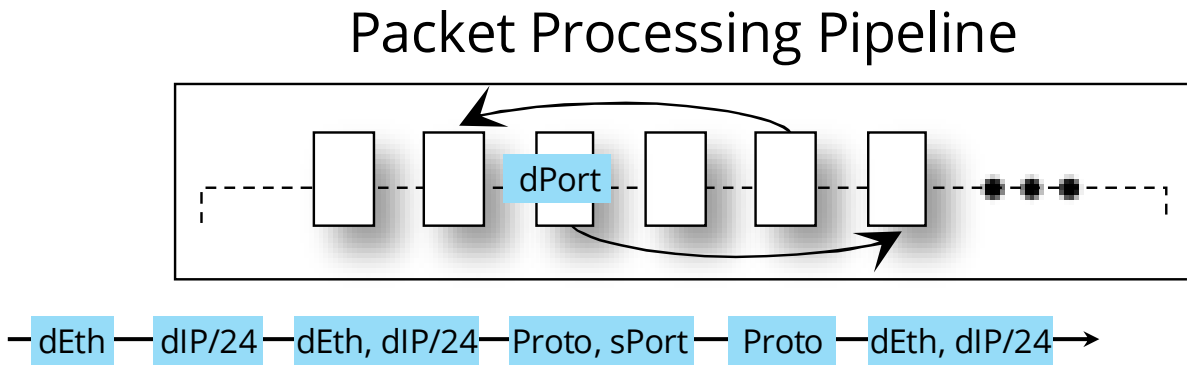
Virtual Switches in the Data Center



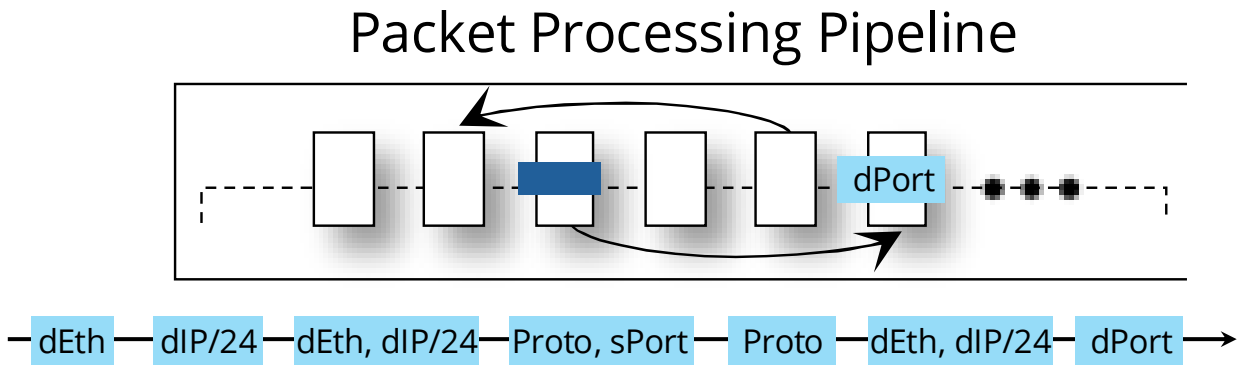
Virtual Switches in the Data Center



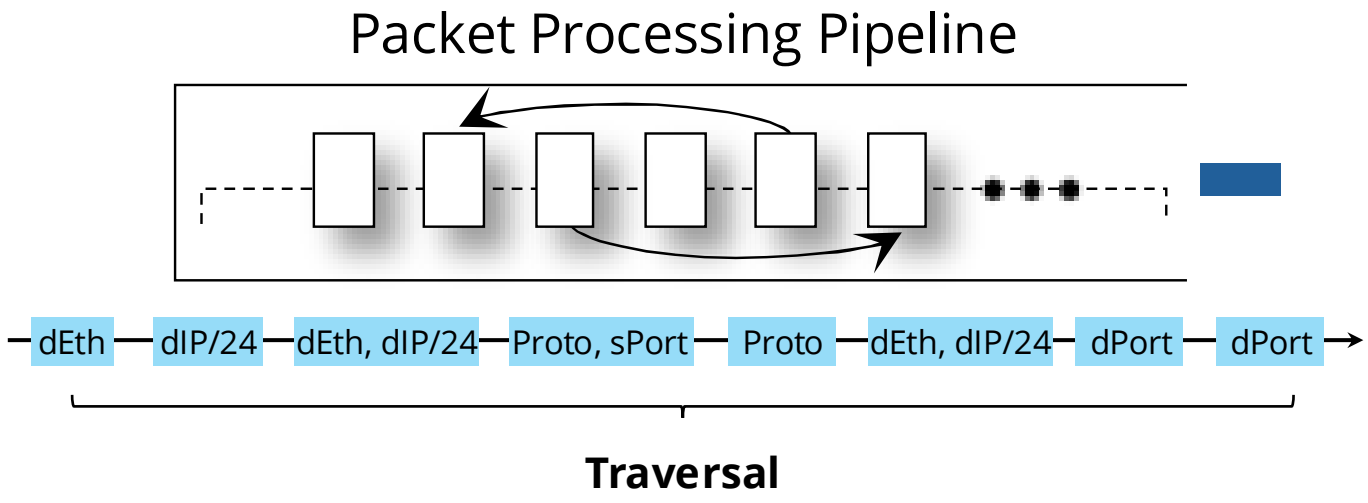
Virtual Switches in the Data Center



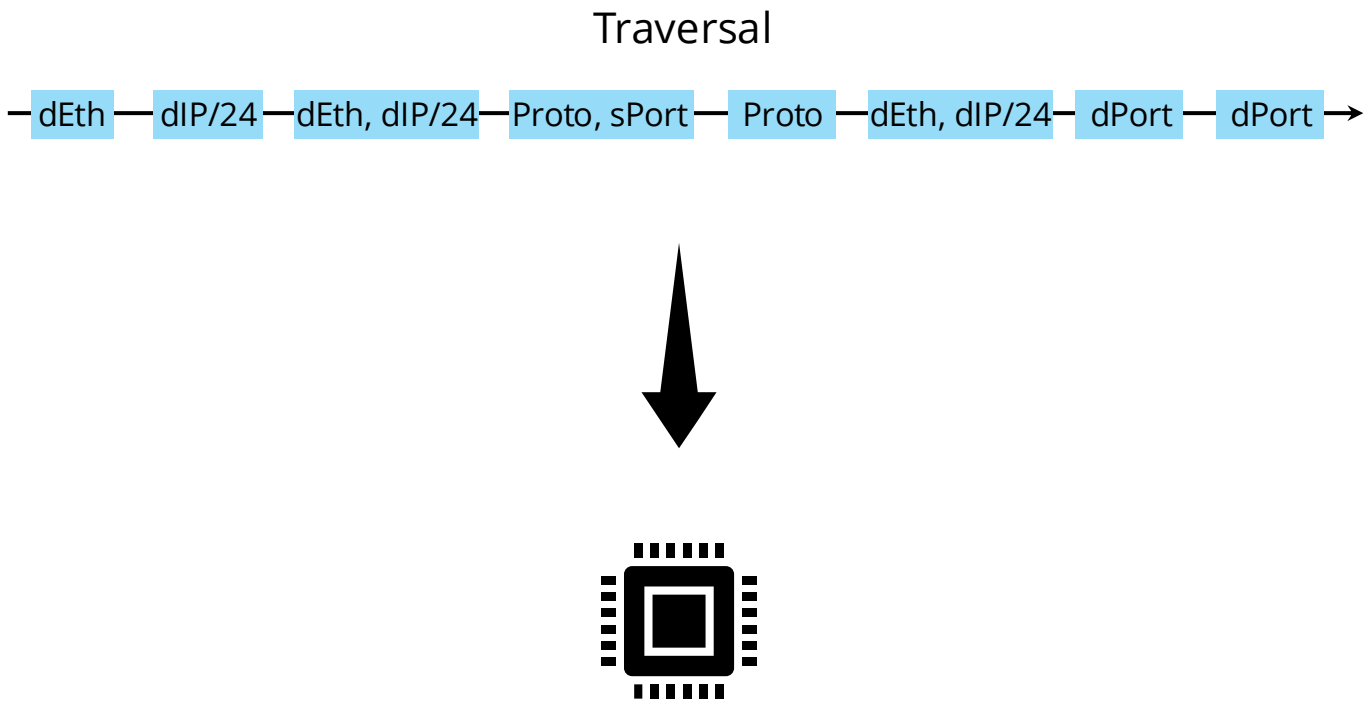
Virtual Switches in the Data Center



Virtual Switches in the Data Center

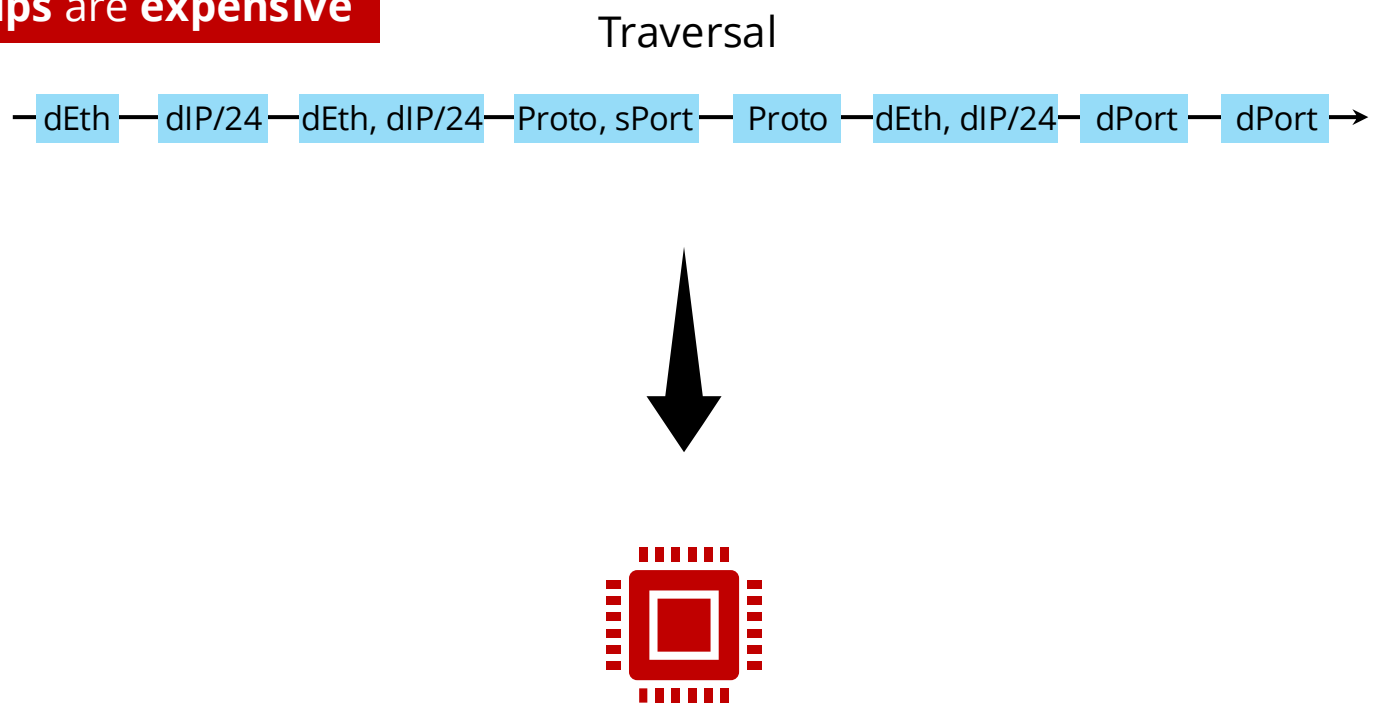


Virtual Switches in the Data Center



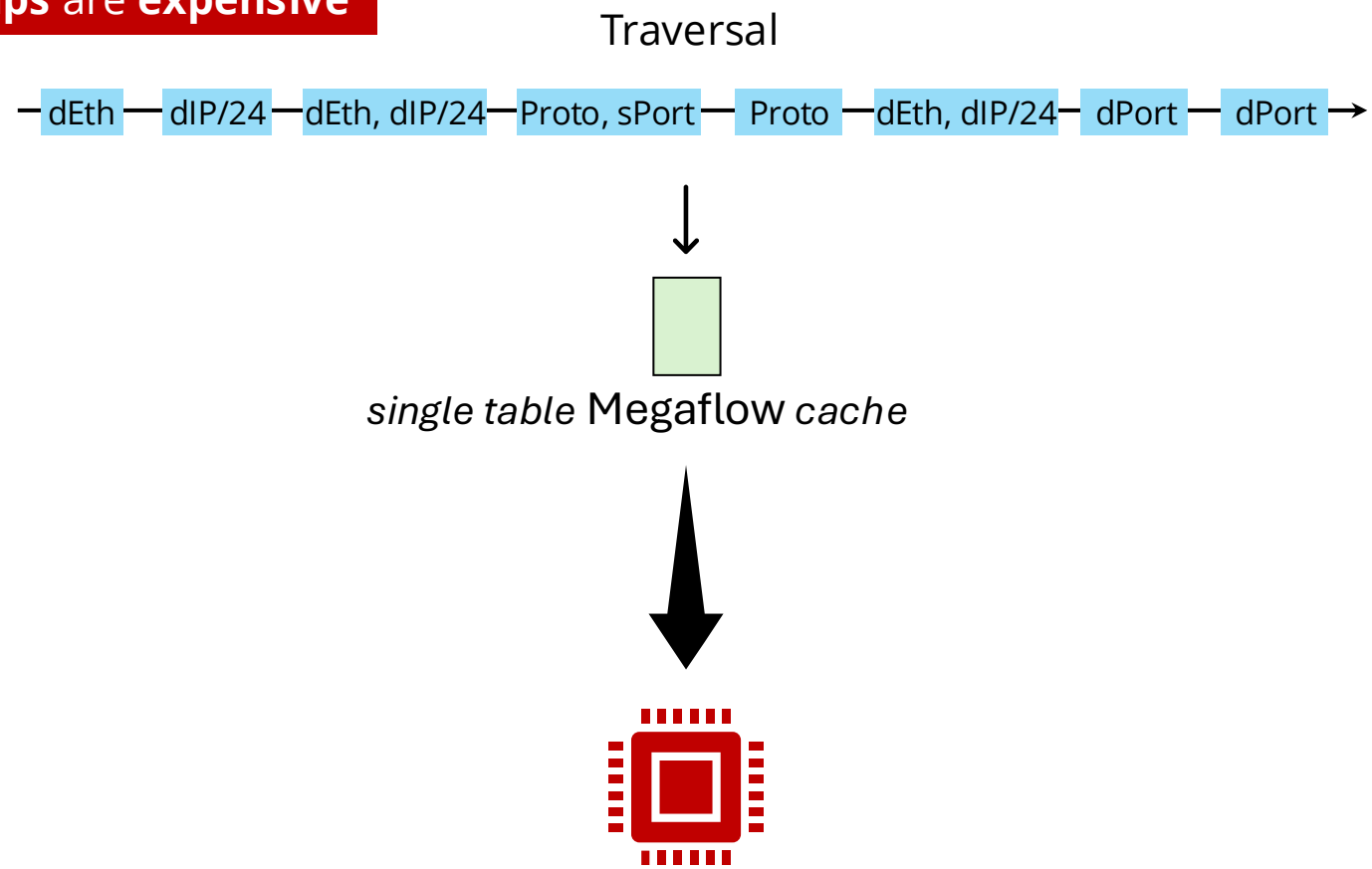
Virtual Switches in the Data Center

Multi-table lookups are **expensive**



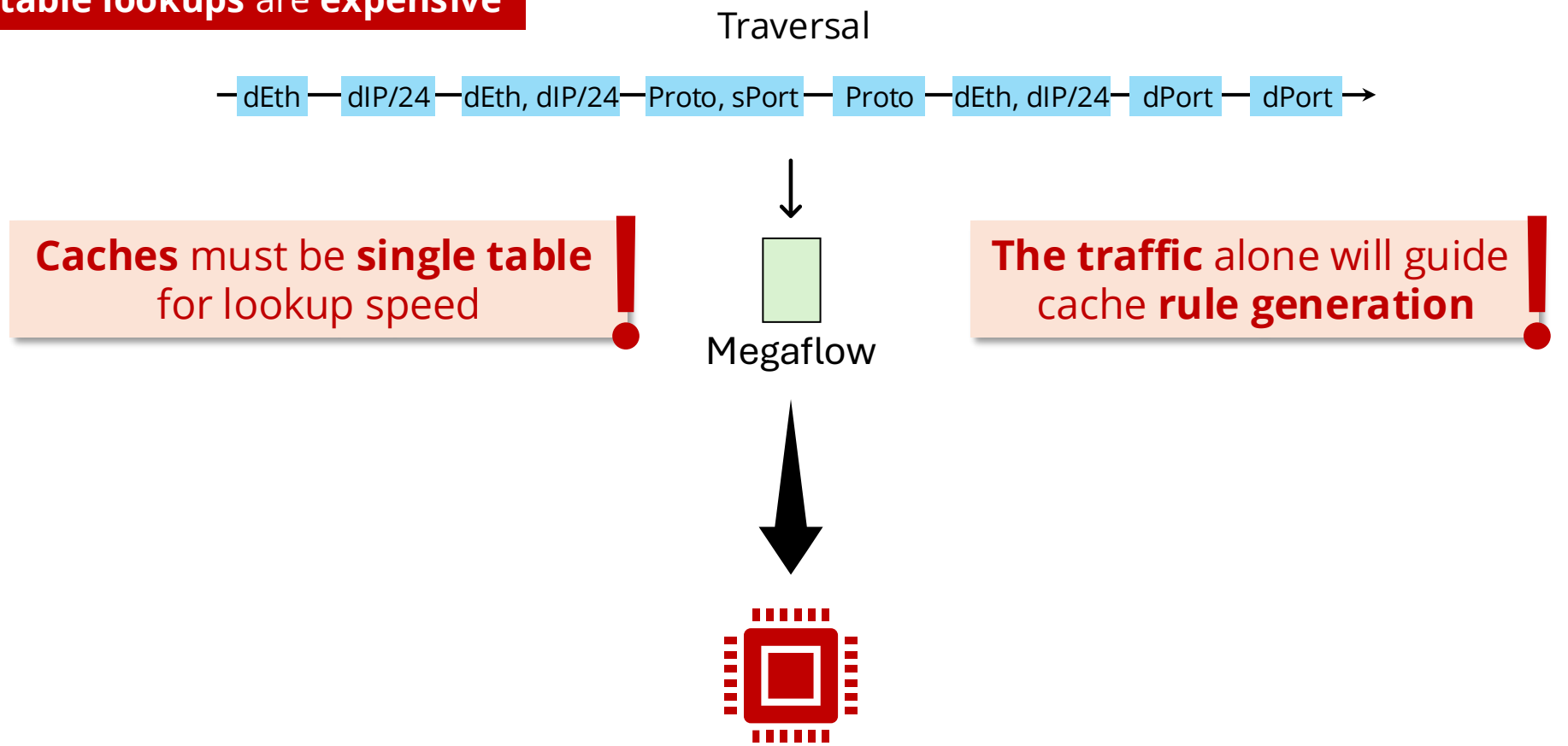
Virtual Switches in the Data Center

Multi-table lookups are **expensive**



Virtual Switches in the Data Center

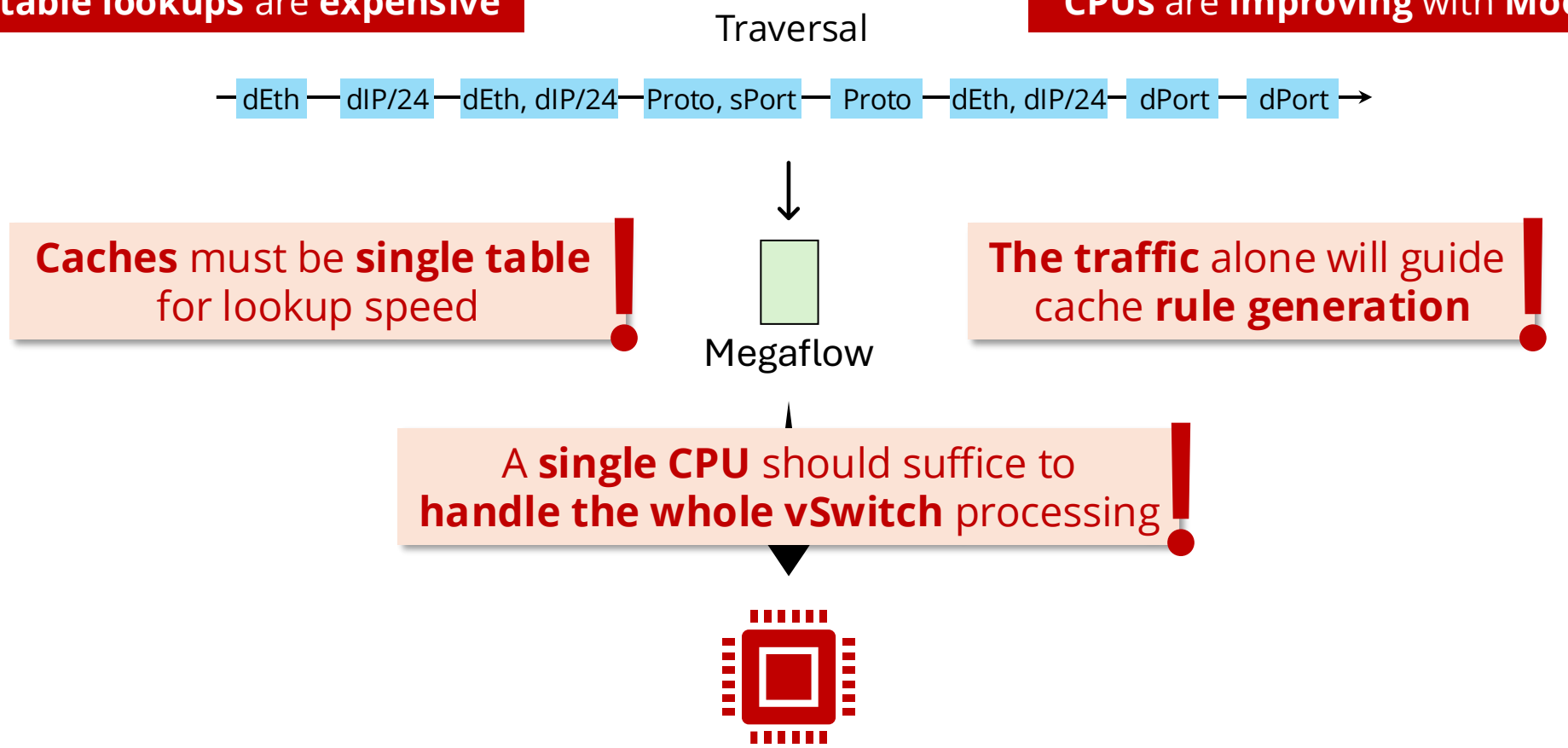
Multi-table lookups are **expensive**



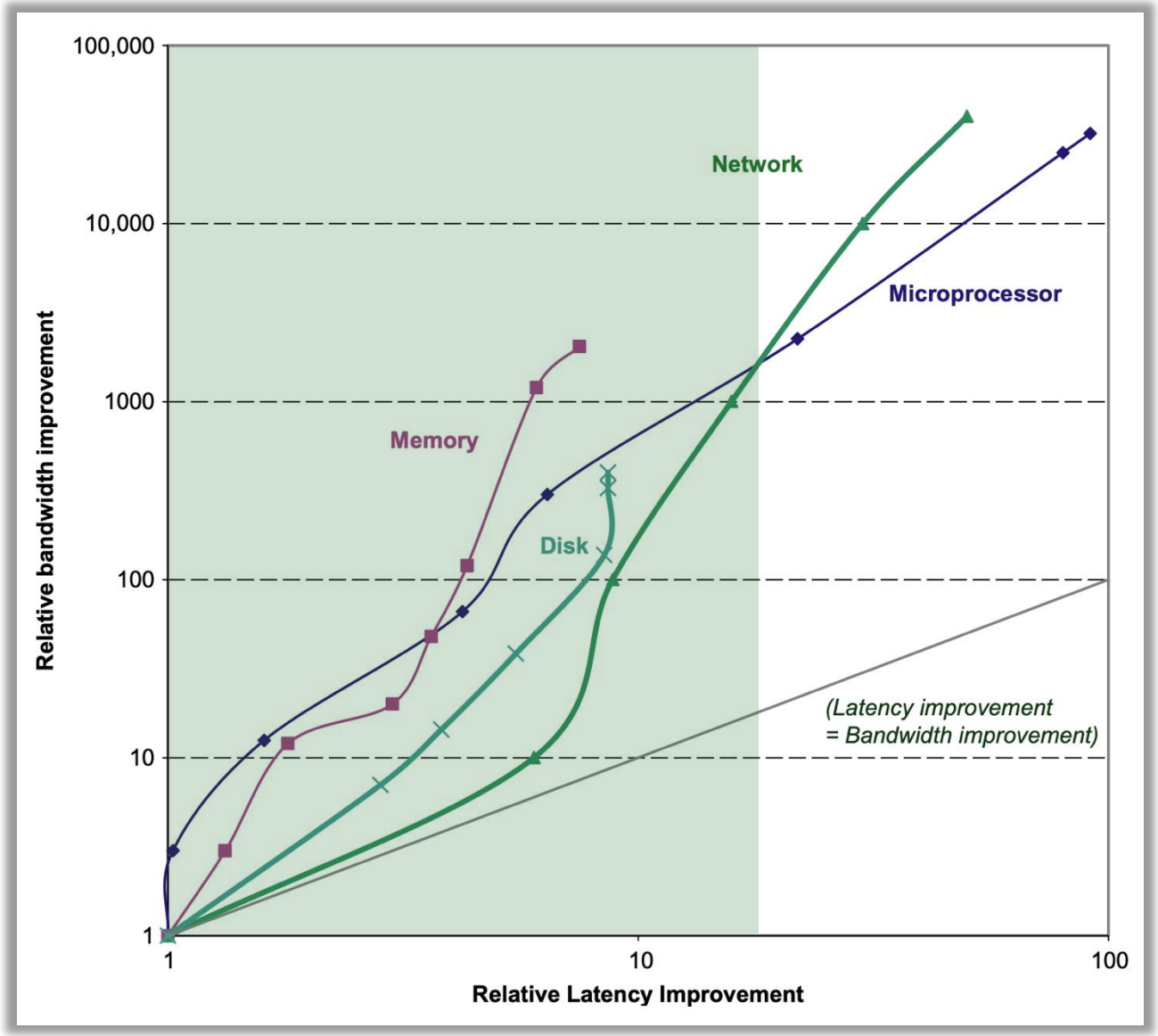
Virtual Switches in the Data Center

Multi-table lookups are **expensive**

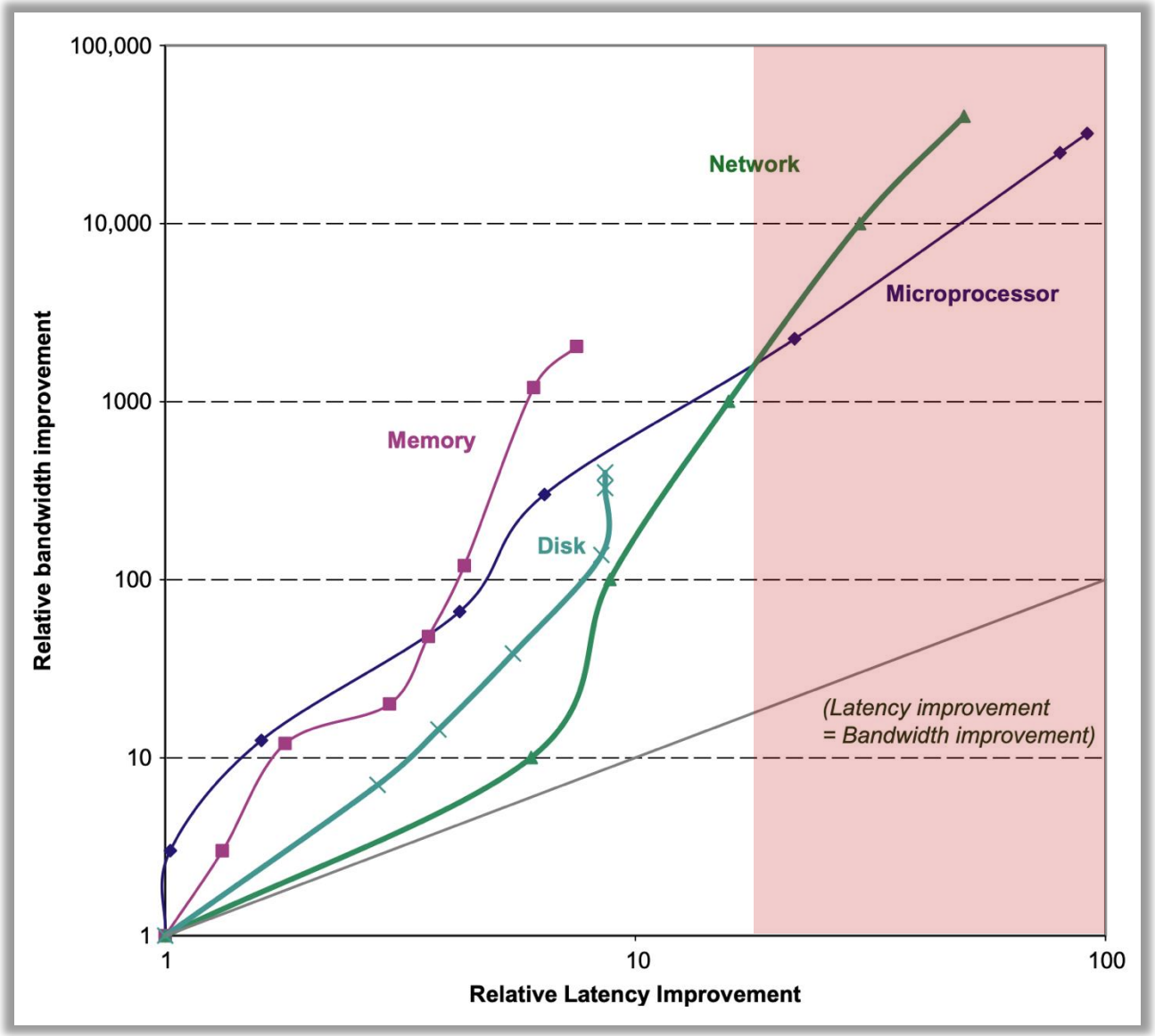
CPUs are **improving** with **Moore's law**



CPU performance has stagnated!



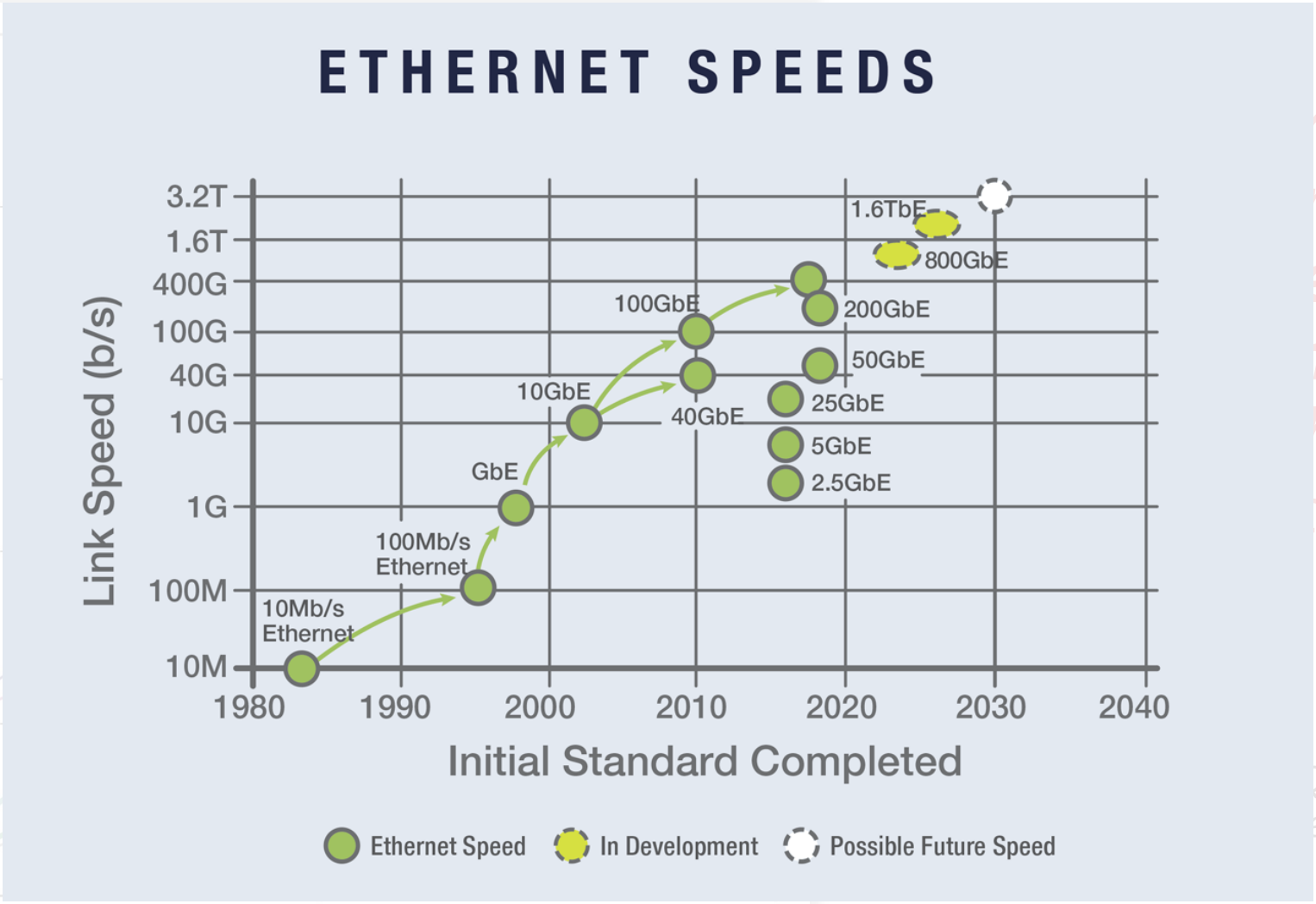
CPU performance has stagnated!



** "… ***Except for networking***, *we note that there were* ***modest improvements in latency and bandwidth*** *in the* ***other three technologies*** *in the six years since the last edition: 0%–23% in latency and 23%–70% in bandwidth"*

**Computer Architecture: A Quantitative Approach*
John Henessy and David Patterson (6th Edition)

CPU performance has stagnated!

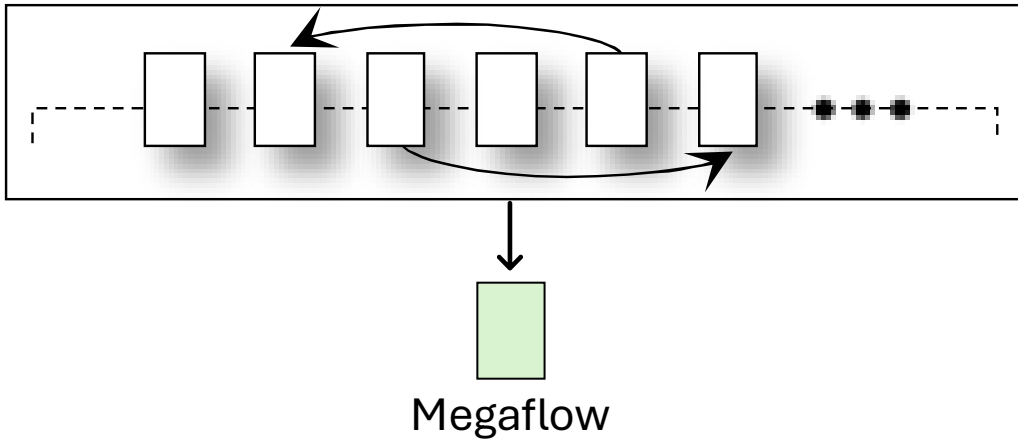


networking,
re were
ements in
***ndwidth** in the*
***hnologies** in*
e the last
in latency and
dwidth"

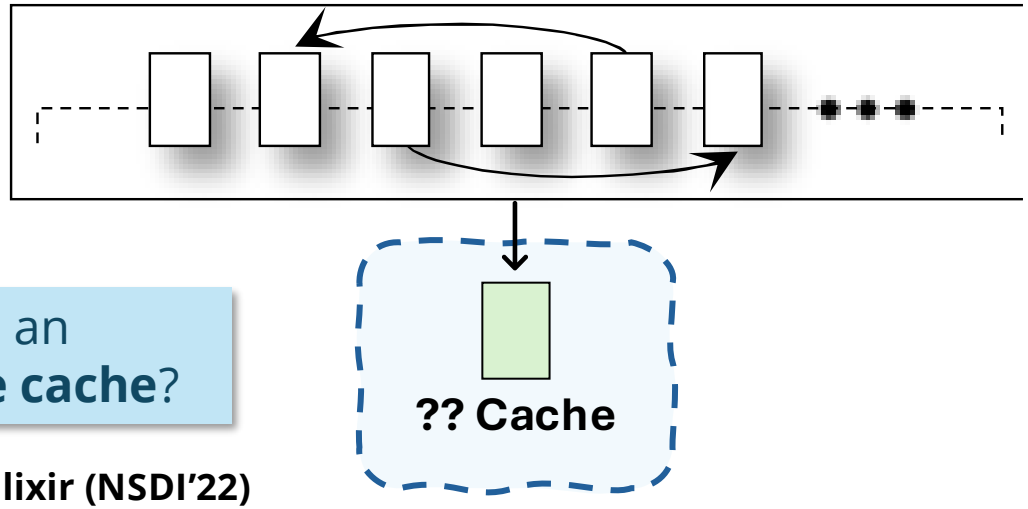
Quantitative Approach
erson (6th Edition)

[1] https://iebmedia.com/technology/industrial-ethernet/ethernet-celebrates-50-years-with-2023-roadmap-and-demo/

How to Rearchitect the vSwitch for Tbps Era?



How to Rearchitect the vSwitch for Tbps Era?



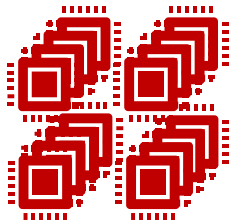
Q1: How to design an **efficient, high hit rate cache**?

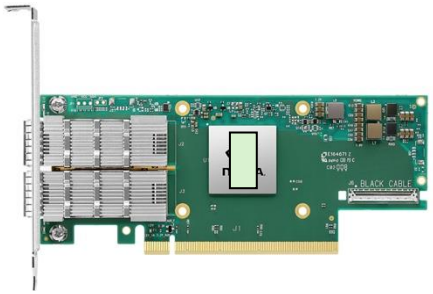
NuevomatchUP (NSDI'22)

"Do packet classification with ML on dozens of CPUs for line rate"

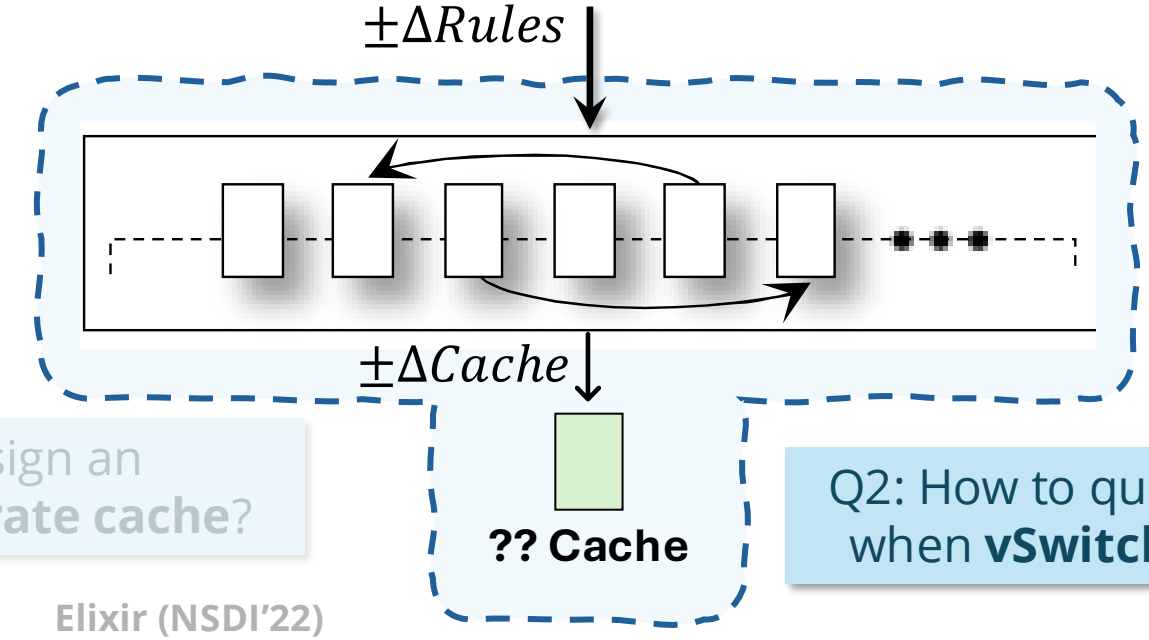
Elixir (NSDI'22)

"Offload subset of Megaflow to limited SmartNIC TCAM memory"





How to Rearchitect the vSwitch for Tbps Era?

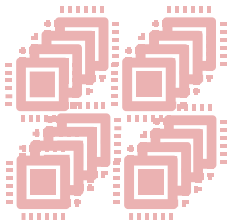


Q1: How to design an **efficient, high hit rate cache**?

Q2: How to quickly **evict stale cache** when **vSwitch rules are updated?**

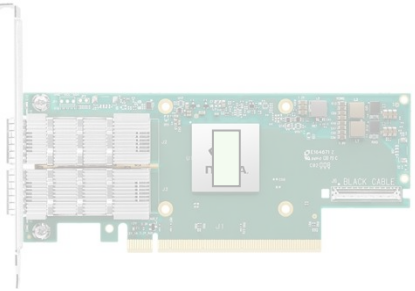
NuevomatchUP (NSDI'22)

"Do packet classification with ML on dozens of CPUs for line rate"

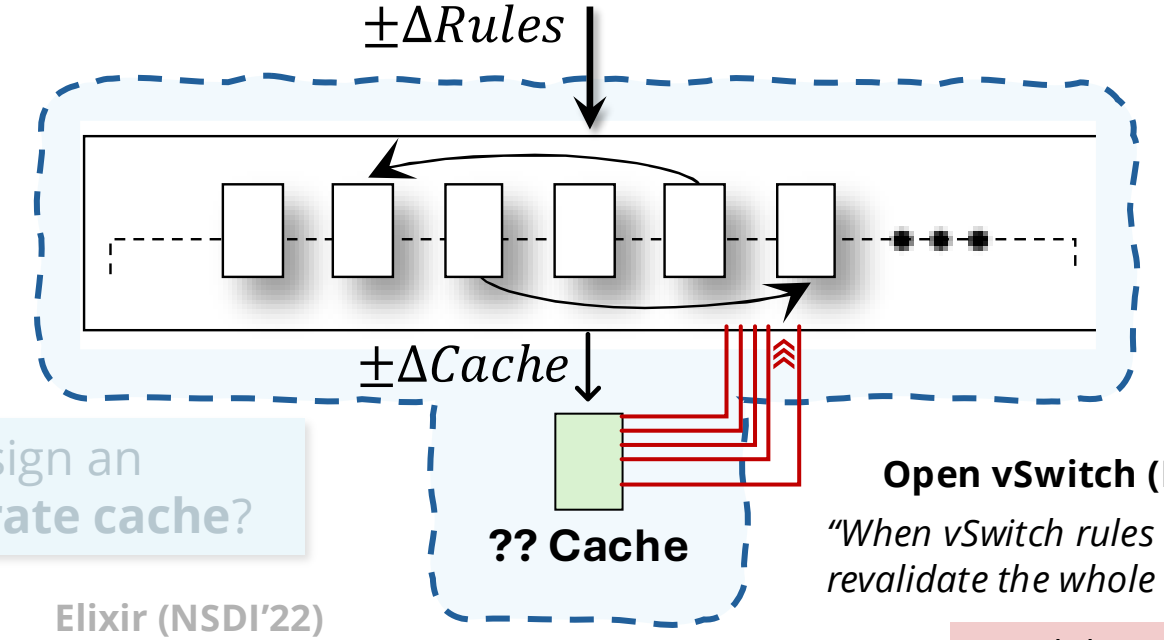


Elixir (NSDI'22)

"Offload subset of Megaflow to limited SmartNIC TCAM memory"



How to Rearchitect the vSwitch for Tbps Era?



Q1: How to design an **efficient, high hit rate cache**?

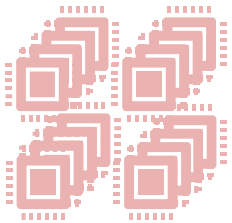
Open vSwitch (NSDI'15)

"When vSwitch rules are updated, revalidate the whole cache again"

Revalidation Cost
$\boldsymbol{O(Entries)}$

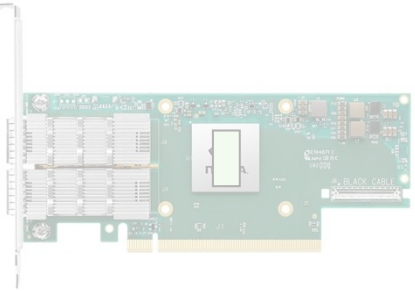
NuevomatchUP (NSDI'22)

"Do packet classification with ML on dozens of CPUs for line rate"



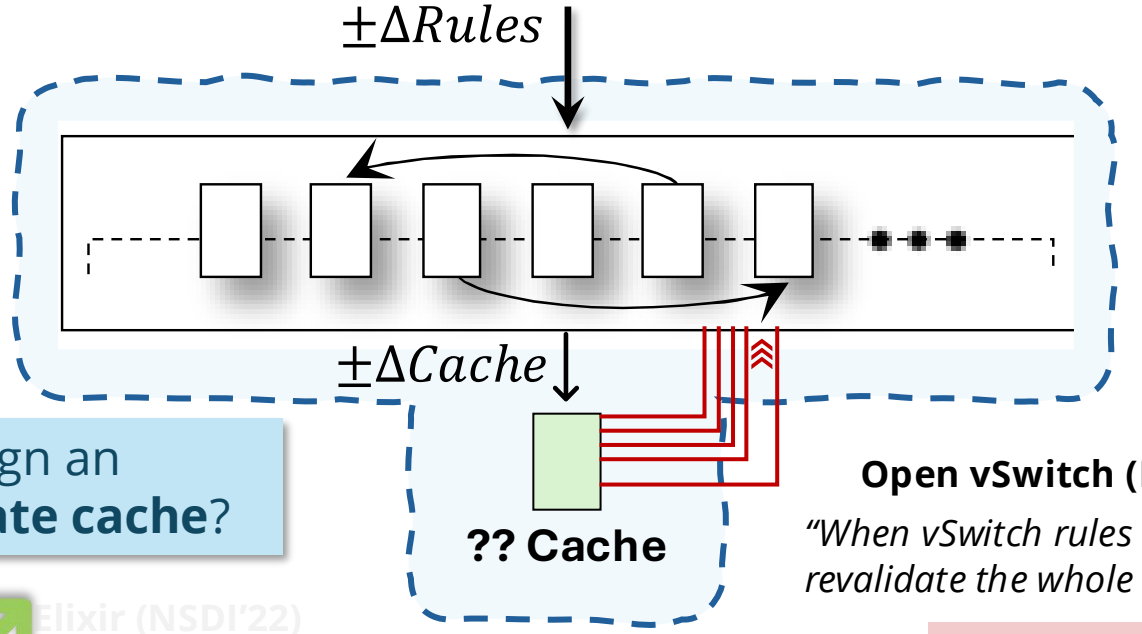
Elixir (NSDI'22)

"Offload subset of Megaflow to limited SmartNIC TCAM memory"



Q2: How to quickly **evict stale cache** when **vSwitch rules are updated?**

How to Rearchitect the vSwitch for Tbps Era?



Q1: How to design an **efficient, high hit rate cache**?

Open vSwitch (NSDI'15)

"When vSwitch rules are updated, revalidate the whole cache again"

Revalidation Cost
$\boldsymbol{O(Entries)}$

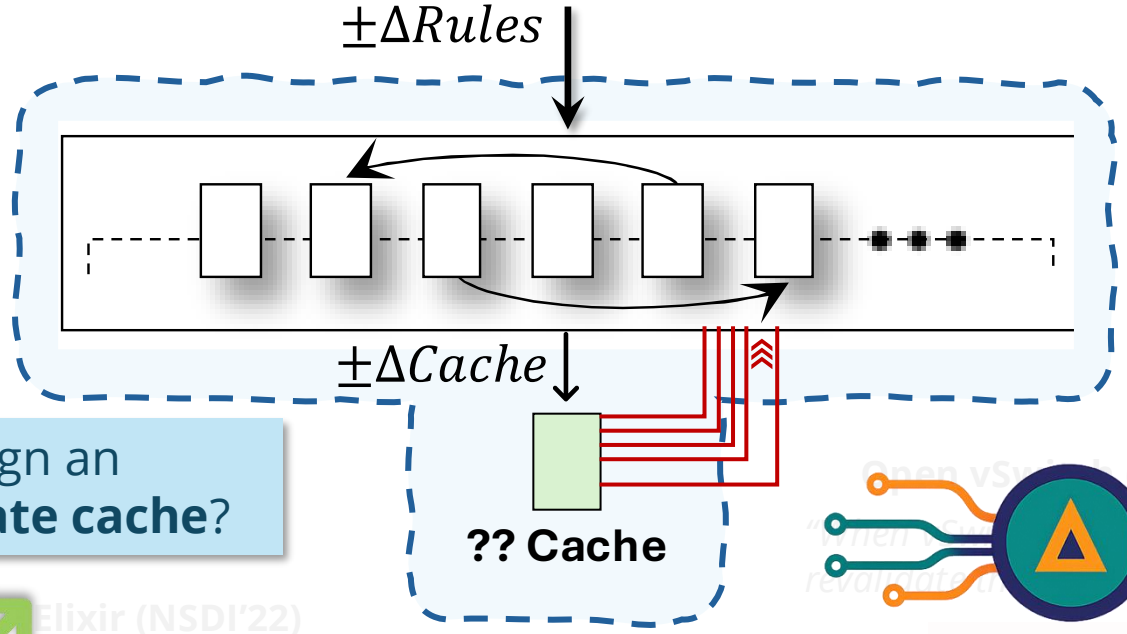
Q2: How to quickly **evict stale cache** when **vSwitch rules are updated?**

NuevomatchUP (NSDI'22) Elixir (NSDI'22)

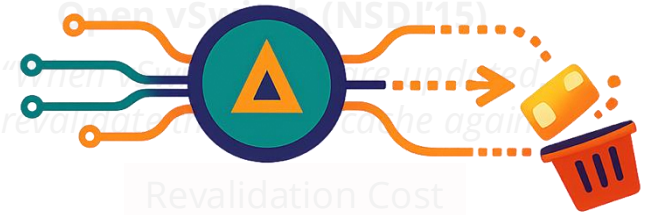
"Do packet classificatio… offload subset of Megaflow to on dozens of CPUs for … SmartNIC TCAM memory"

GvS
GIGAFLOW
VIRTUAL SWITCH

How to Rearchitect the vSwitch for Tbps Era?



Q1: How to design an **efficient, high hit rate cache**?

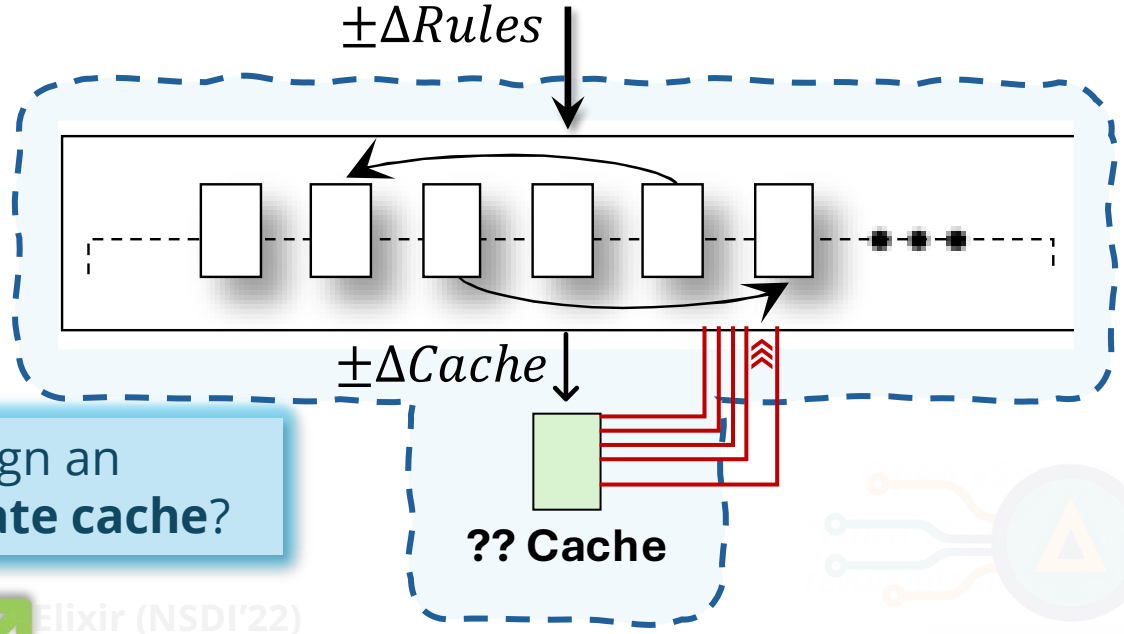


Kairo Cache Eviction

Q2: How to quickly **evict stale cache** when **vSwitch rules are updated?**

GvS
GIGAFLOW
VIRTUAL SWITCH

How to Rearchitect the vSwitch for Tbps Era?



Q1: How to design an **efficient, high hit rate cache**?

GvS
GIGAFLOW
VIRTUAL SWITCH

HOT CHIPS ASPLOS 2025

Google Summer of Code

Gigaflow: Pipeline-Aware Sub-Traversal Caching for Modern SmartNICs

Annus Zulfiqar

Ali Imran, Venkat Kunaparaju, Ben Pfaff,

Gianni Antichi, Muhammad Shahbaz

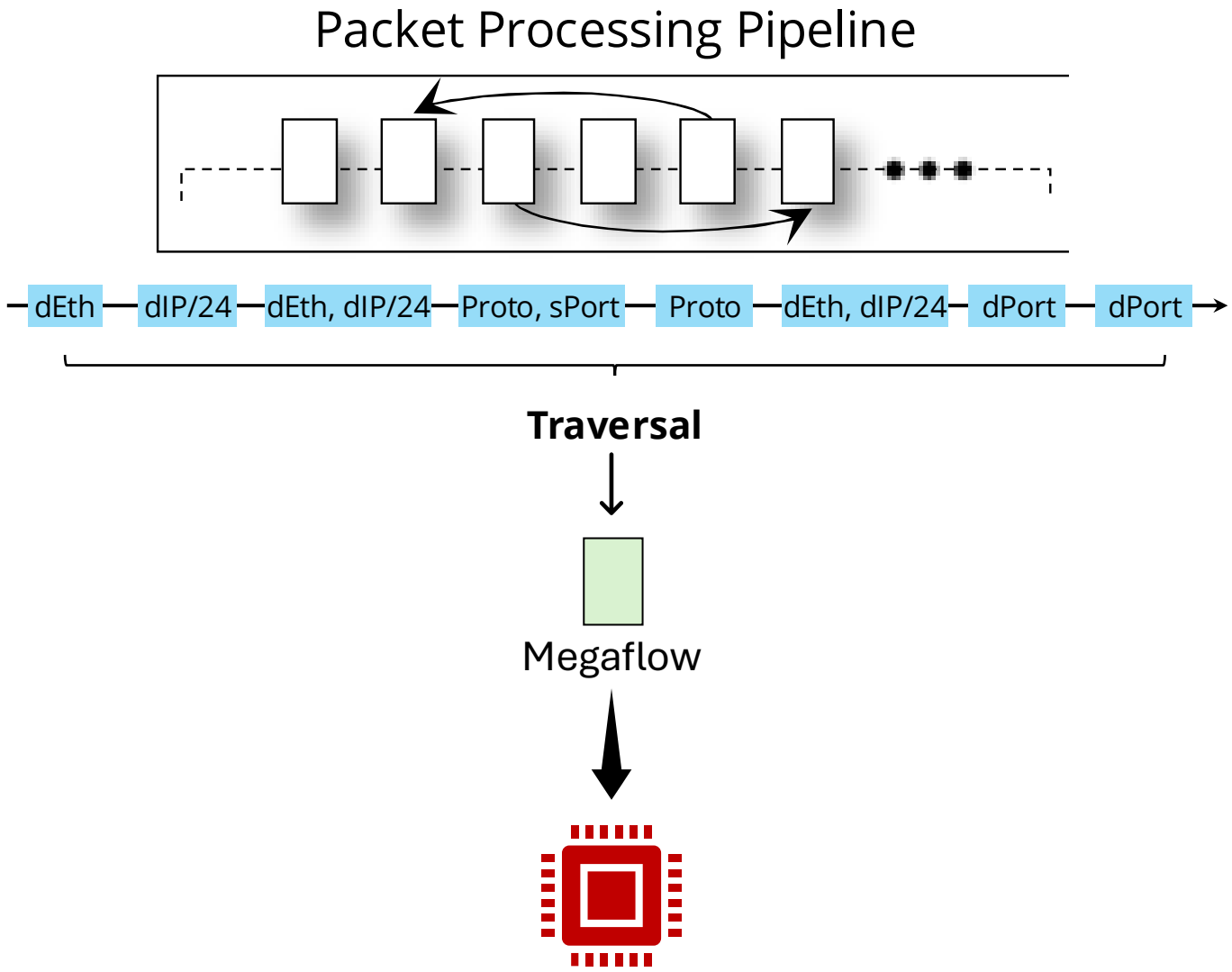
UNIVERSITY OF MICHIGAN

PURDUE UNIVERSITY

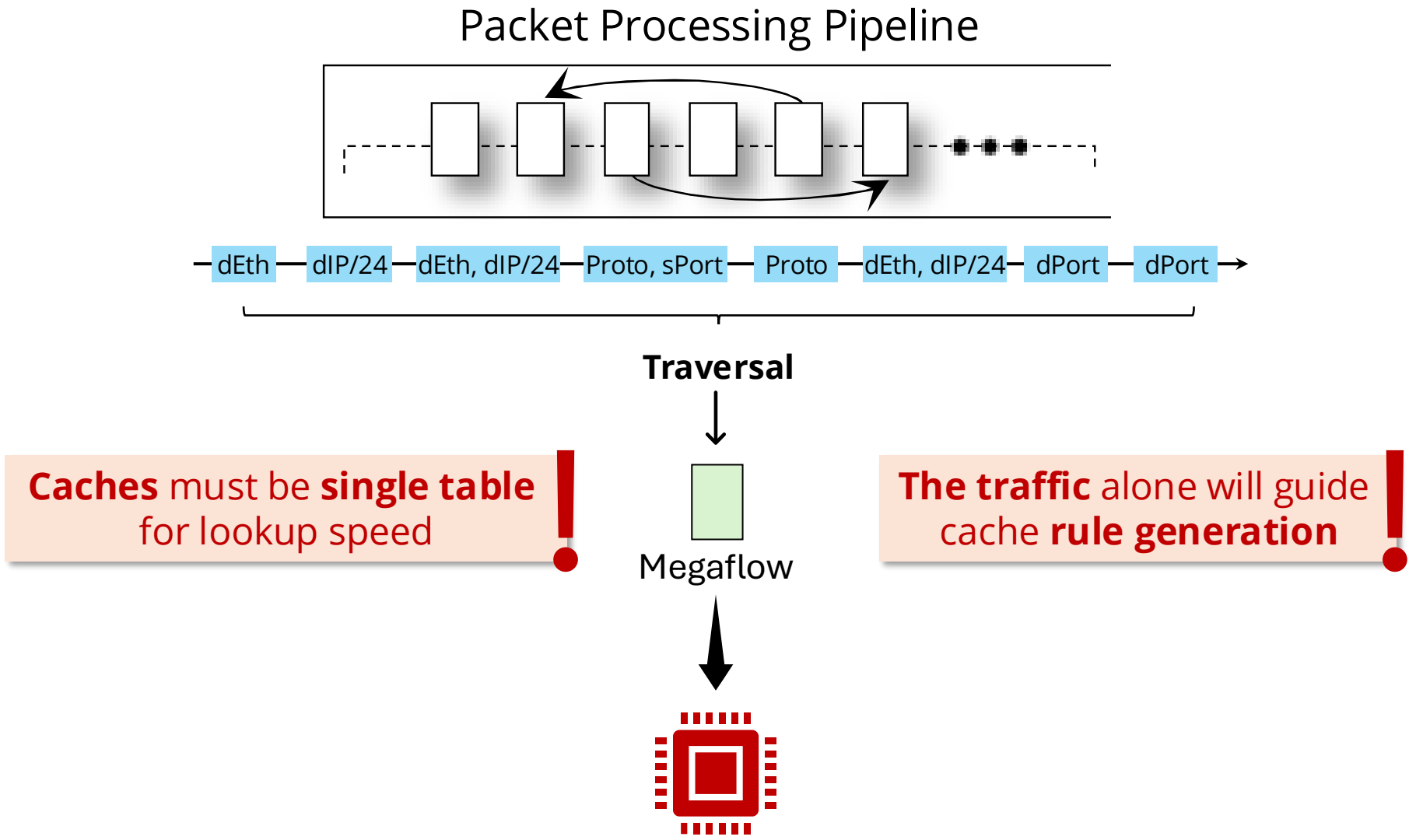
feldera

POLITECNICO MILANO 1863

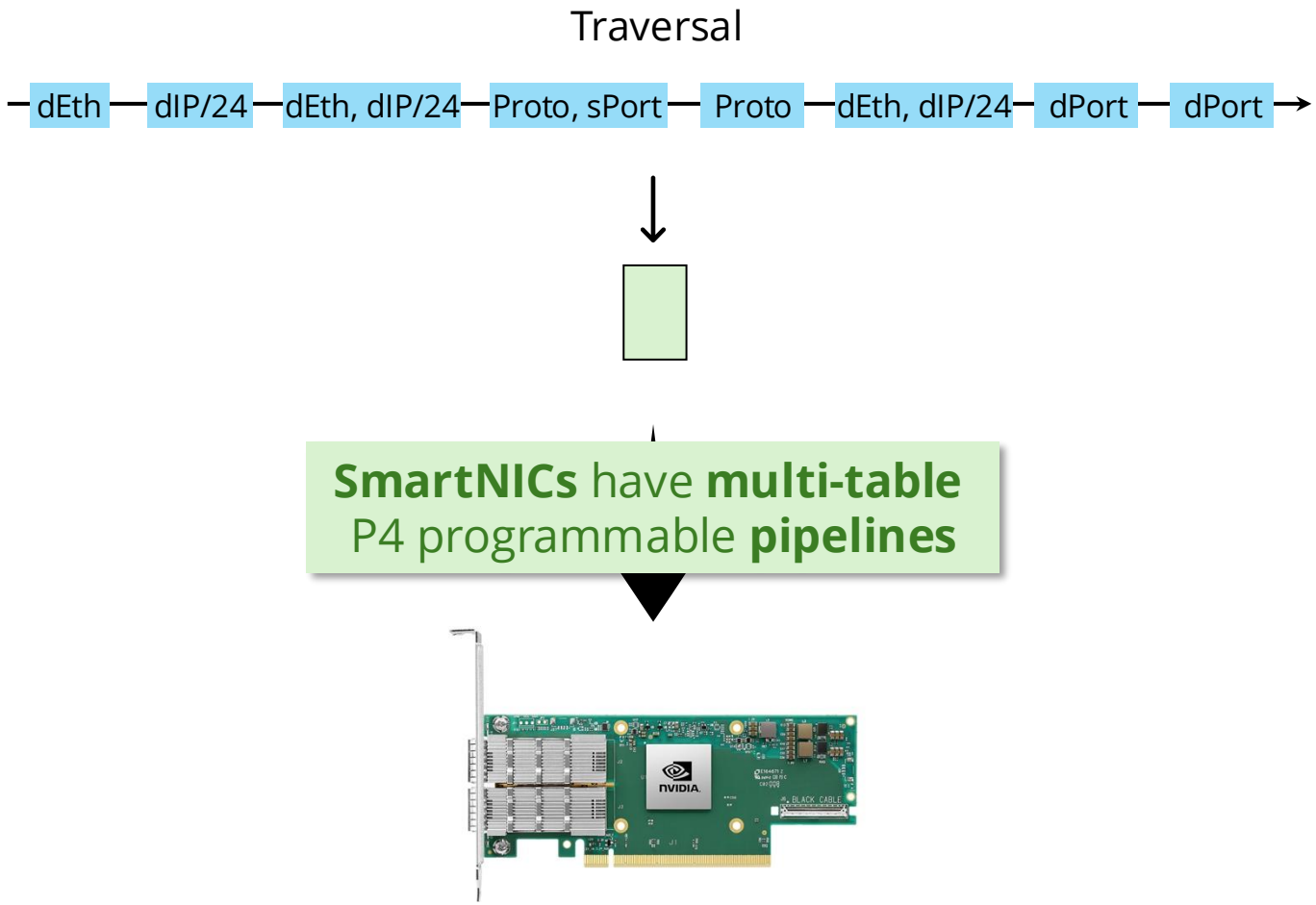
Virtual Switches in the Data Center



Virtual Switches in the Data Center



The Target Architecture has Changed

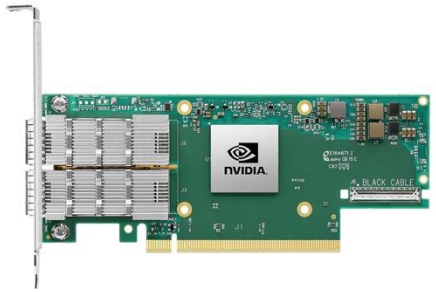


The Target Architecture has Changed

Traversal

dEth — dIP/24 — dEth, dIP/24 — Proto, sPort — Proto — dEth, dIP/24 — dPort — dPort →

It's time for a **clean slate** **vSwitch cache** design

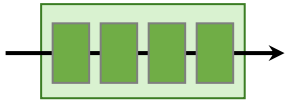


Offload the vSwitch to SmartNIC Tables?

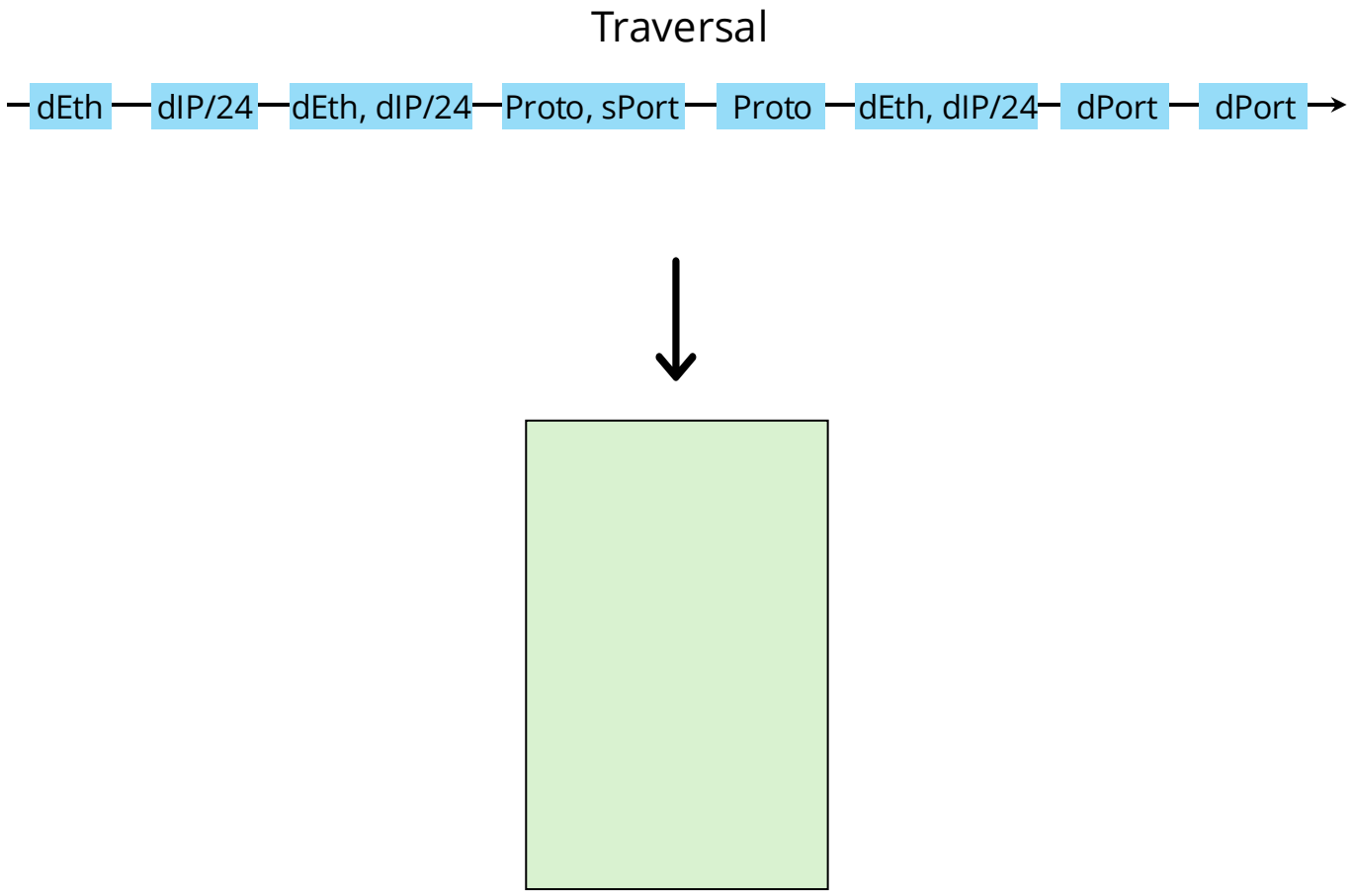
vSwitches provide an **infinite resource abstraction** to the controller

We **can't limit** the **vSwitch** packet pipeline's **flexibility**!!

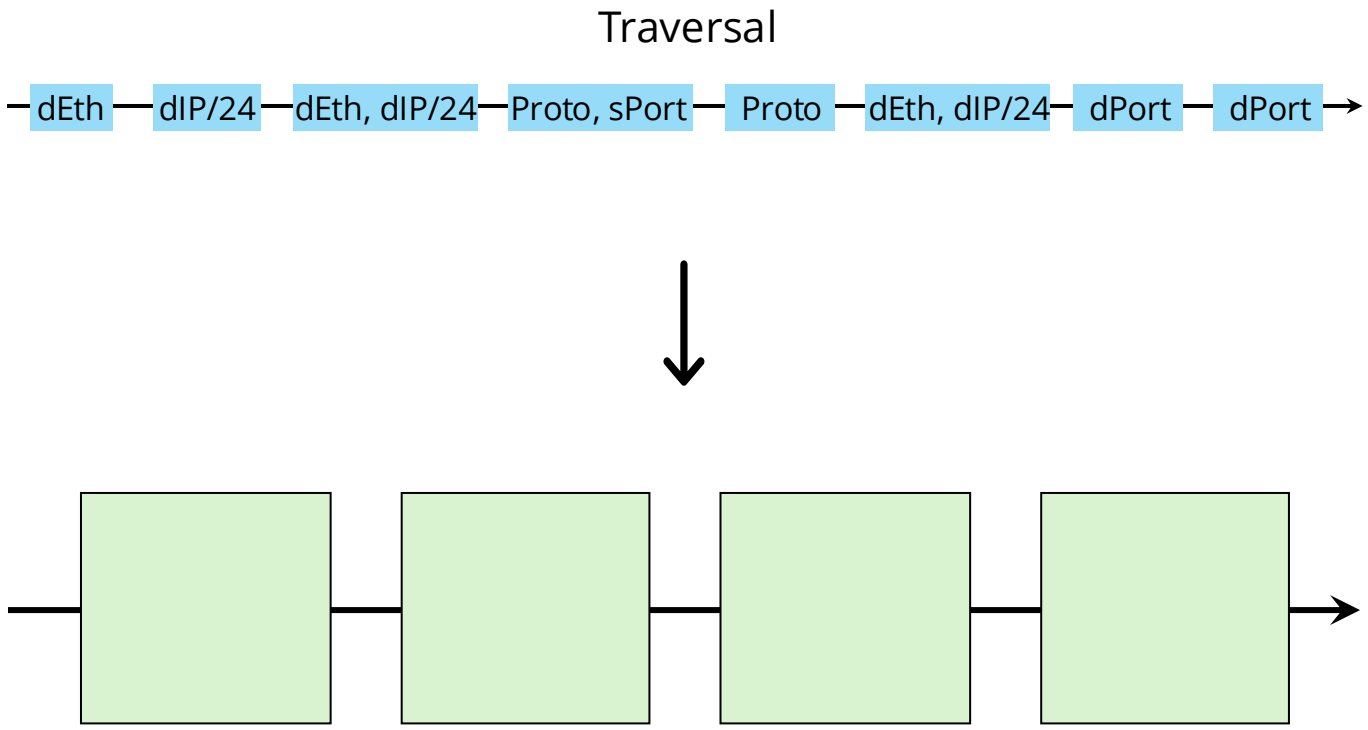
SmartNIC pipelines have a **handful of tables** and offer **limited control flow**



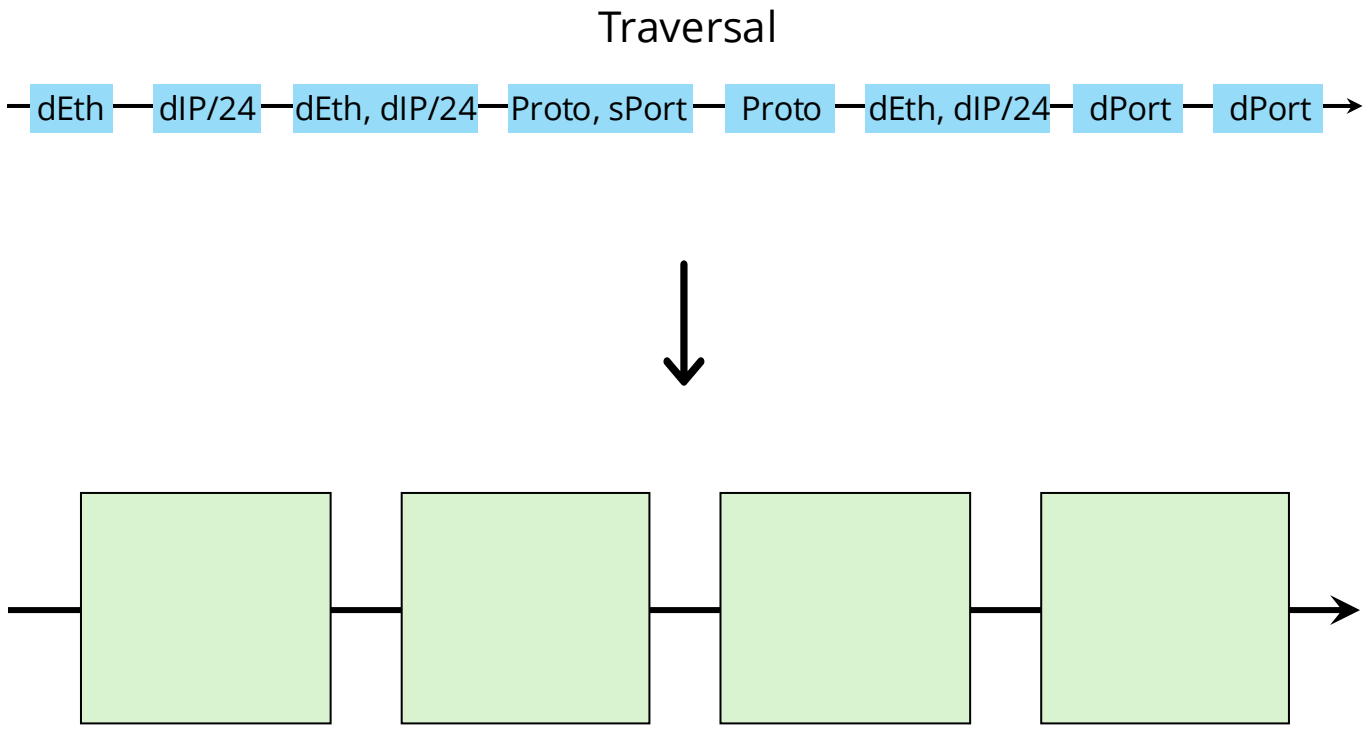
Cache Localities with Multiple Tables!



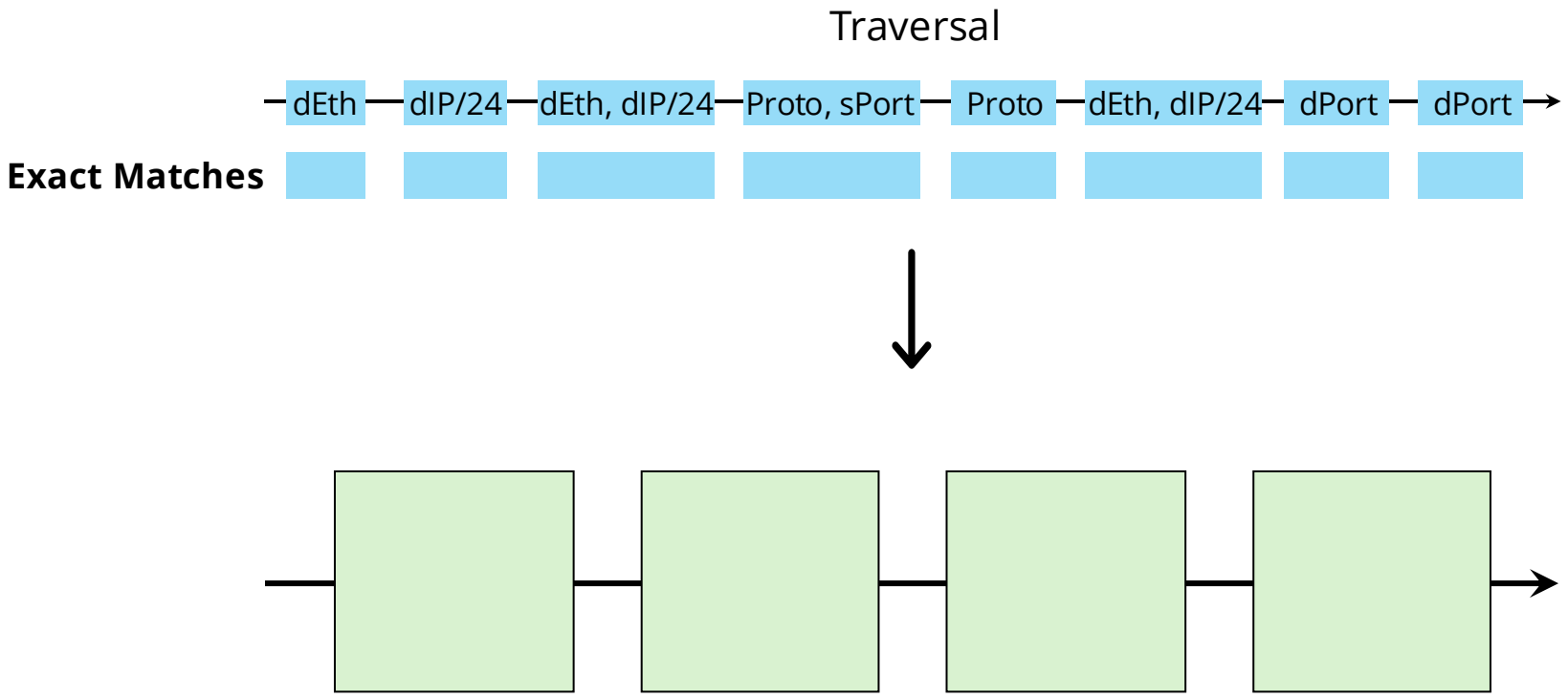
Cache Localities with Multiple Tables!



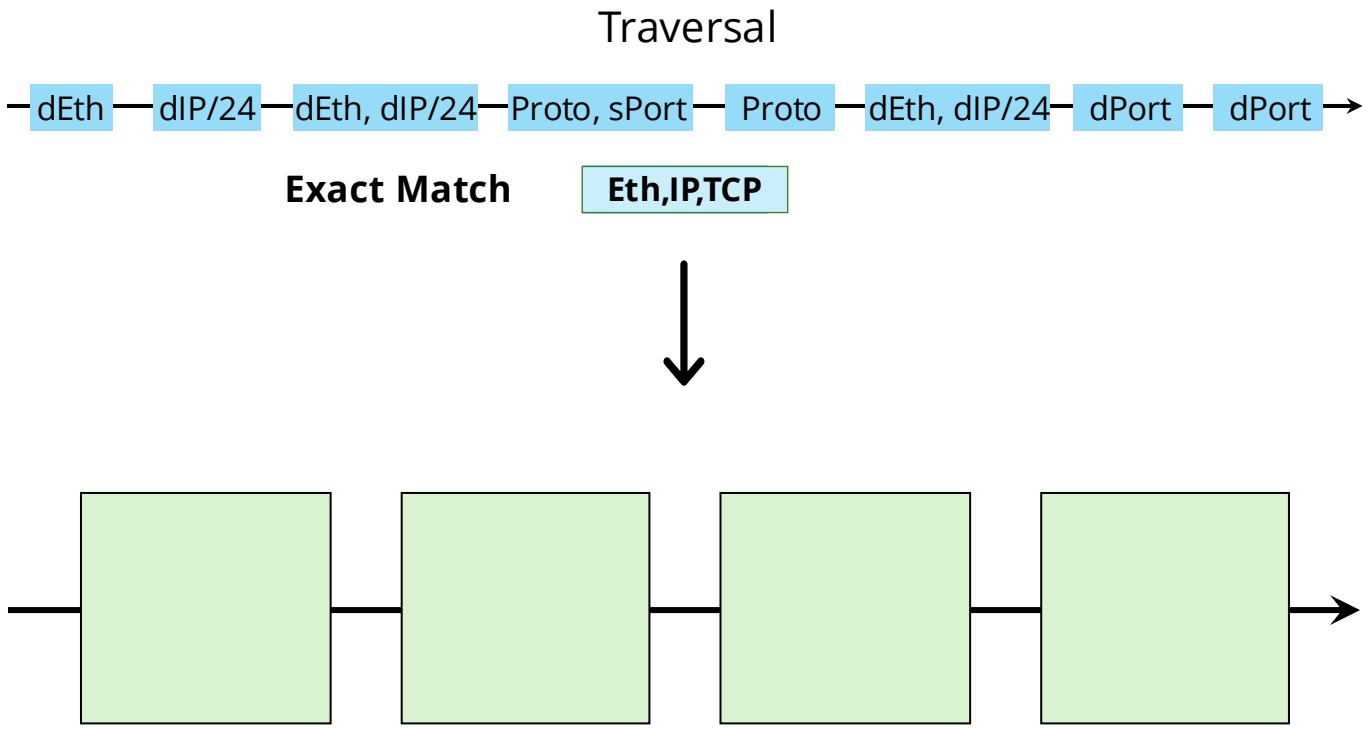
Caching *Temporal Locality* in the SmartNIC



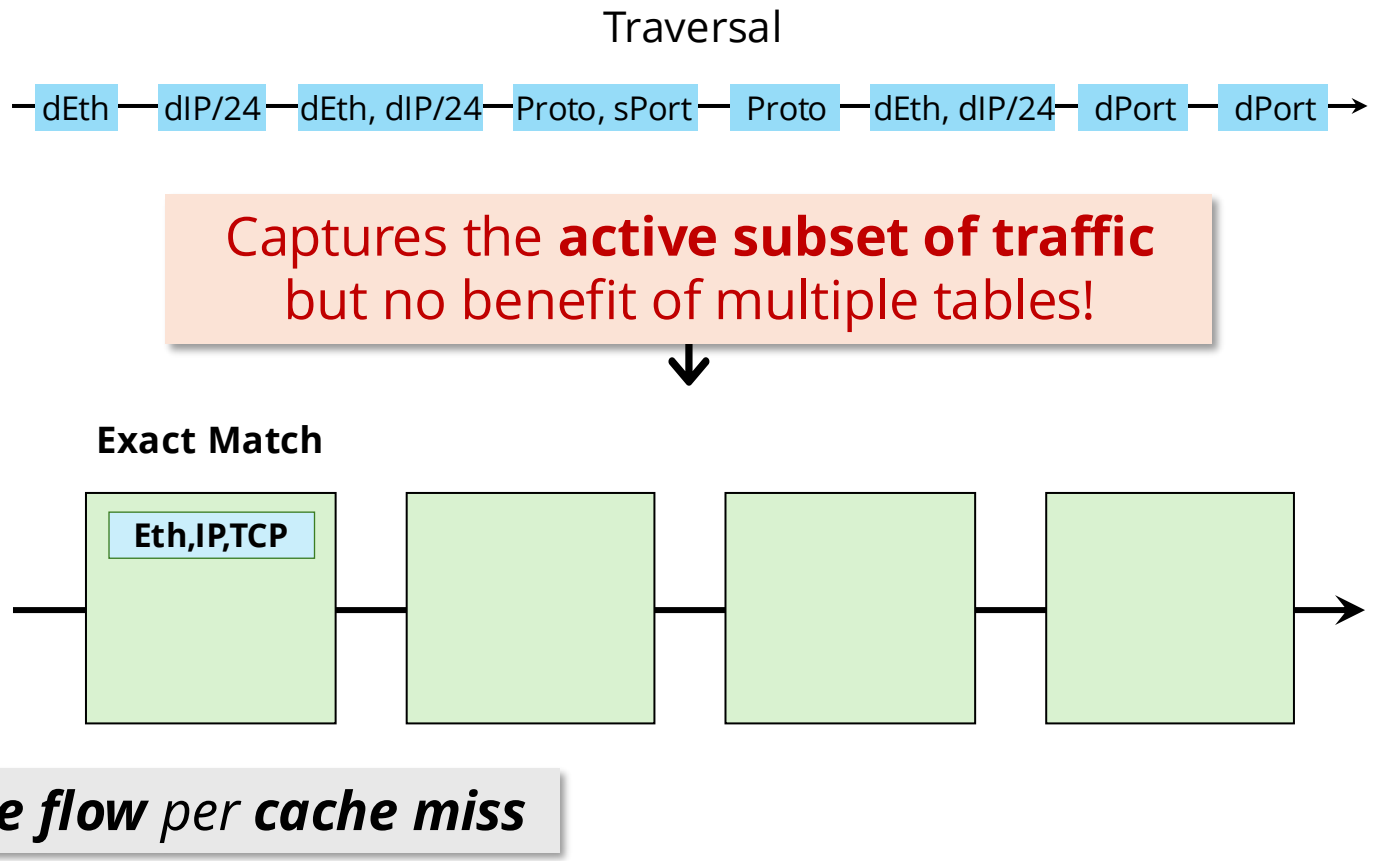
Caching *Temporal Locality* in the SmartNIC



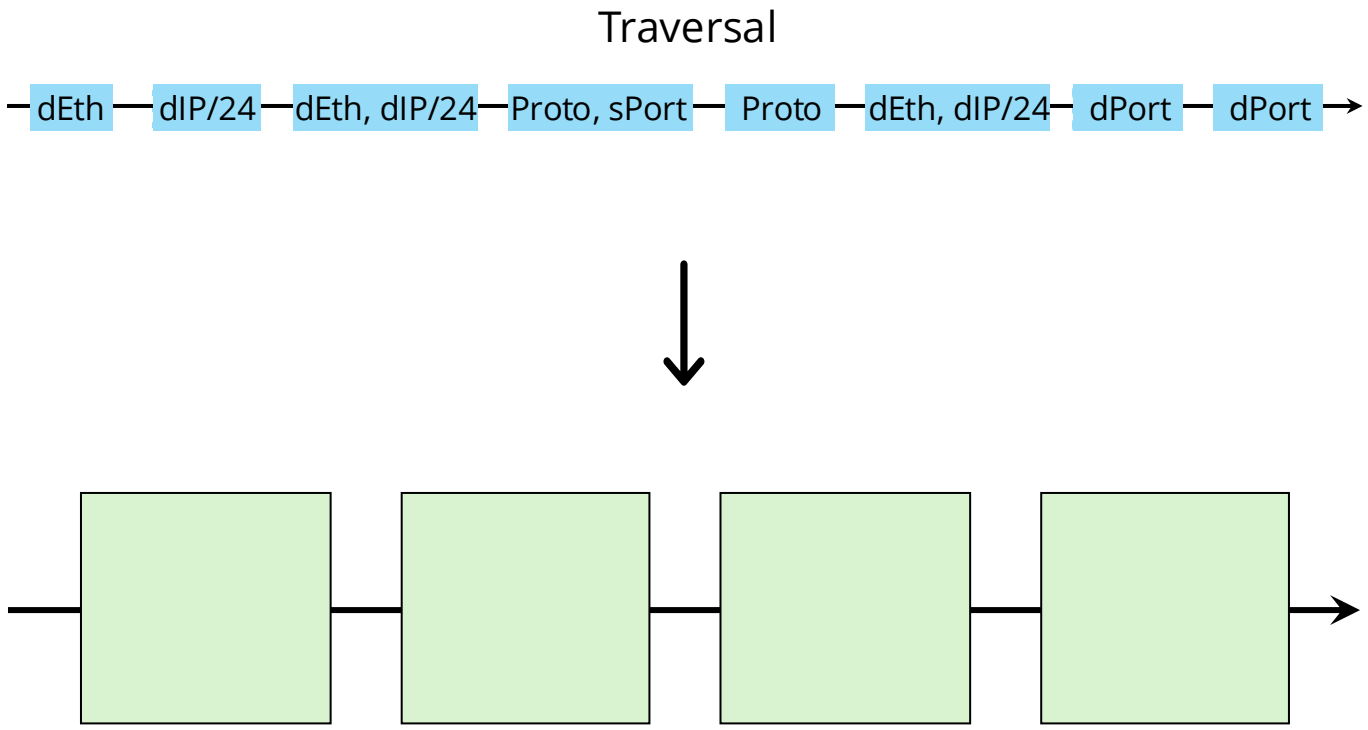
Caching *Temporal Locality* in the SmartNIC



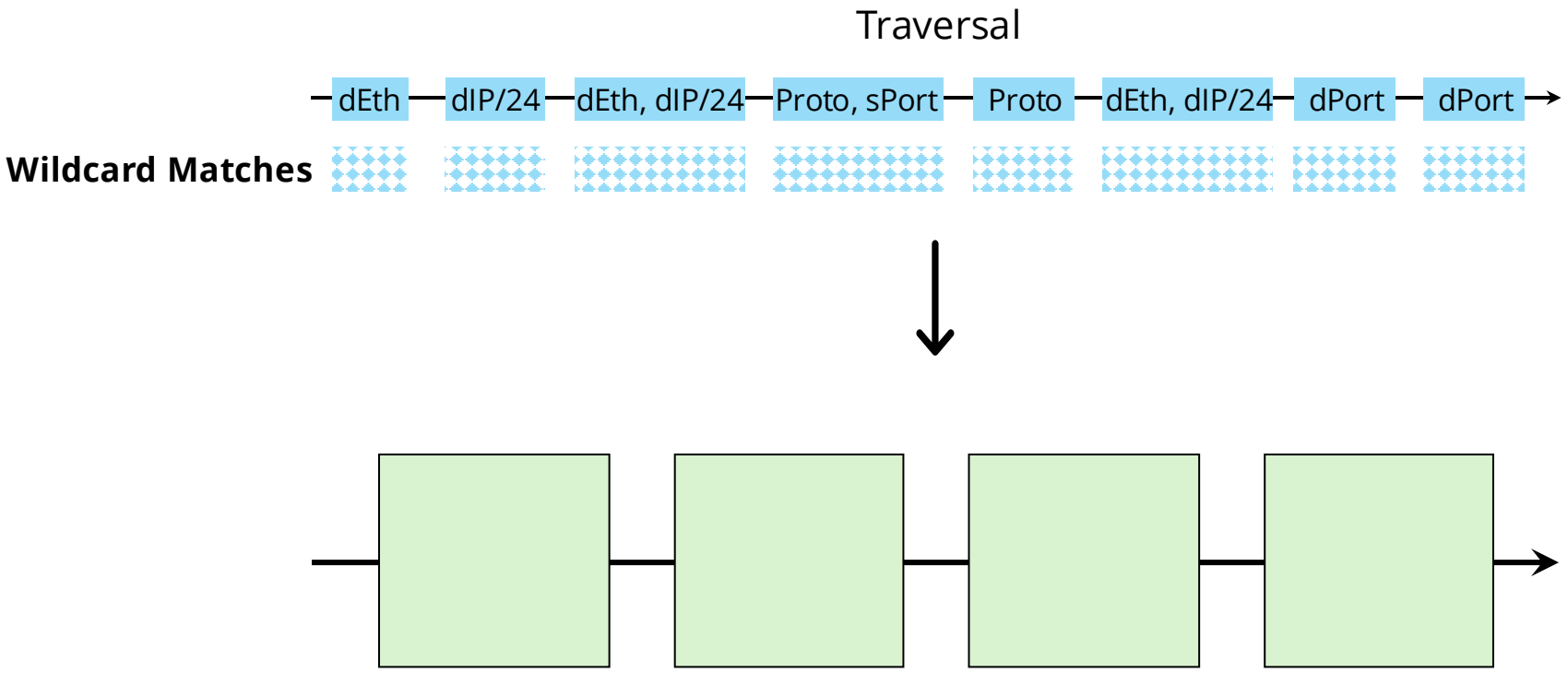
Caching *Temporal Locality* in the SmartNIC



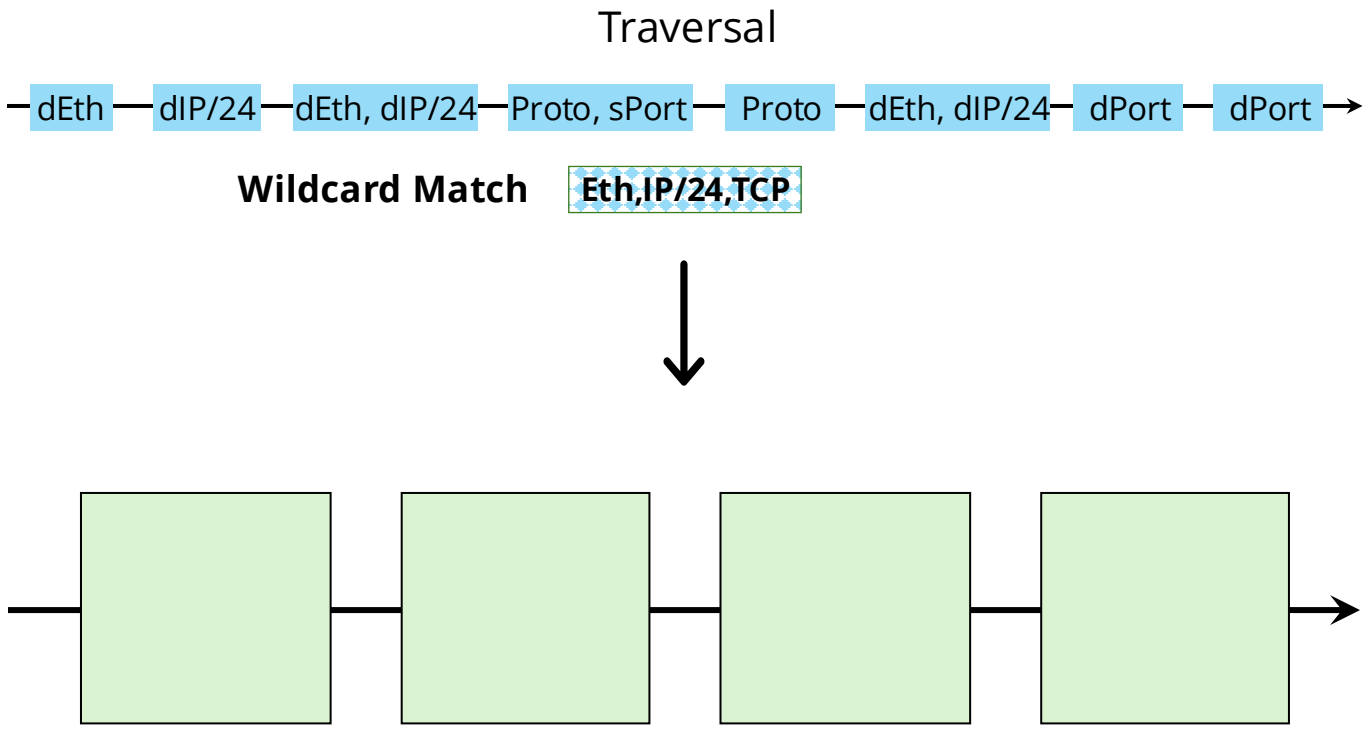
Caching *Spatial Locality* in the SmartNIC



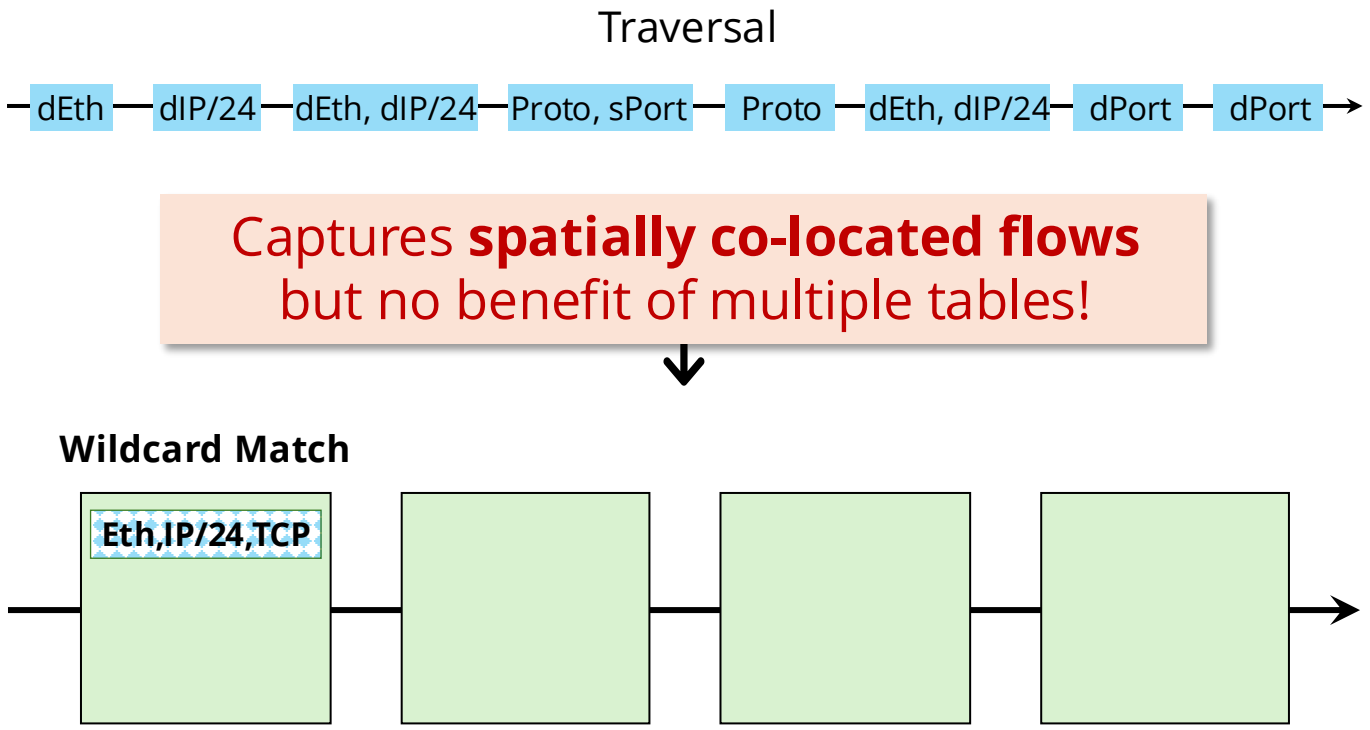
Caching *Spatial Locality* in the SmartNIC



Caching *Spatial Locality* in the SmartNIC

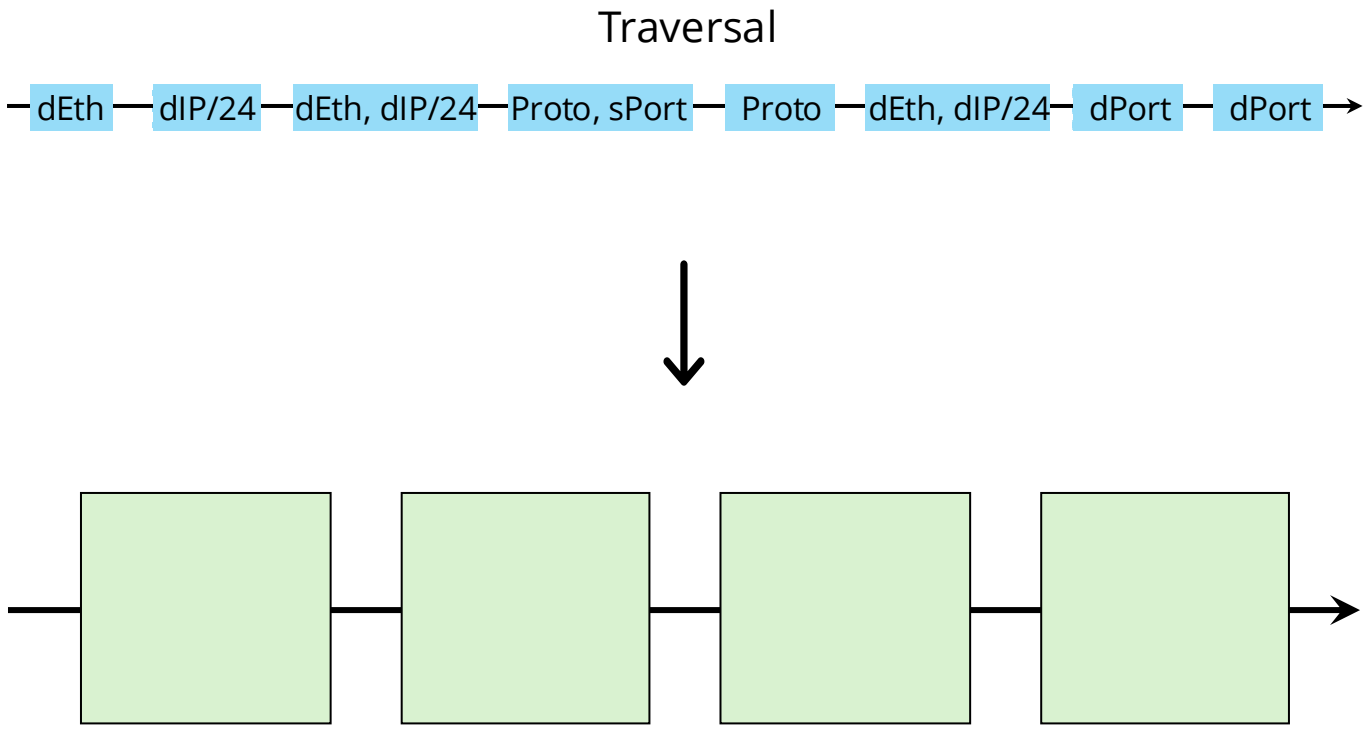


Caching *Spatial Locality* in the SmartNIC

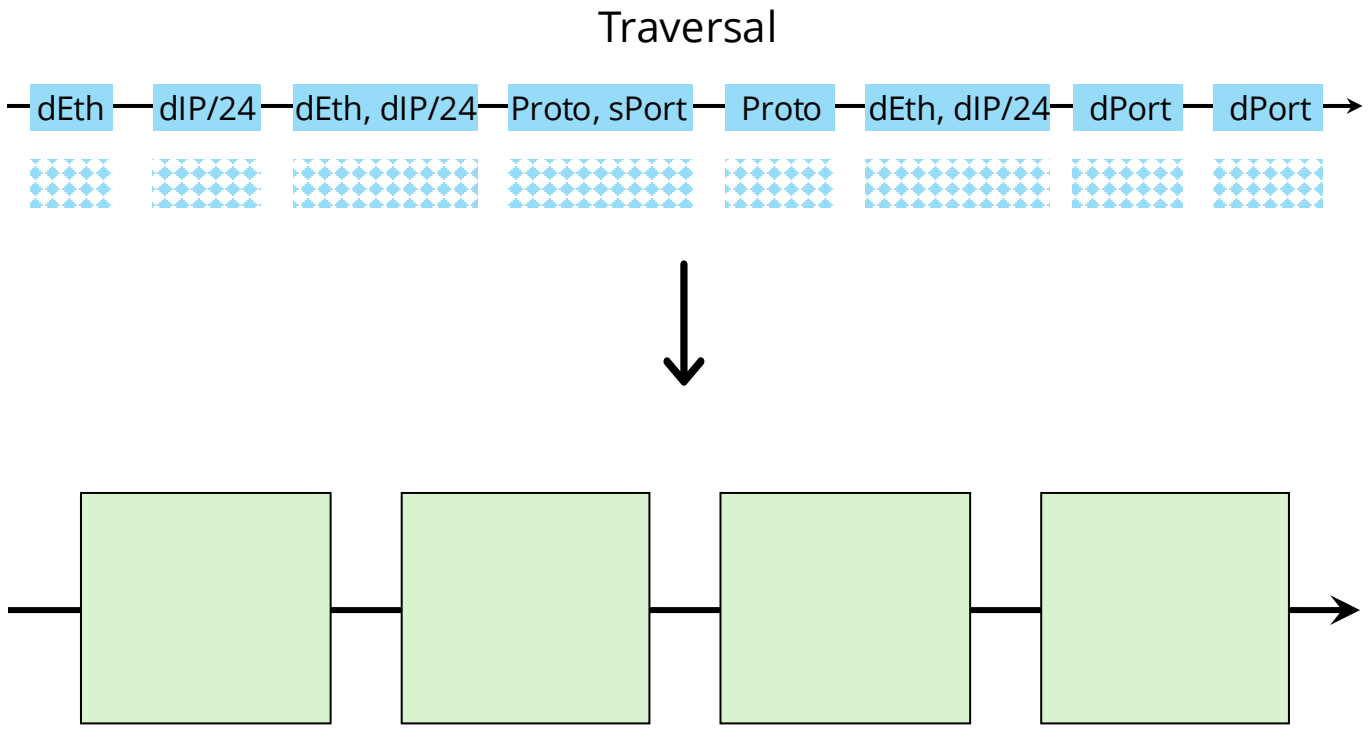


One traversal** per **cache miss
Traversal = set of flows

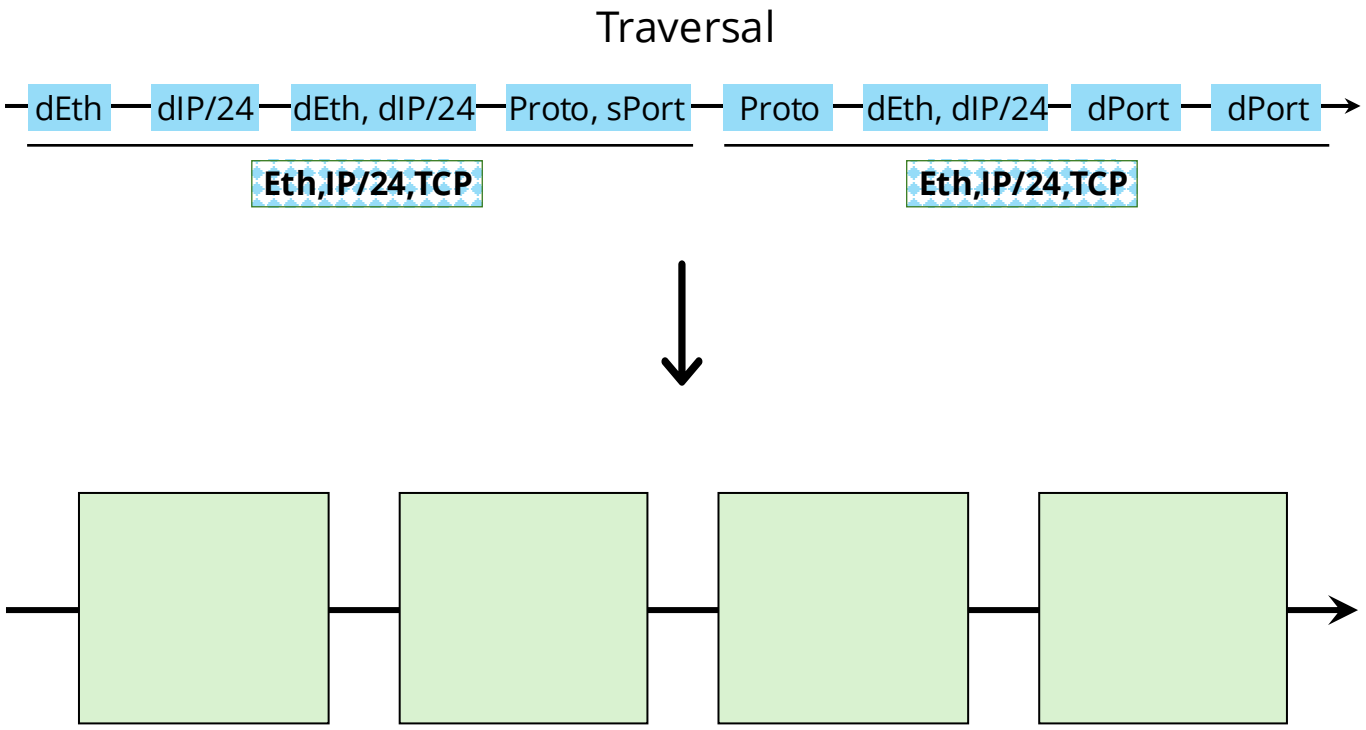
How about *Partitioning* the Traversal?



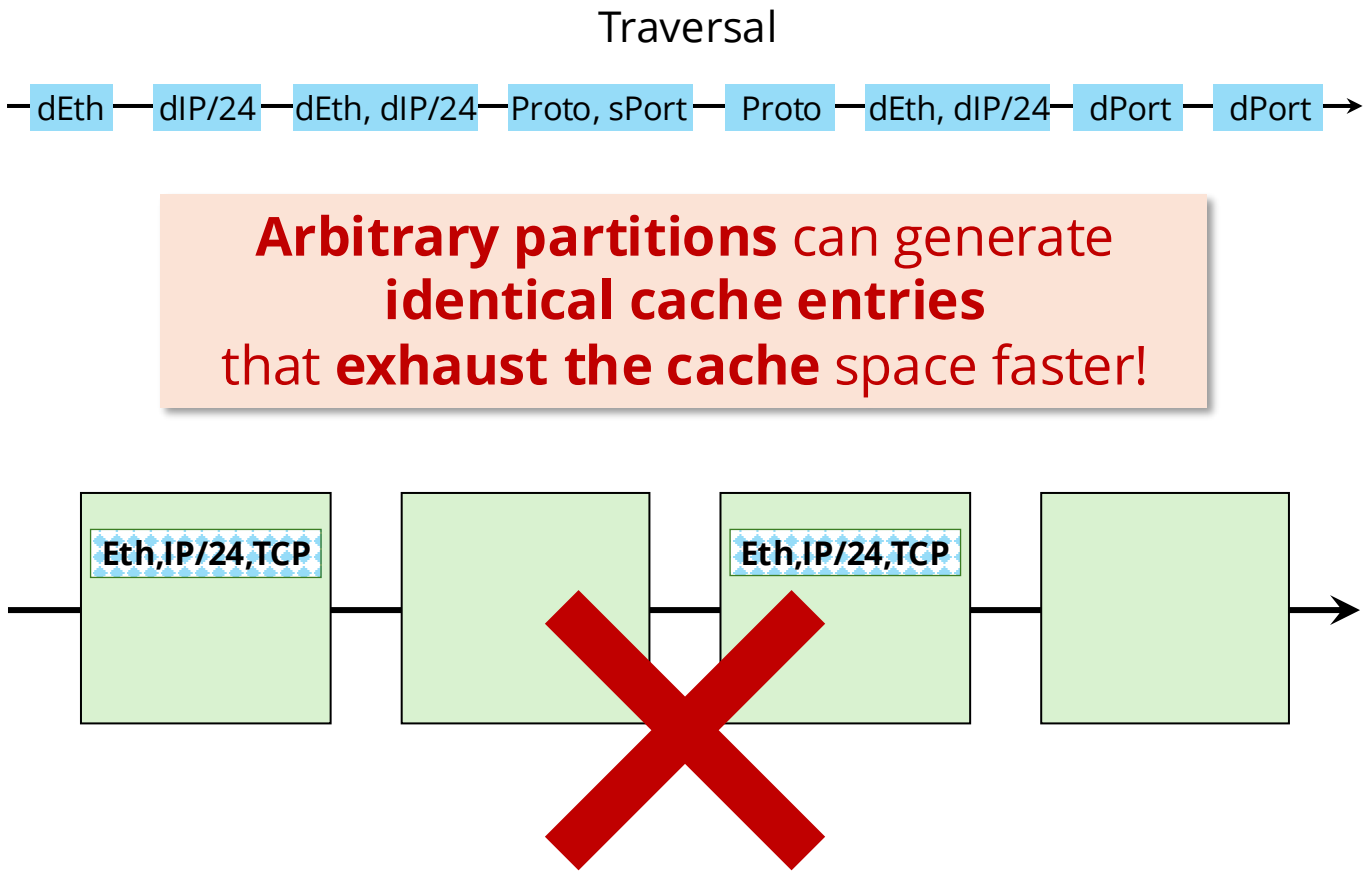
How about *Partitioning* the Traversal?



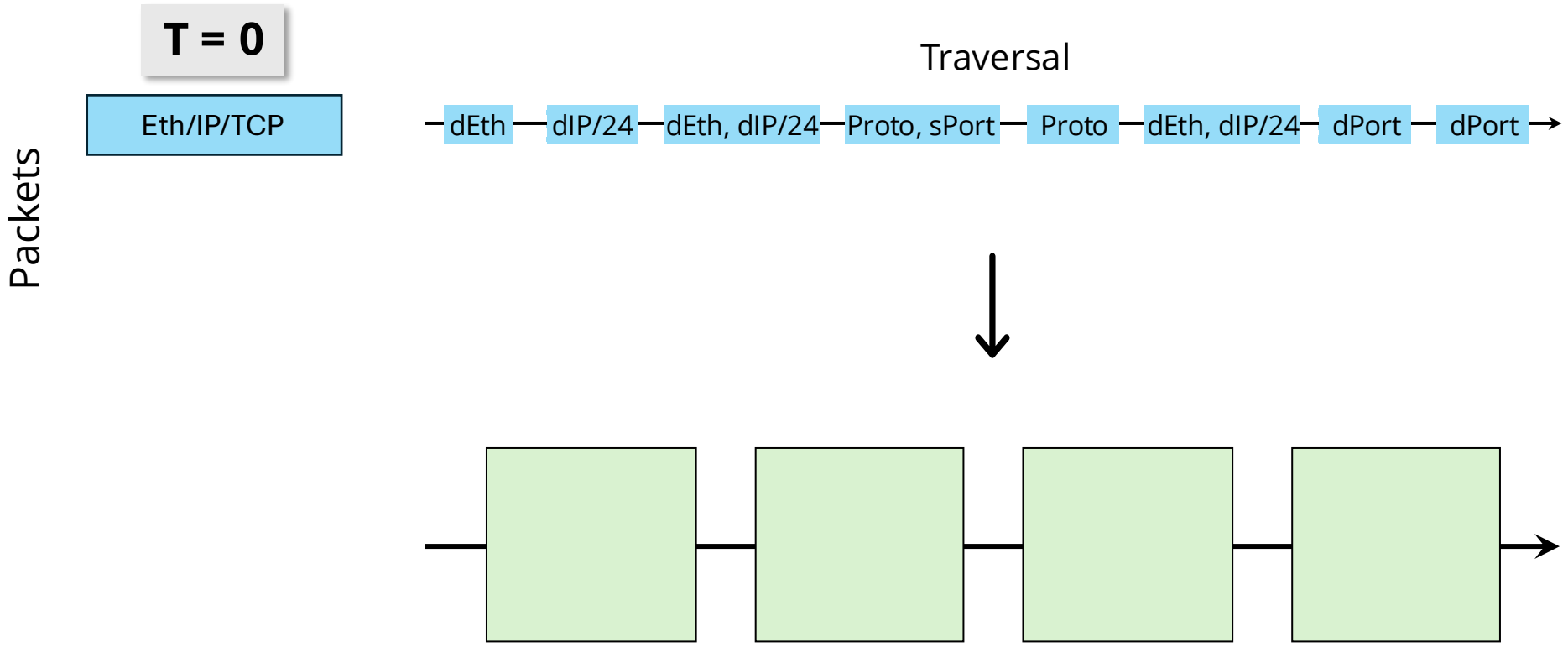
How about *Partitioning* the Traversal?



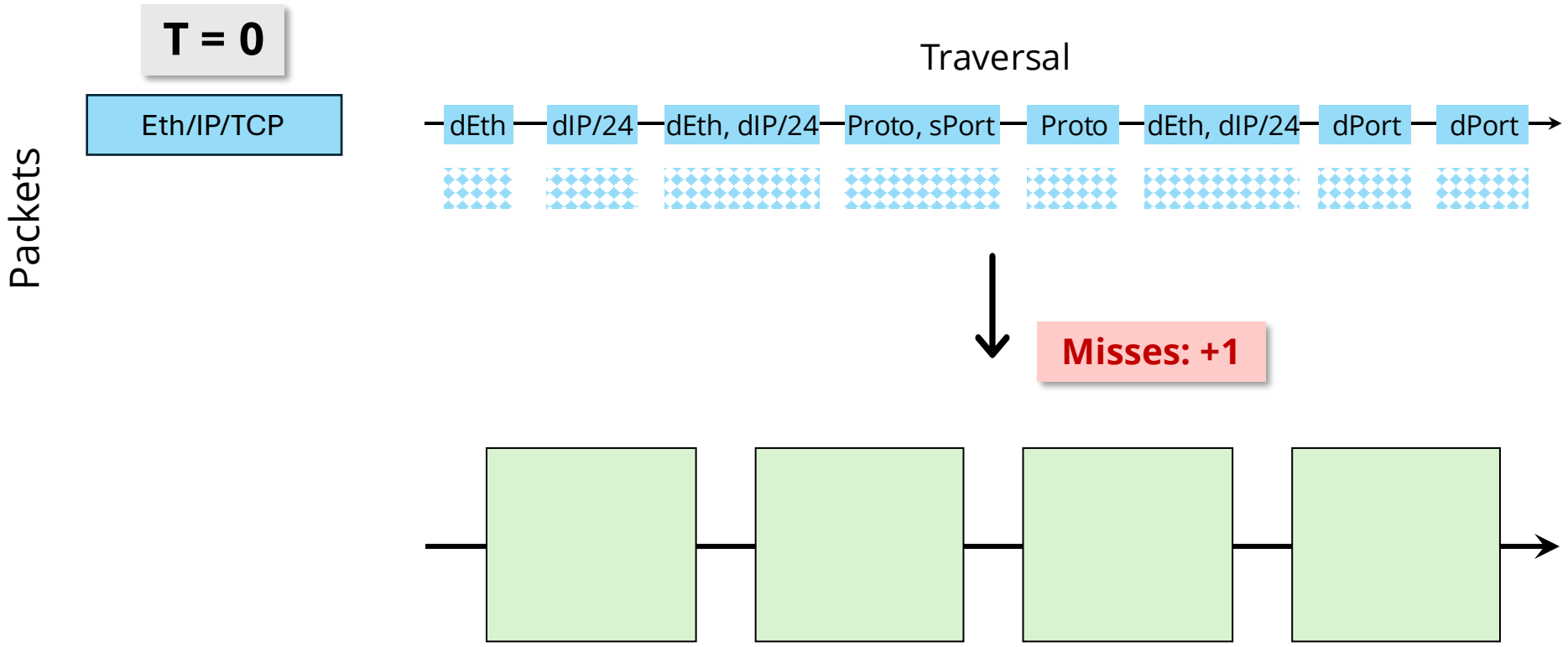
How about *Partitioning* the Traversal?



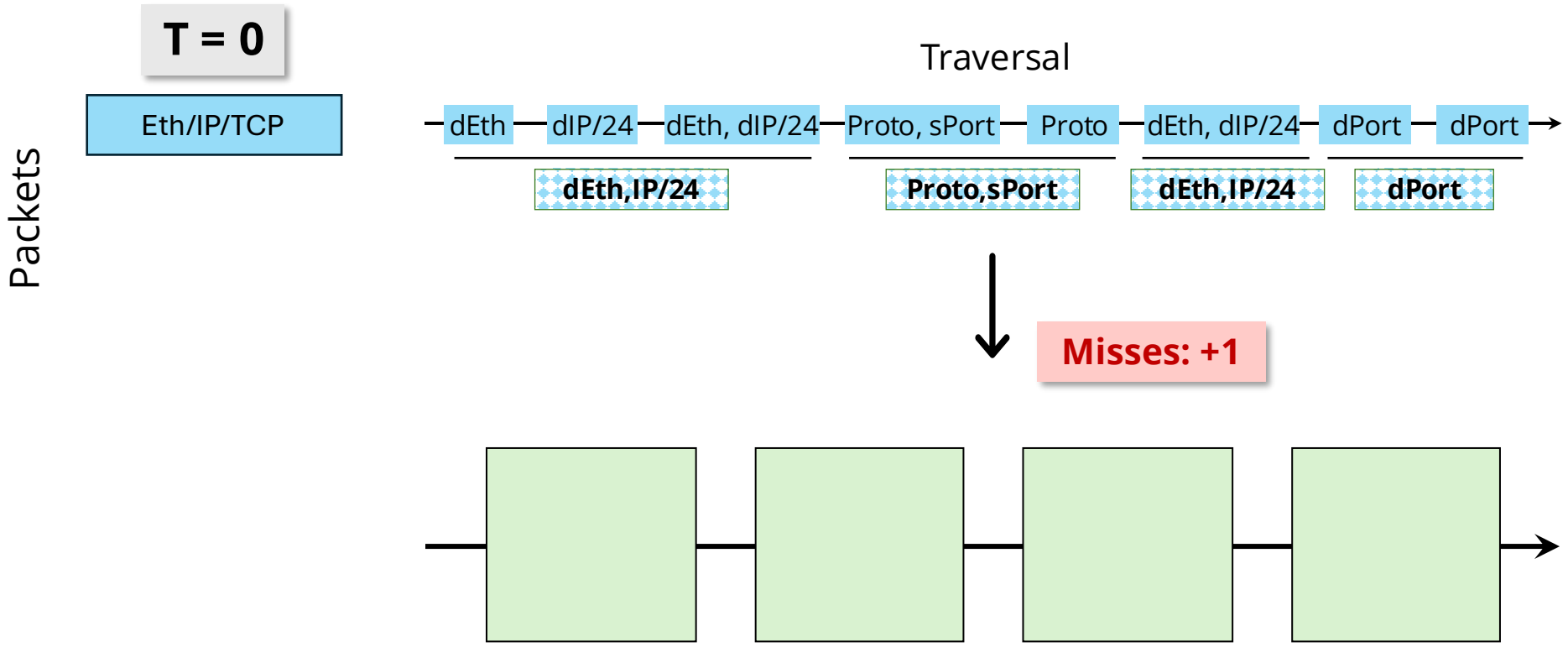
How about Smartly *Partitioning* the Traversal?



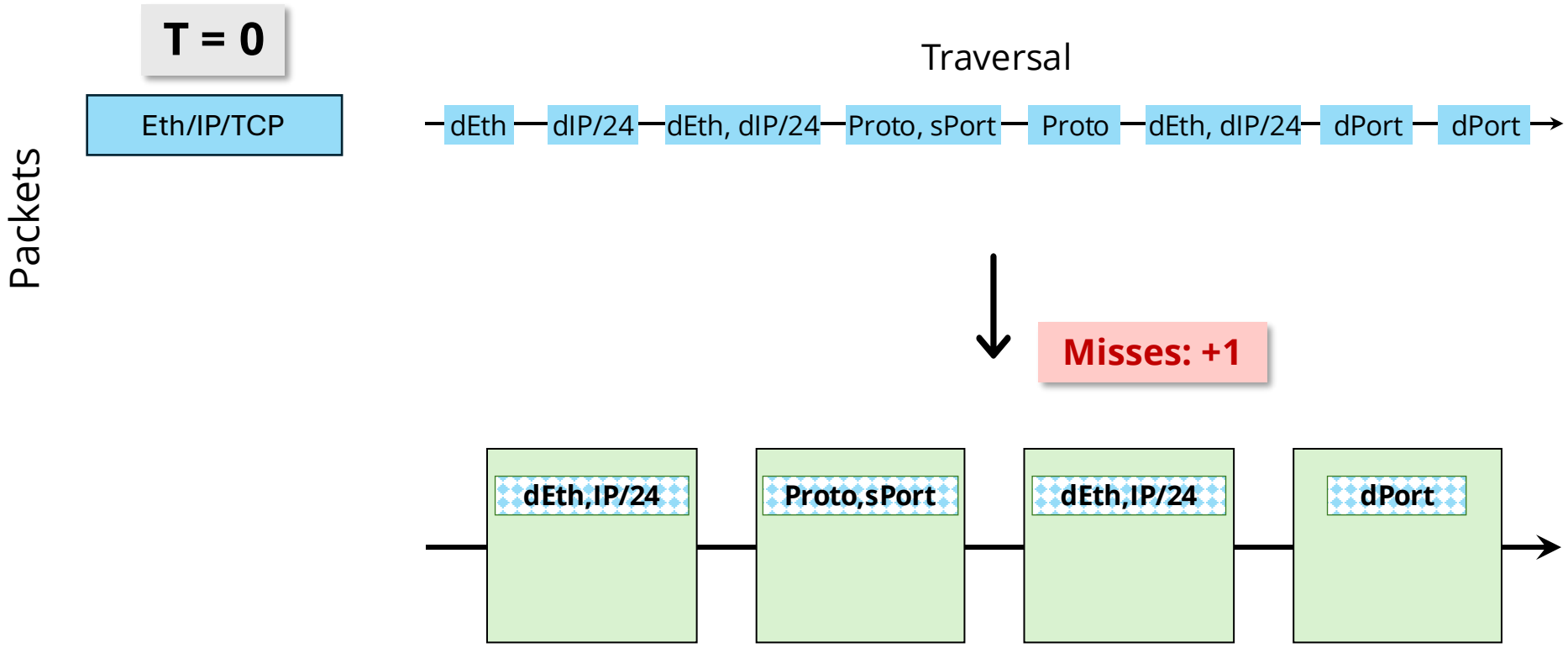
How about Smartly *Partitioning* the Traversal?



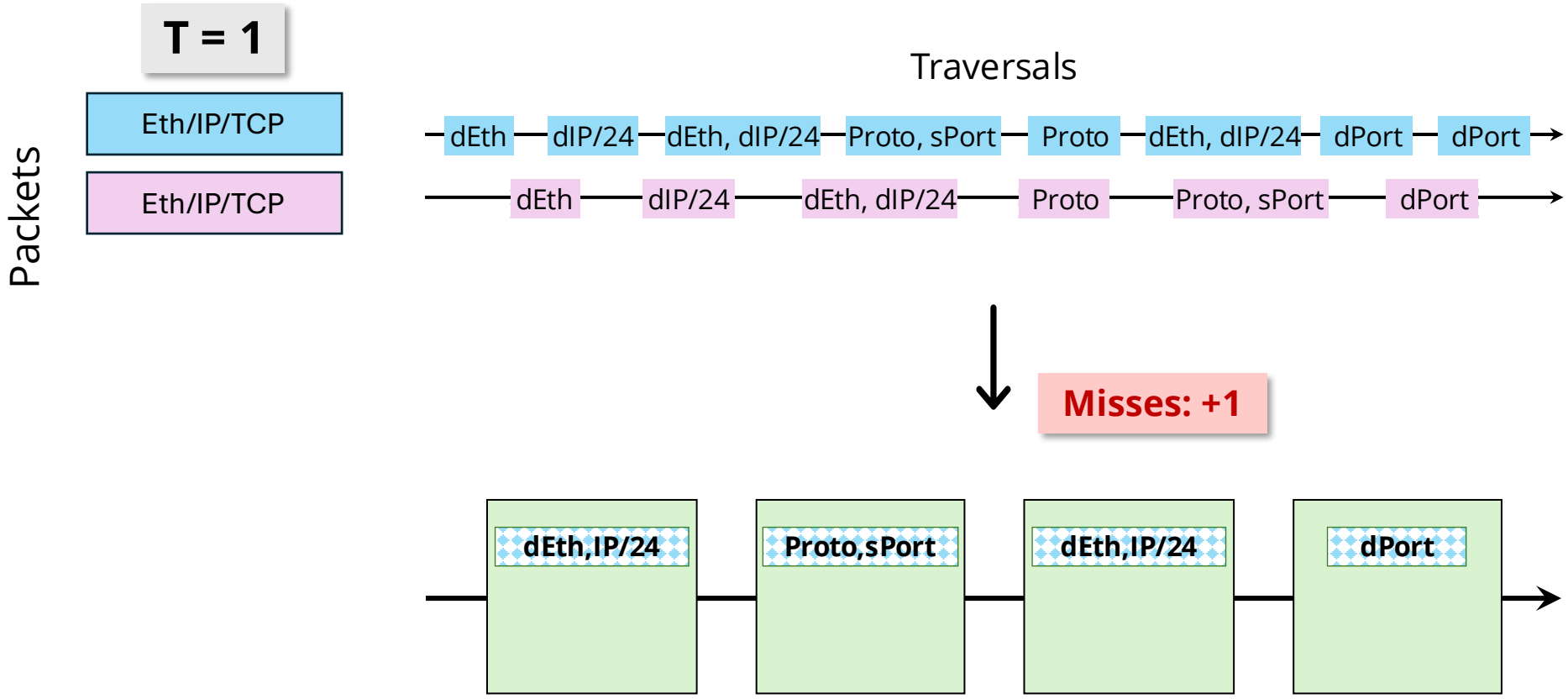
How about Smartly *Partitioning* the Traversal?



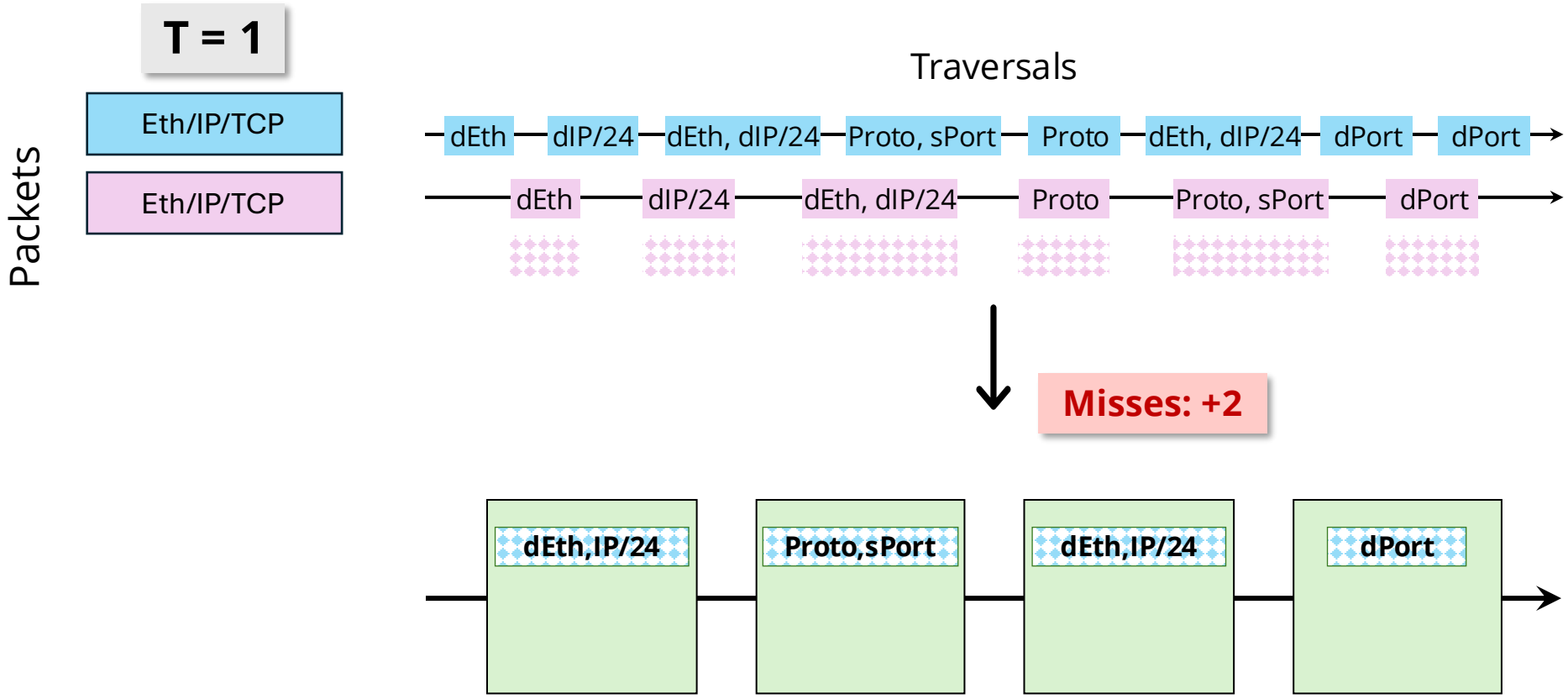
How about Smartly *Partitioning* the Traversal?



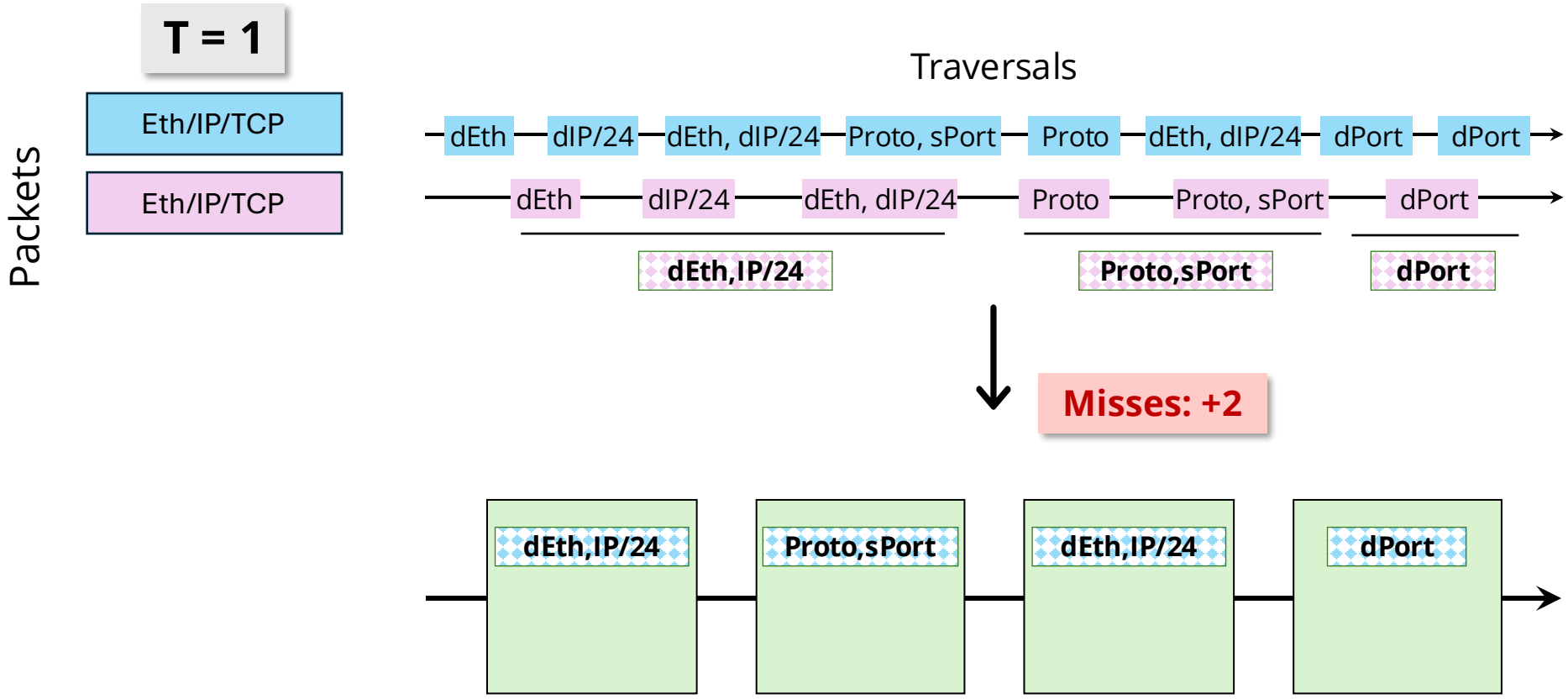
How about Smartly *Partitioning* the Traversal?



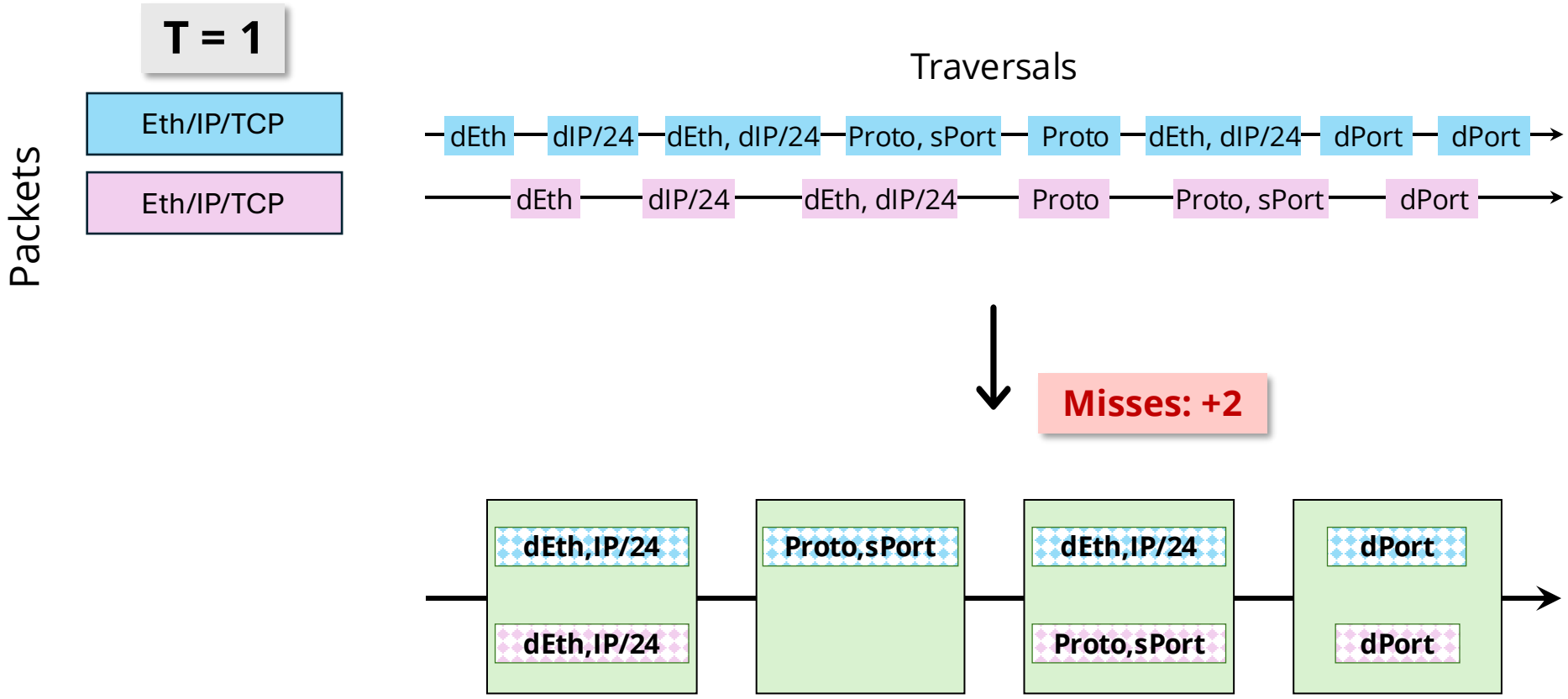
How about Smartly *Partitioning* the Traversal?



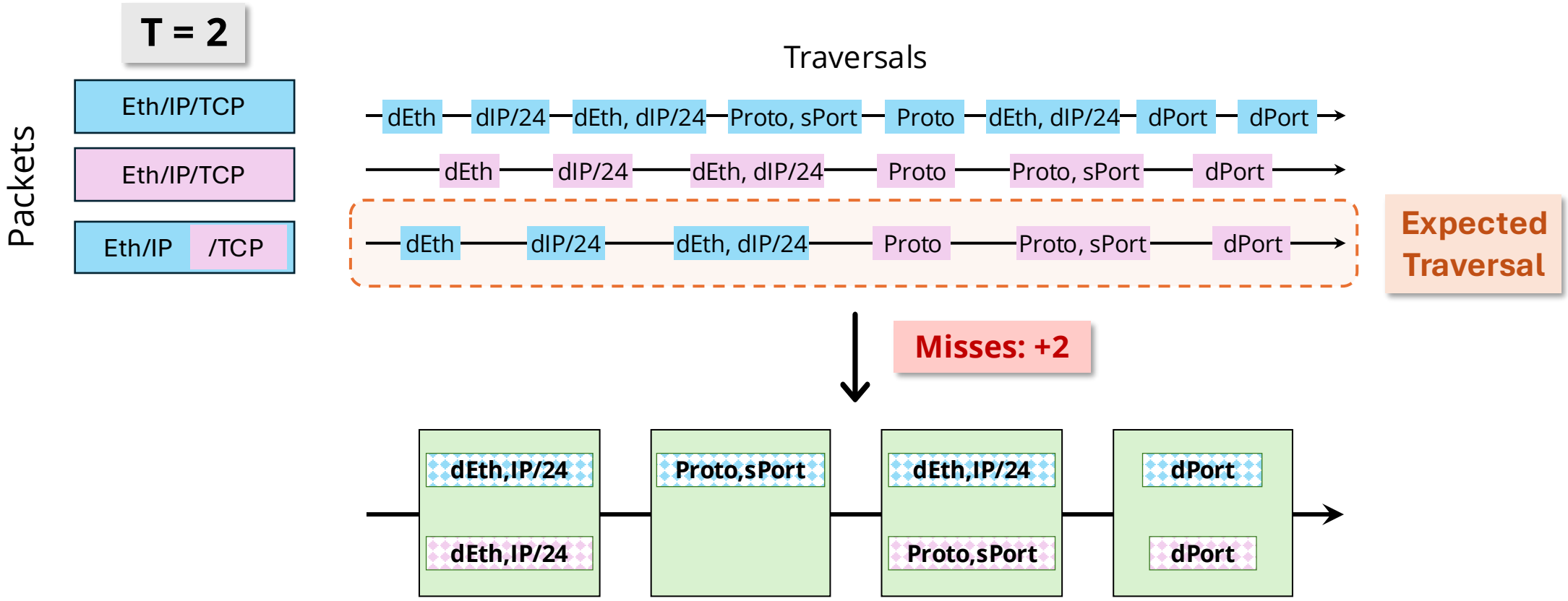
How about Smartly *Partitioning* the Traversal?



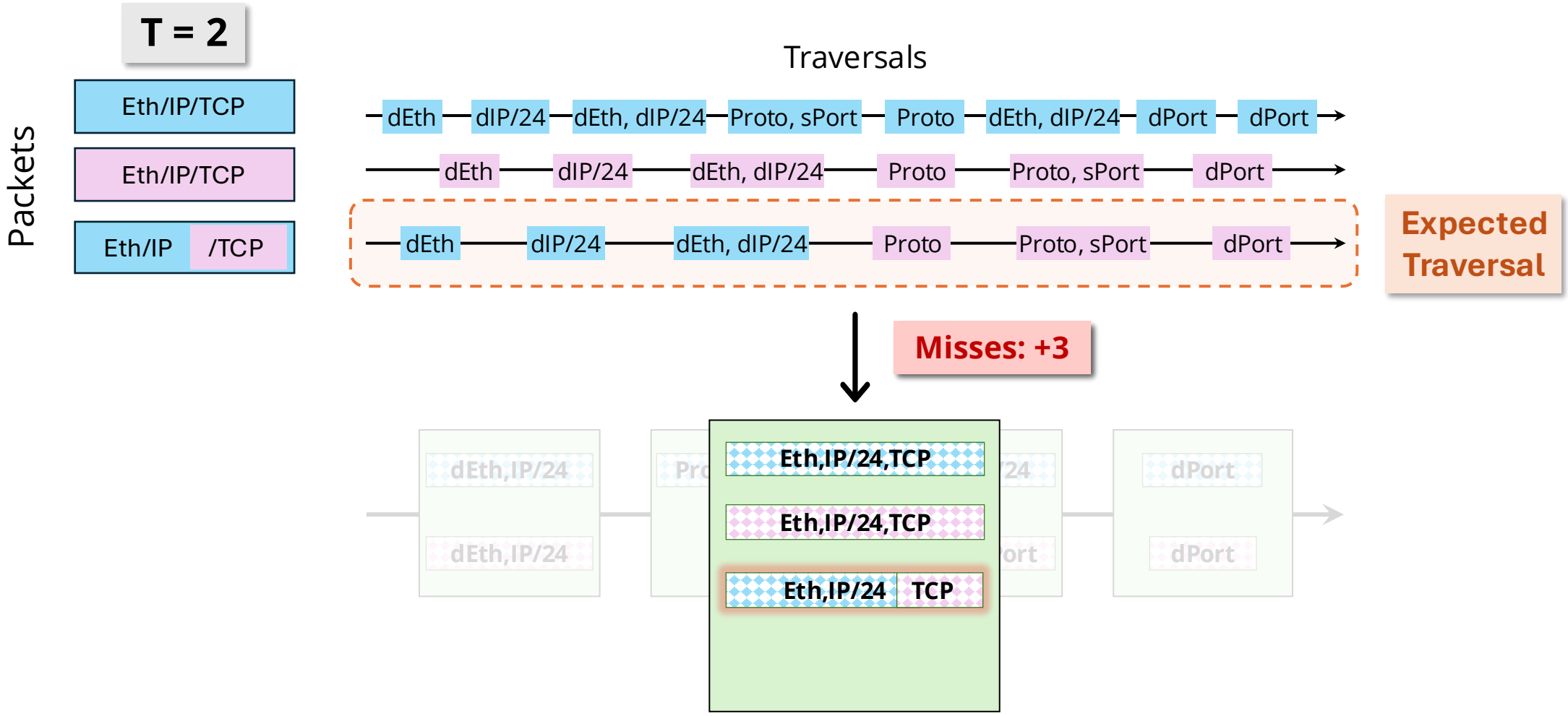
How about Smartly *Partitioning* the Traversal?



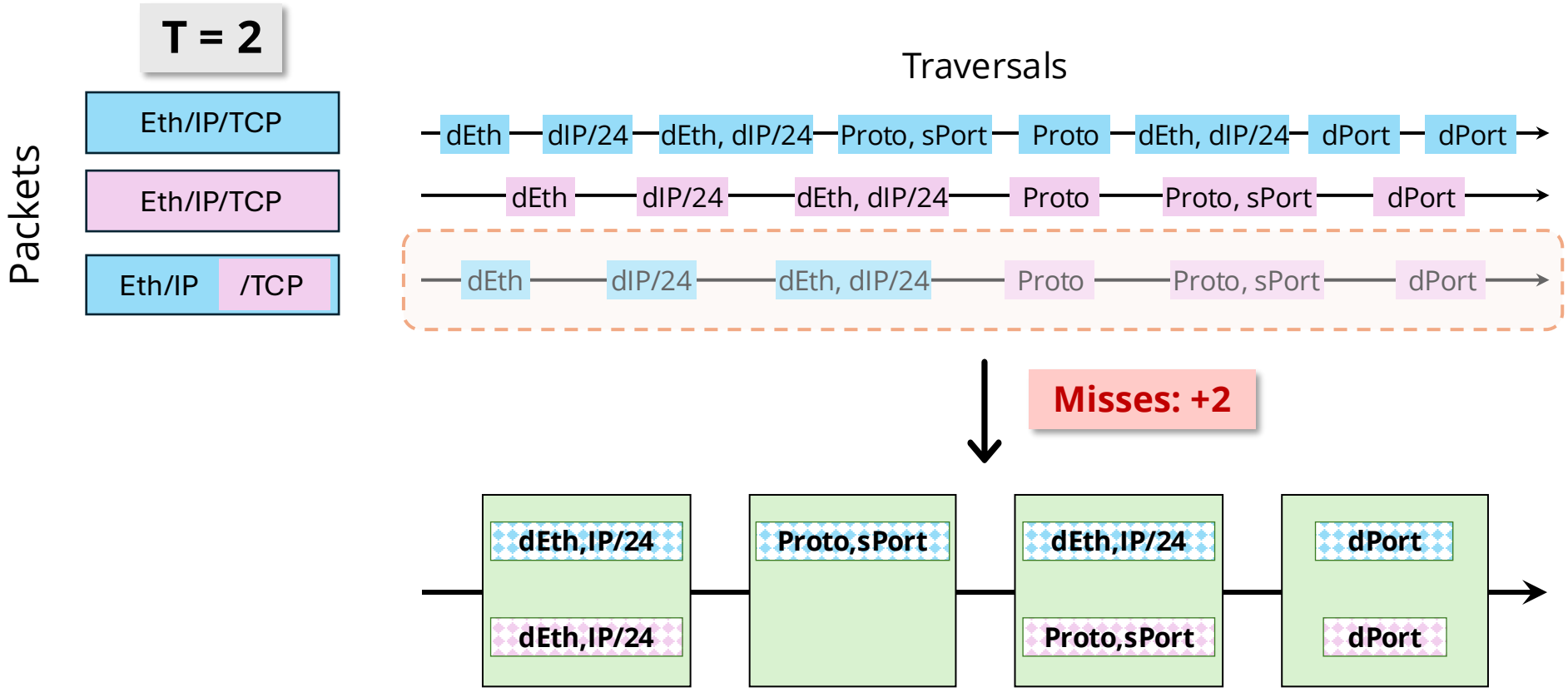
How about Smartly *Partitioning* the Traversal?



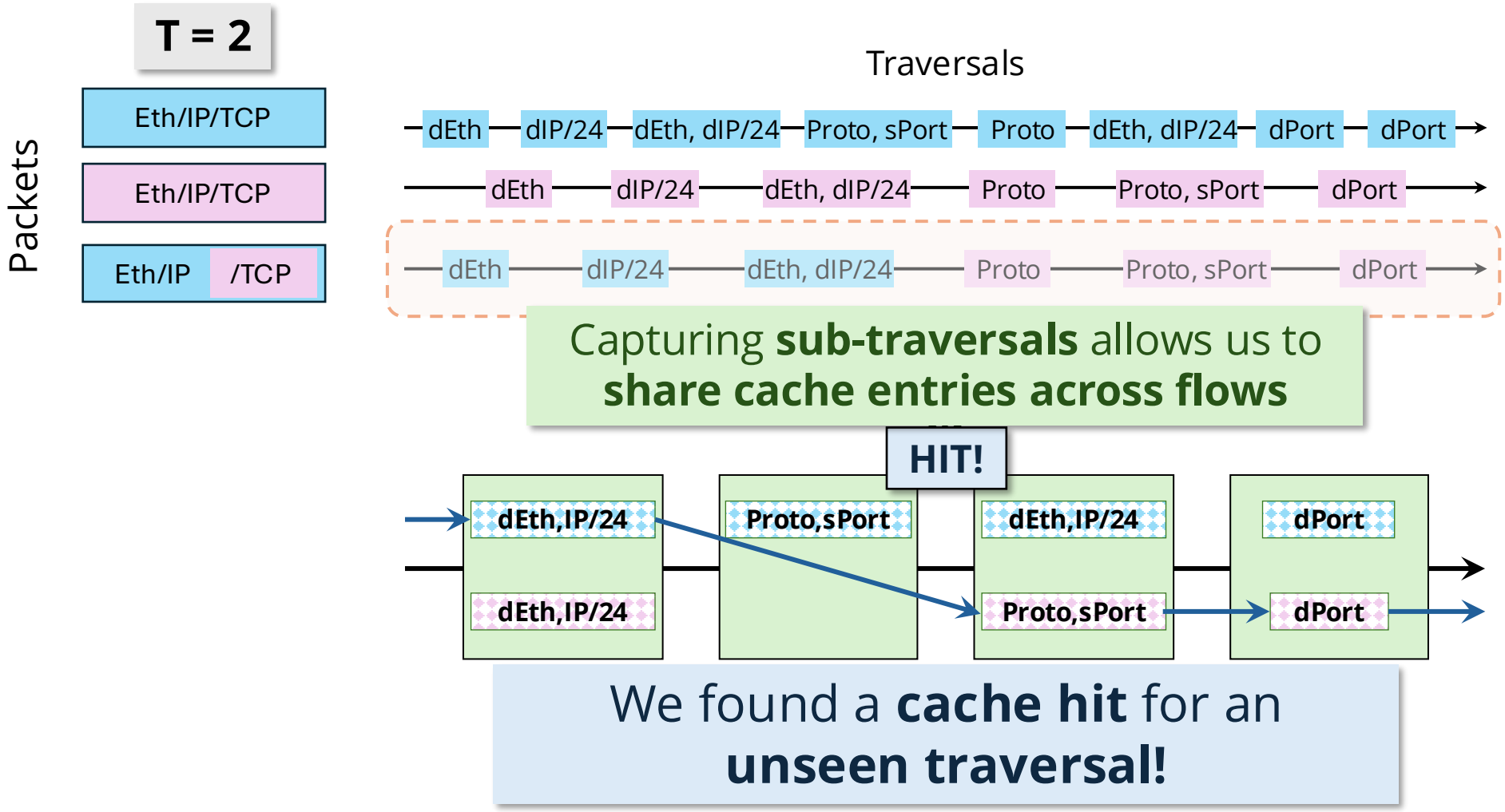
How about Smartly *Partitioning* the Traversal?



How about Smartly *Partitioning* the Traversal?



How about Smartly *Partitioning* the Traversal?

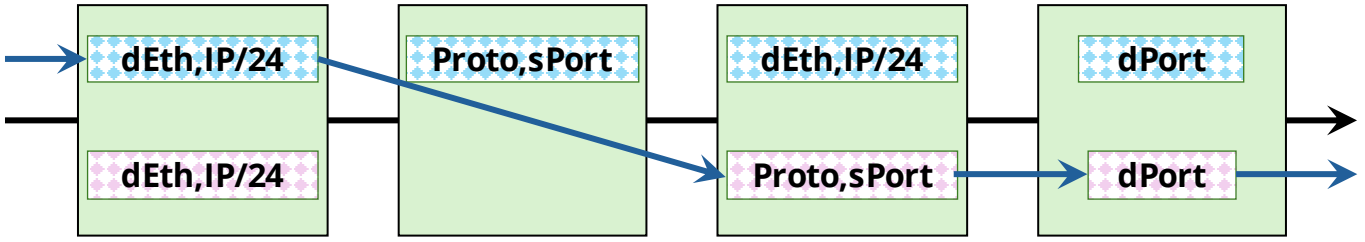


Sub-Traversal Caching with Gigaflow

Sub-traversals shared across flows
as cache entries in the SmartNIC

Optimal sub-traversals as cache entries
capture **pipeline-aware locality**

Up to **two orders of magnitude improvement** in **rule space coverage!**



Sub-Traversal Caching with Gigaflow

Optimal sub-traversals as cache entries capture **pipeline-aware locality**

Up to **two orders of magnitude improvement** in **rule space coverage!**

SmartNIC Cache	Real-World vSwitch Pipelines (Ps)				
	U1	**U2**	**U3**	**U4**	**U5**
MF	32K	32K	32K	32K	32K
GF	??				

Rule Space Coverage of Megaflow 32K (MF)
vs Gigaflow 4x8K (GF) Cache

Capturing Pipeline-Aware Locality

Algorithm 1 Strawman Approach

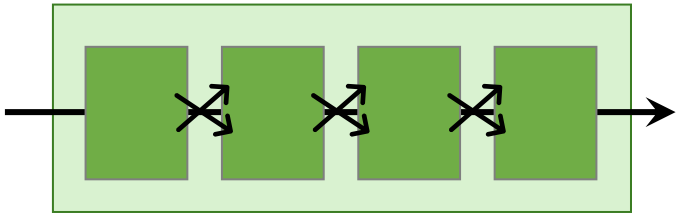
Input: Set of traversals $T = \{t_1, t_2, \ldots, t_N\}$
Output: Sub-traversals with maximum rule space coverage

This cost is **proportional** to **active flows!**
Keeps **increasing** with each **cache miss!**
Infeasible at line rate!

How to capture **pipeline-aware** locality ***efficiently*?**

Capturing Pipeline-Aware Locality *Efficiently*

To goal is to **maximize** the **rule space coverage** in the SmartNIC tables



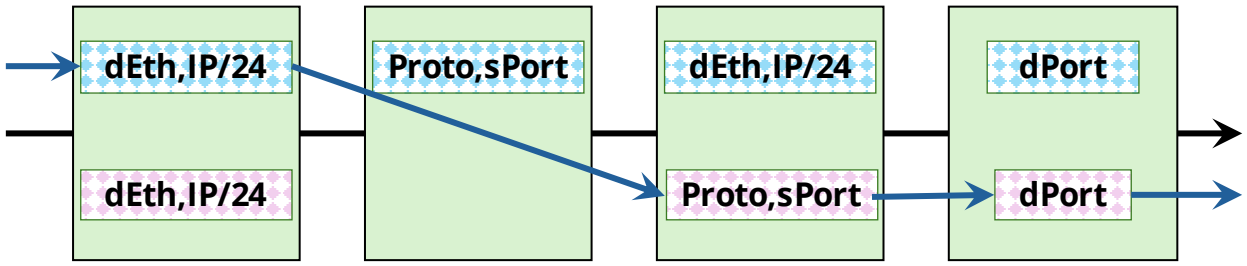
This **cross-product** rule space **should be maximized!**

Pipeline-Aware locality → **Disjointedness**

Maximize *the* ***field-level disjointedness*** *among consecutive sub-traversals*

Capturing Pipeline-Aware Locality *Efficiently*

To goal is to **maximize** the **rule space coverage** in the SmartNIC tables



This **cross-product** rule space **should be maximized!**

Pipeline-Aware locality → **Disjointedness**

Maximize *the* ***field-level disjointedness*** *among consecutive sub-traversals*

Rules Spaces With Different Localities

Microflow Cache

Megaflow Cache

Gigaflow Cache

cache entry

traversals

flows

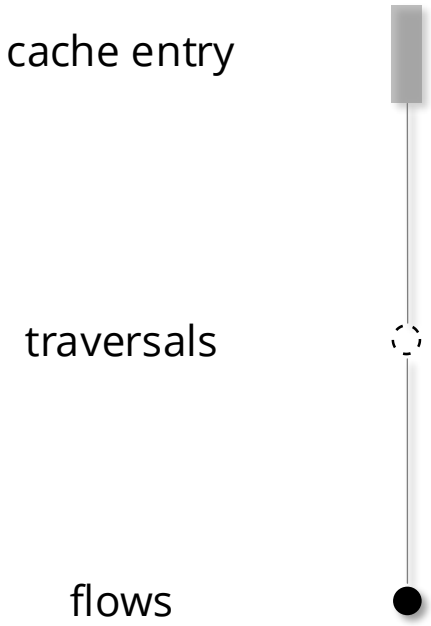
Rules Spaces With Different Localities

Temporal Locality

Microflow Cache

Megaflow Cache

Gigaflow Cache



One flow** per **cache miss

Rules Spaces With Different Localities

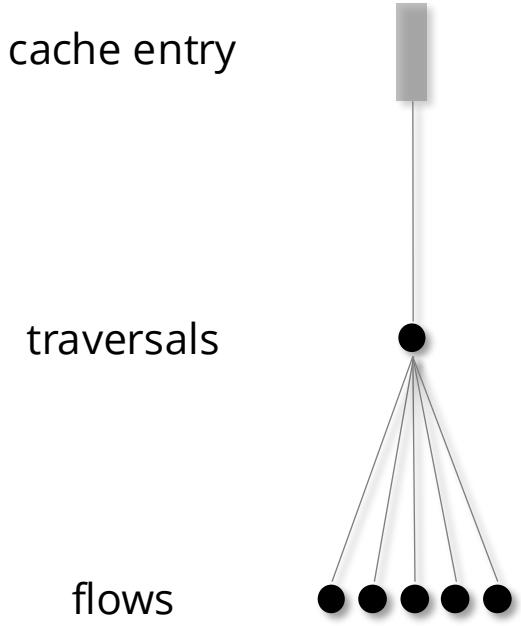
Temporal Locality

Microflow Cache

One flow** per **cache miss

Temporal + Spatial Locality

Megaflow Cache



One traversal** per **cache miss
= Set of flows

Gigaflow Cache

Rules Spaces With Different Localities

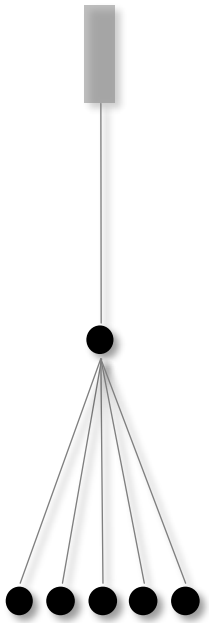
Temporal Locality

Microflow Cache

One flow** per **cache miss

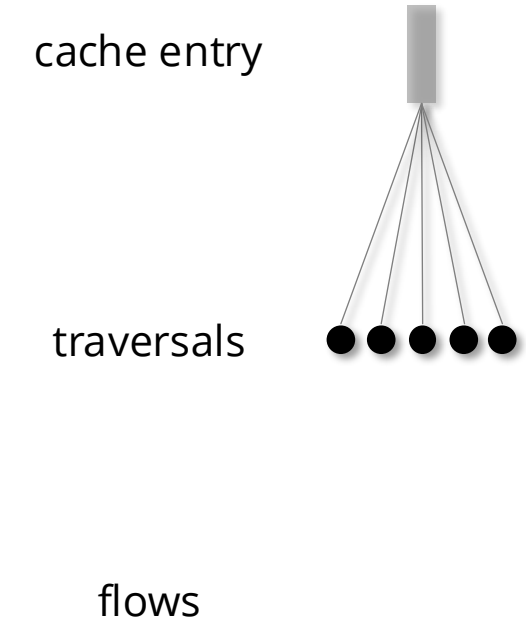
Temporal + Spatial Locality

Megaflow Cache



One traversal** per **cache miss
= Set of flows

Gigaflow Cache



Rules Spaces With Different Localities

Temporal Locality

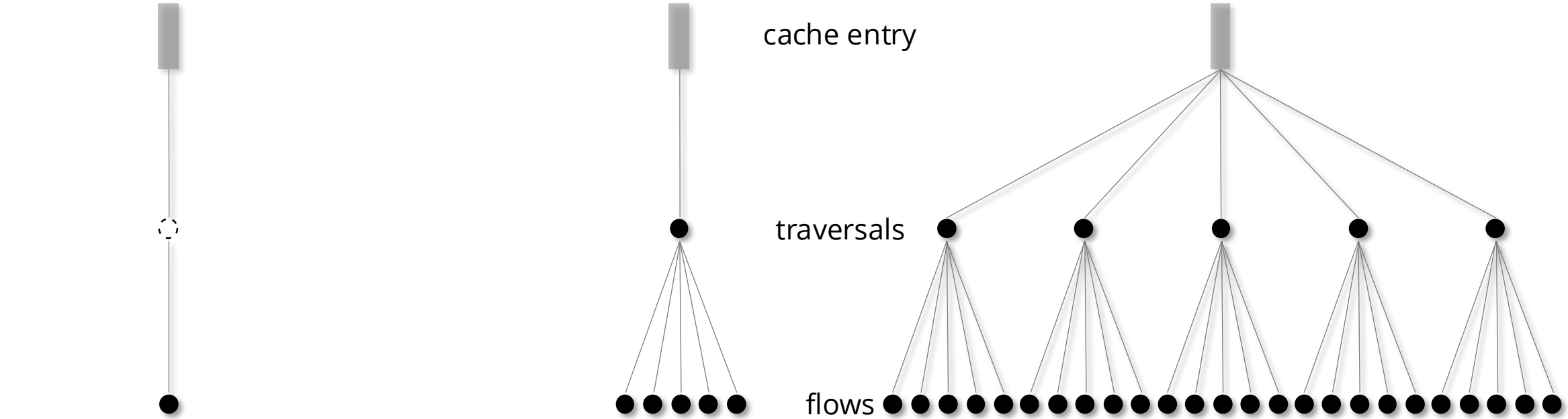
Microflow Cache

Temporal + Spatial Locality

Megaflow Cache

Temporal + Spatial + Pipeline-Aware Locality

Gigaflow Cache

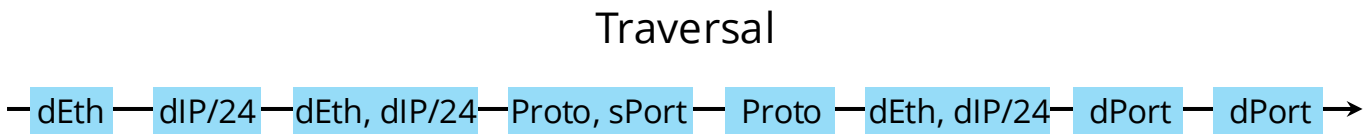


One flow** per **cache miss

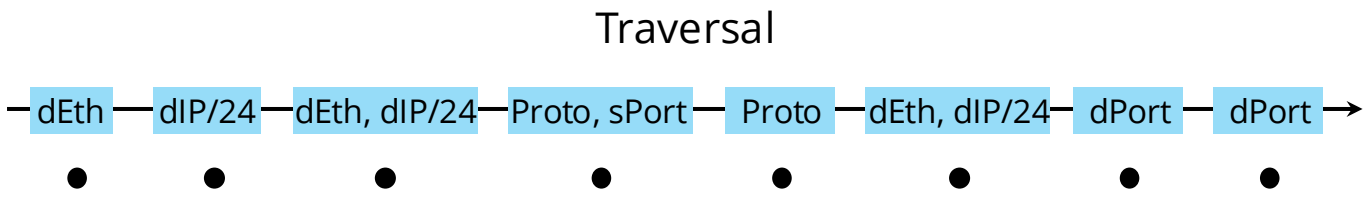
One traversal** per **cache miss
= Set of flows

Set of traversals** per **cache miss
= Set of set of flows

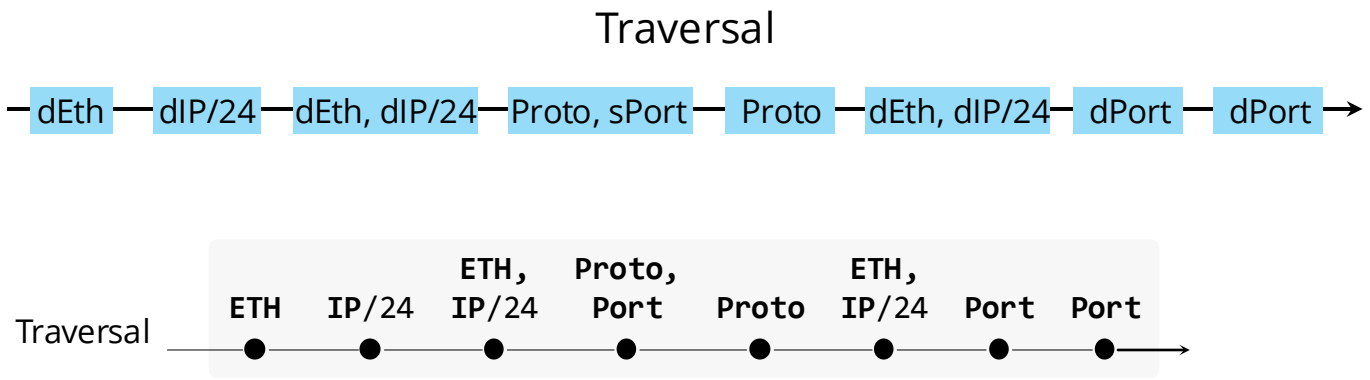
Disjointedness for Pipeline-Aware Locality



Disjointedness for Pipeline-Aware Locality

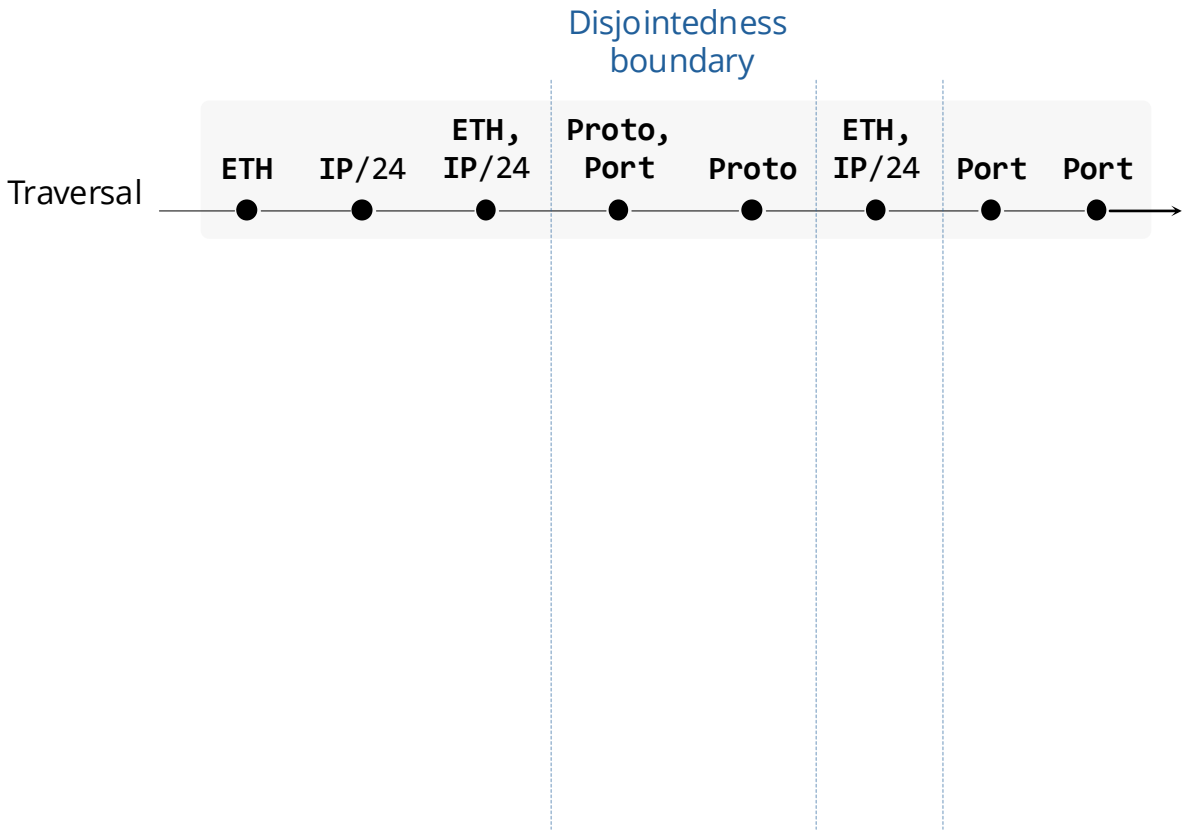


Disjointedness for Pipeline-Aware Locality

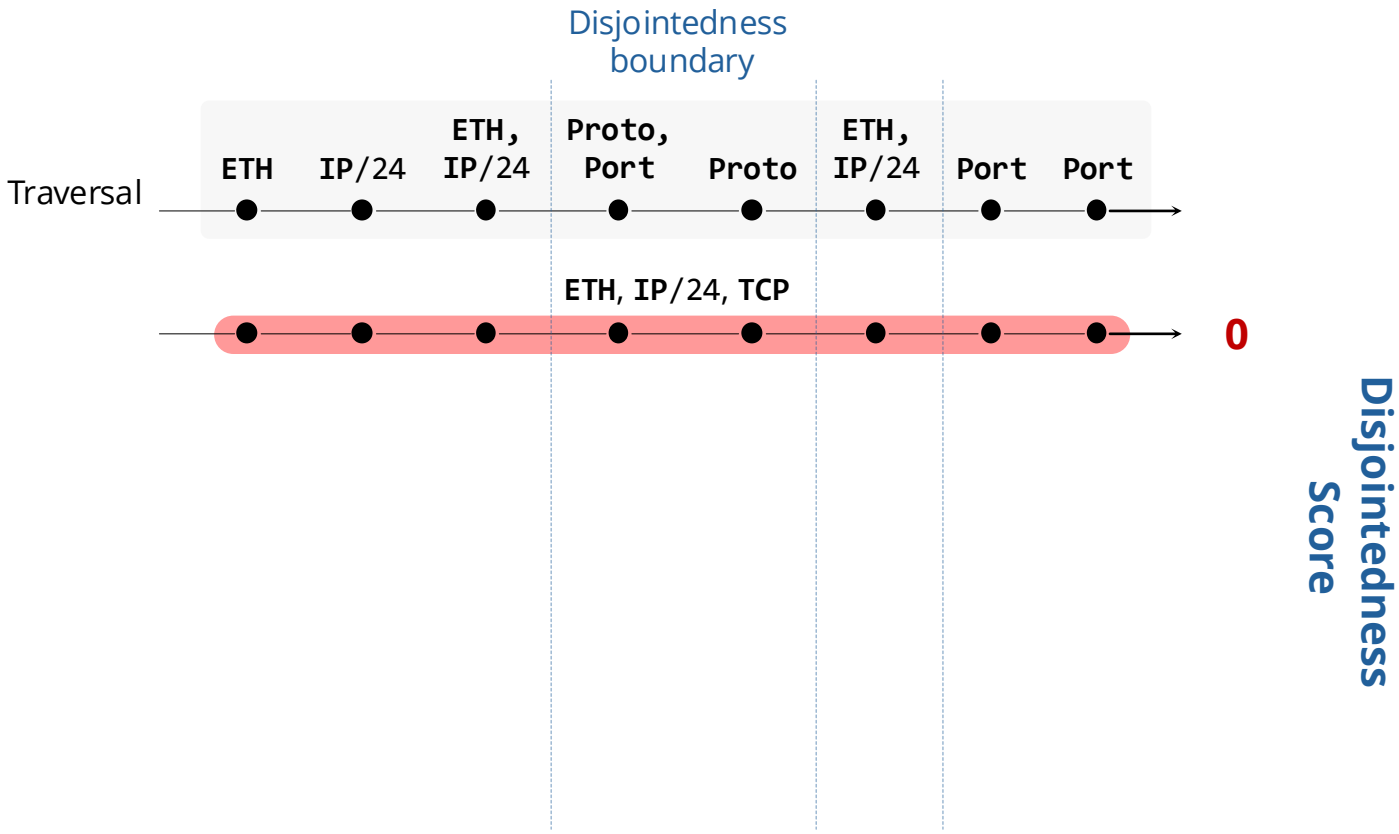


Disjointedness for Pipeline-Aware Locality

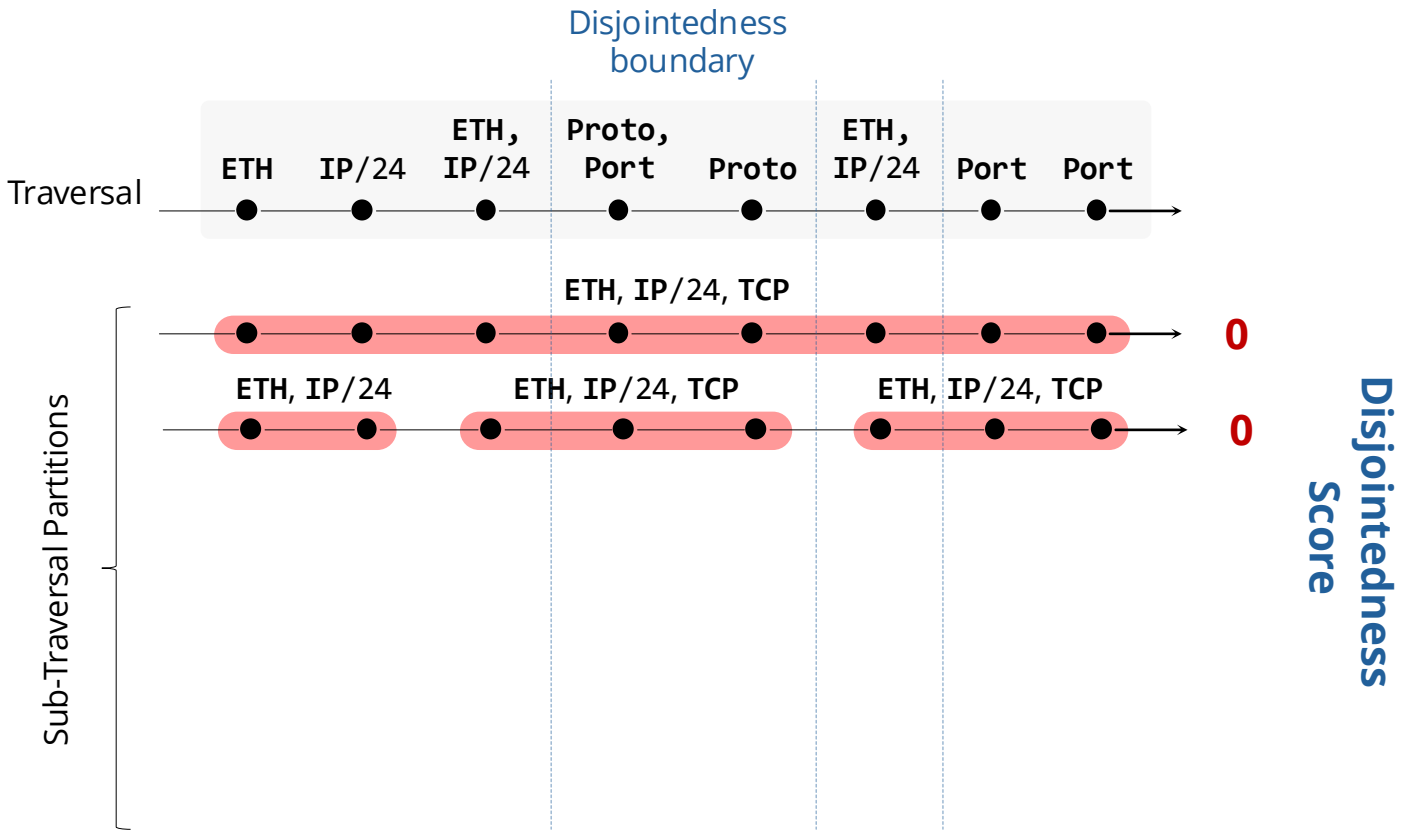
Disjointedness for Pipeline-Aware Locality



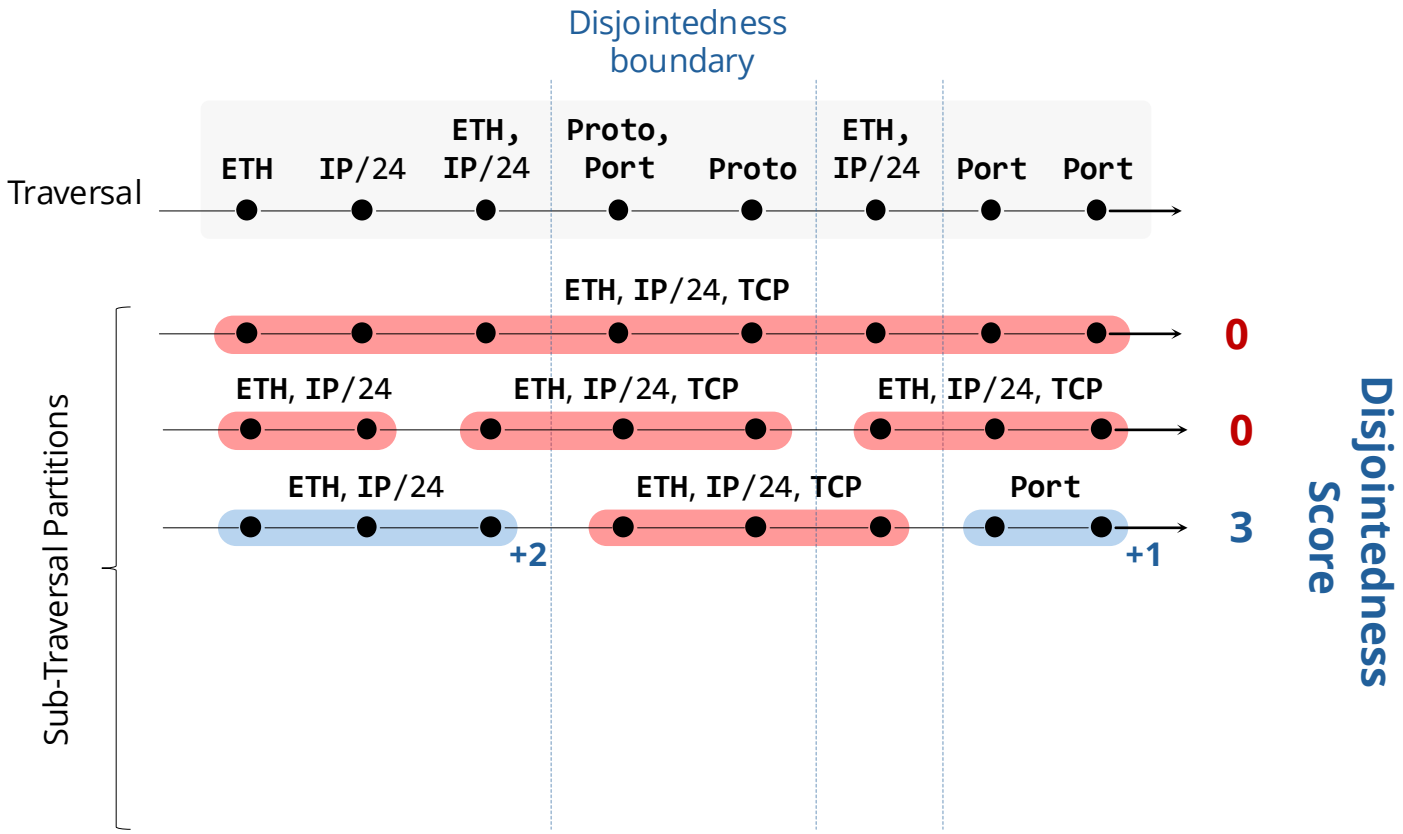
Disjointedness for Pipeline-Aware Locality



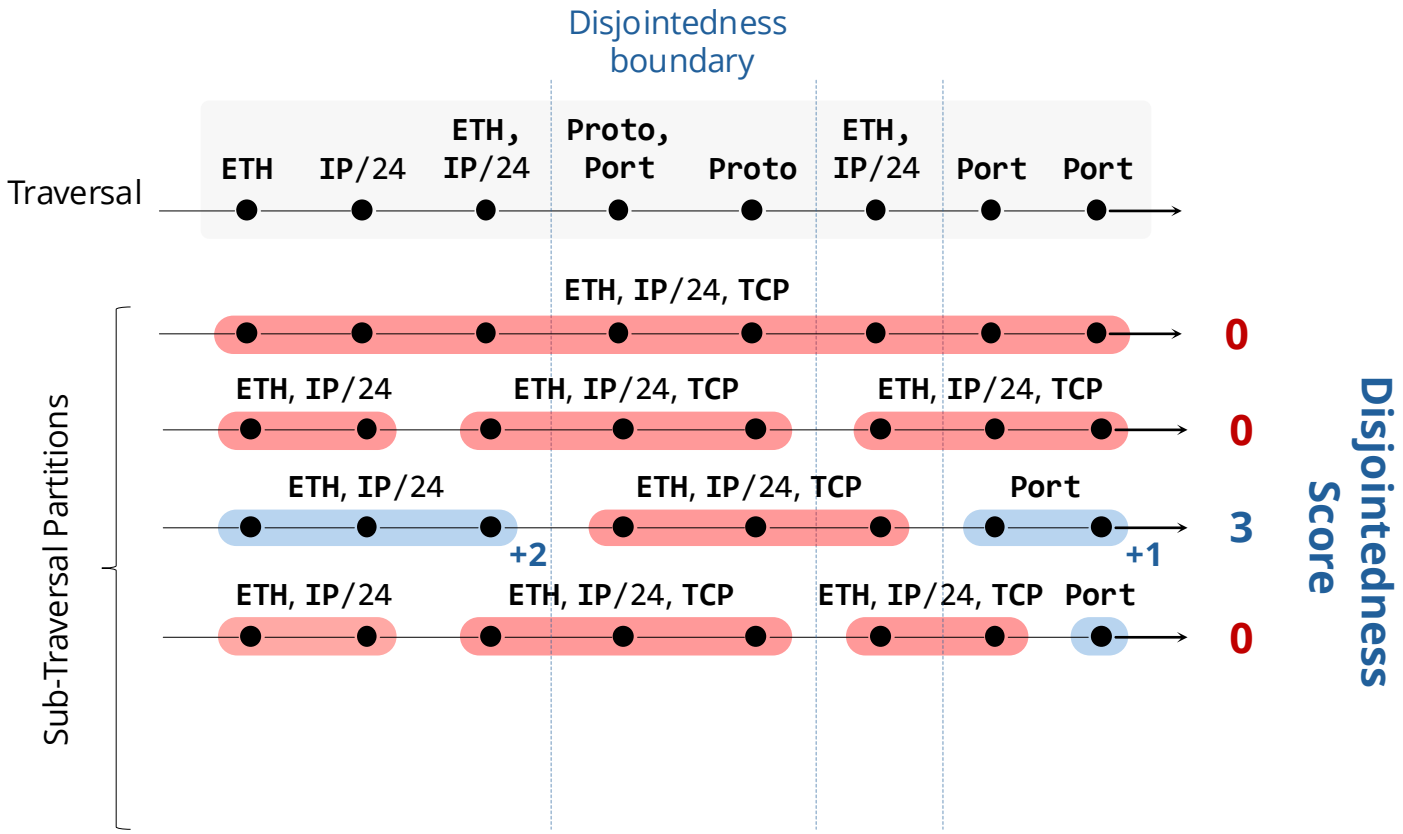
Disjointedness for Pipeline-Aware Locality



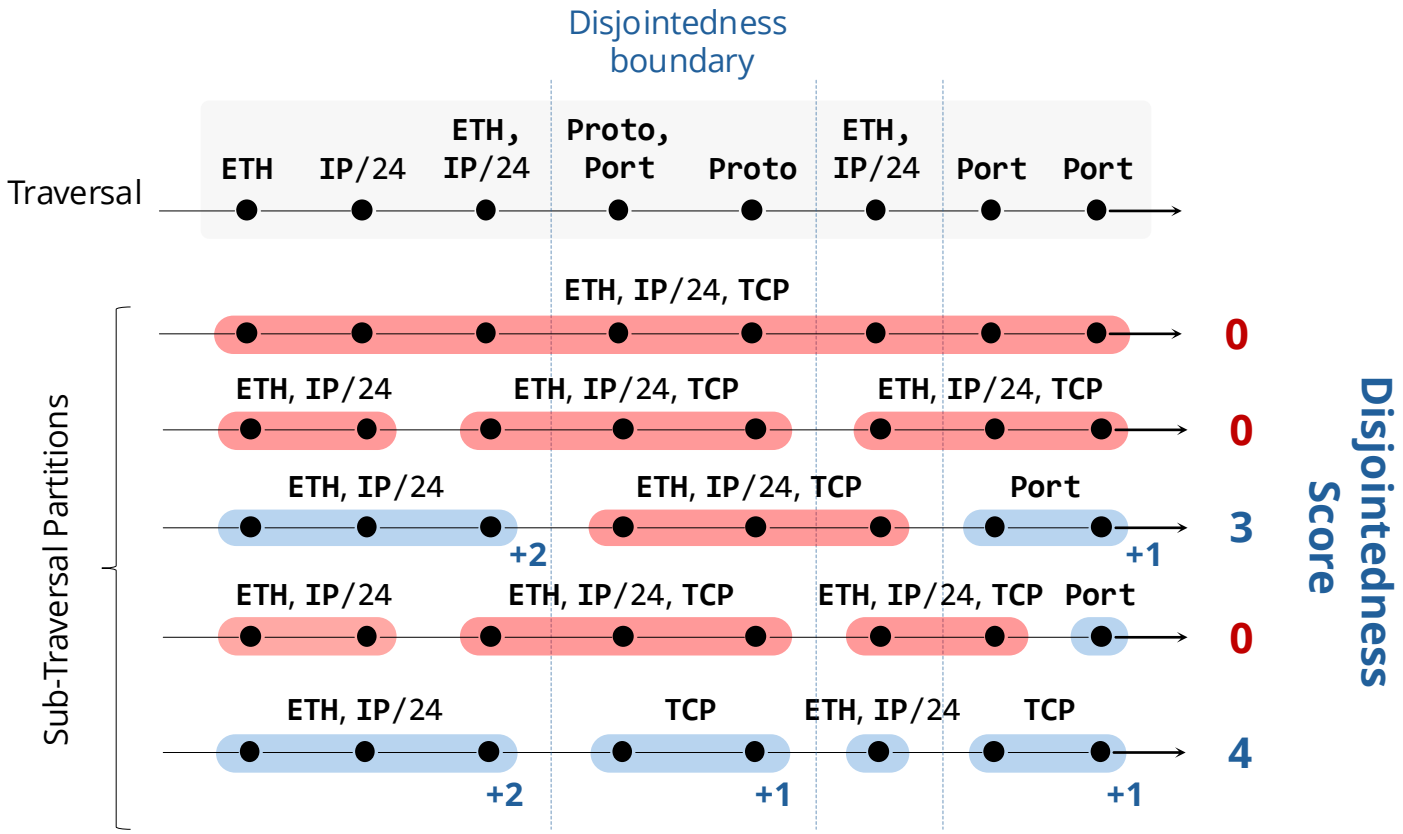
Disjointedness for Pipeline-Aware Locality



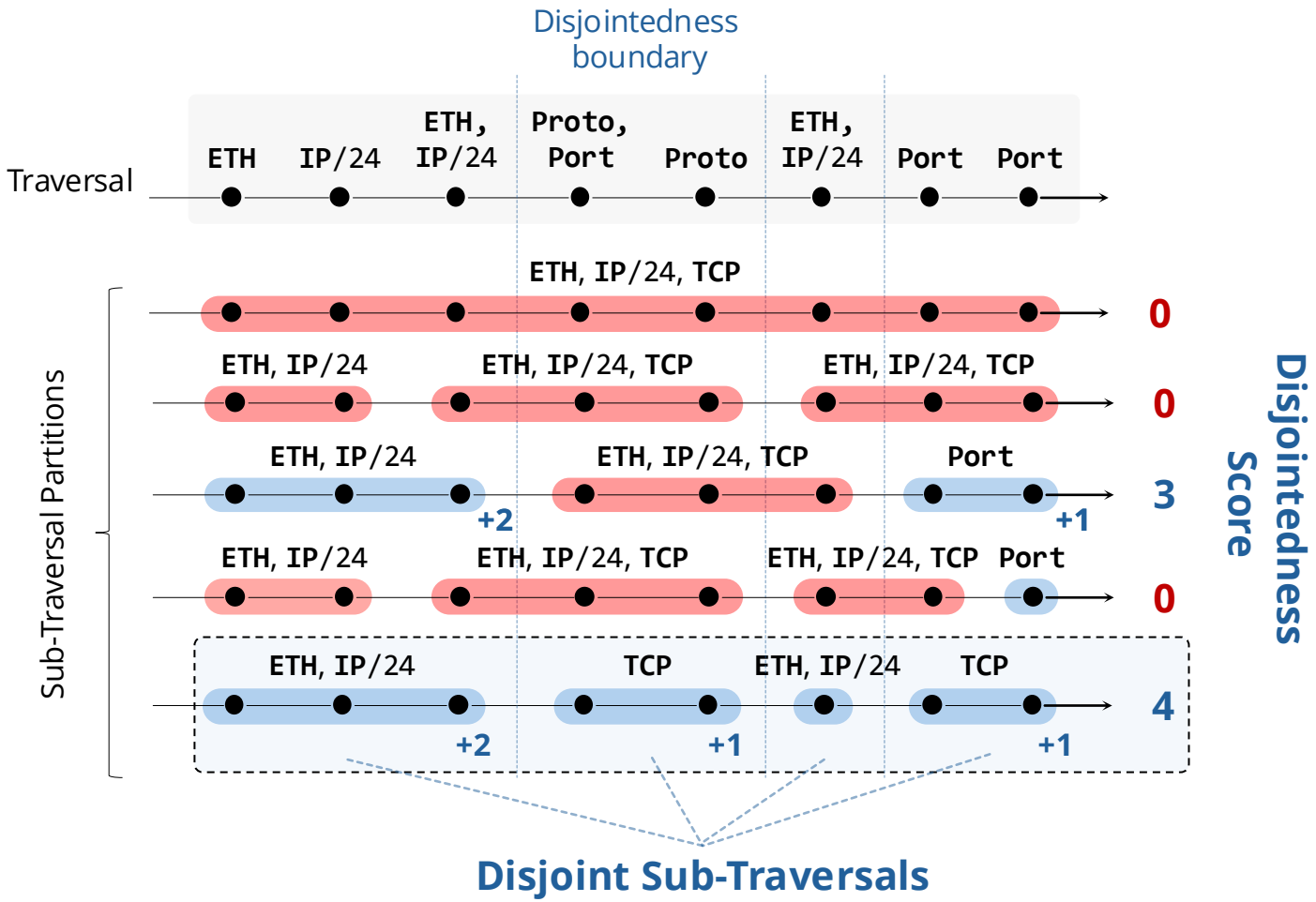
Disjointedness for Pipeline-Aware Locality



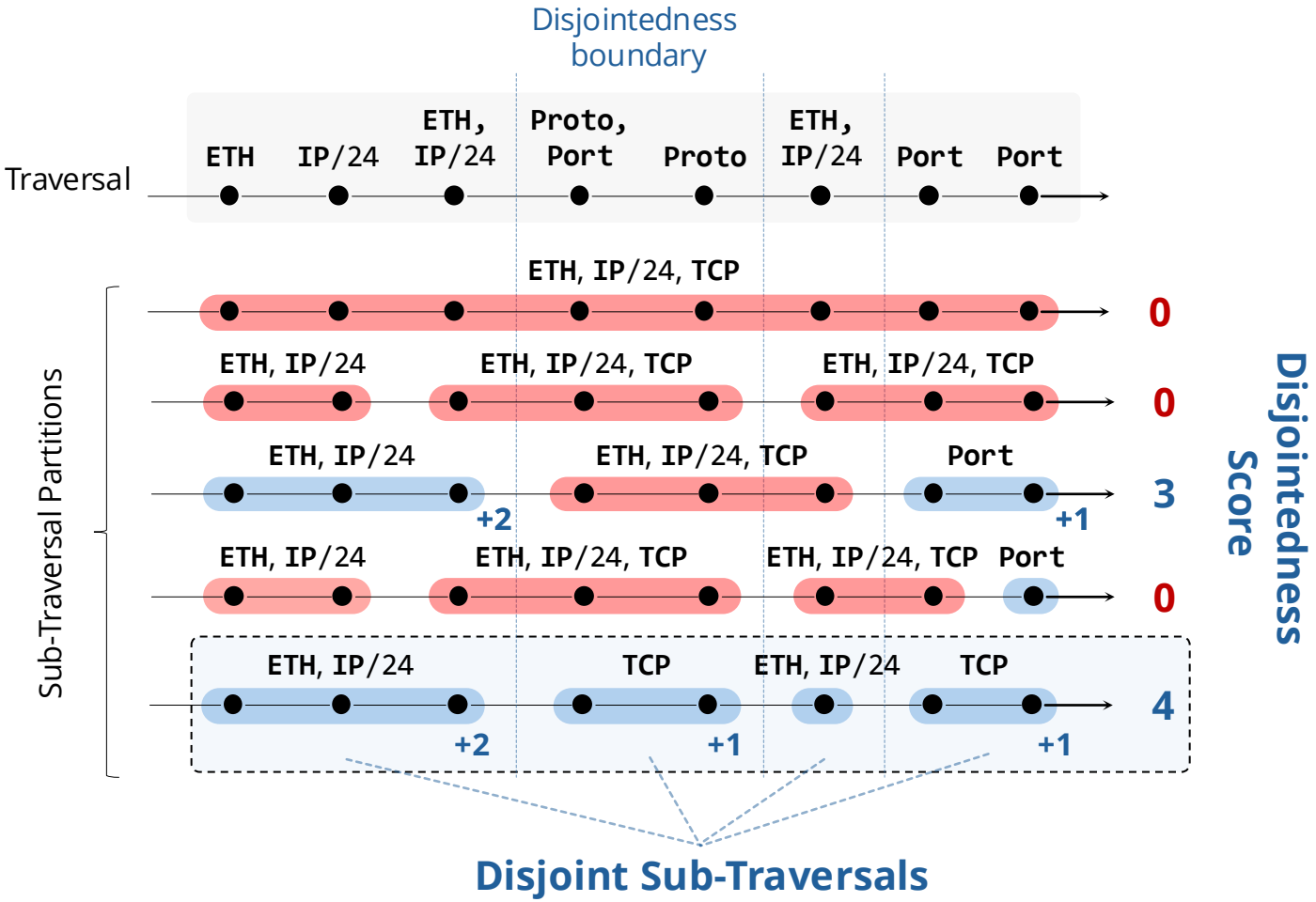
Disjointedness for Pipeline-Aware Locality



Disjointedness for Pipeline-Aware Locality



Disjointedness for Pipeline-Aware Locality



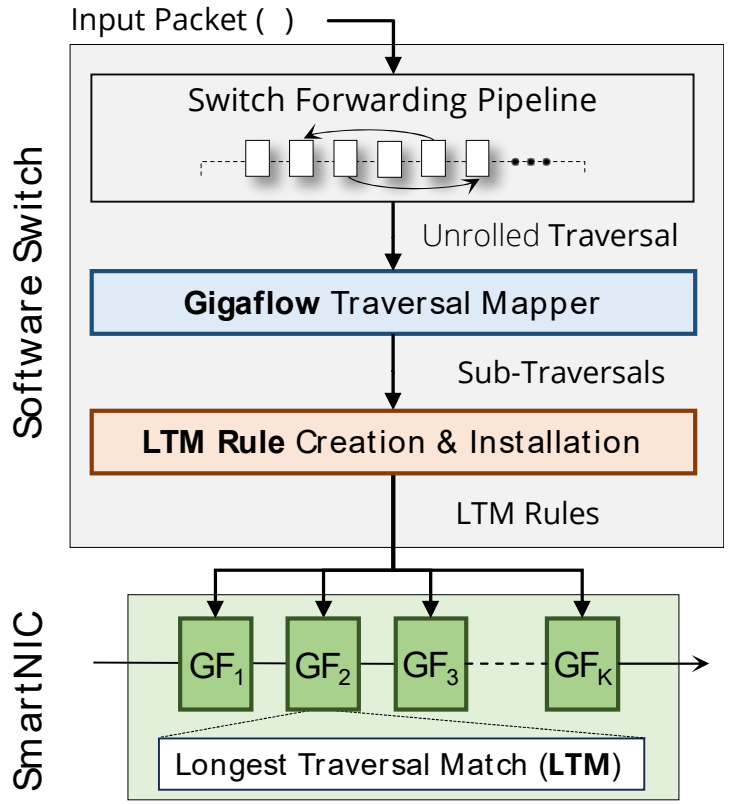
Worst Case Overhead	**5x**
Amortized Overhead	**2x**

Software processing overhead of Disjoint Partitioning in Gigaflow

Overall cost is **negligible** because **90%** of **cache misses** are **reduced**

Gigaflow Cache Implementation

- Integrated in the **Open vSwitch**
- Cache offload for **Xilinx Alveo U250**
- **Xilinx P4 SDNet** for hardware tables



High-level view of the Gigaflow in OVS

vSwitch Pipelines and Traffic Workloads

Pipeline	Description	Tables	Traversals
U1: OFD	CORD's Openflow data plane abstraction (OFDPA) for HW/SW switch integration	10	5
U2: PSC	An example L2L3-ACL pipeline implemented for the Pisces paper	7	2
U3: OLS	OVN Logical Switch pipeline to manage logical virtual network topologies in OVS	30	23
U4: ANT	Antrea pipeline implementing networking and security for Kubernetes clusters	22	20
U5: OTL	Openflow Table Type Patterns (TTP) to configure L2L3-ACL policies using OVS	8	11

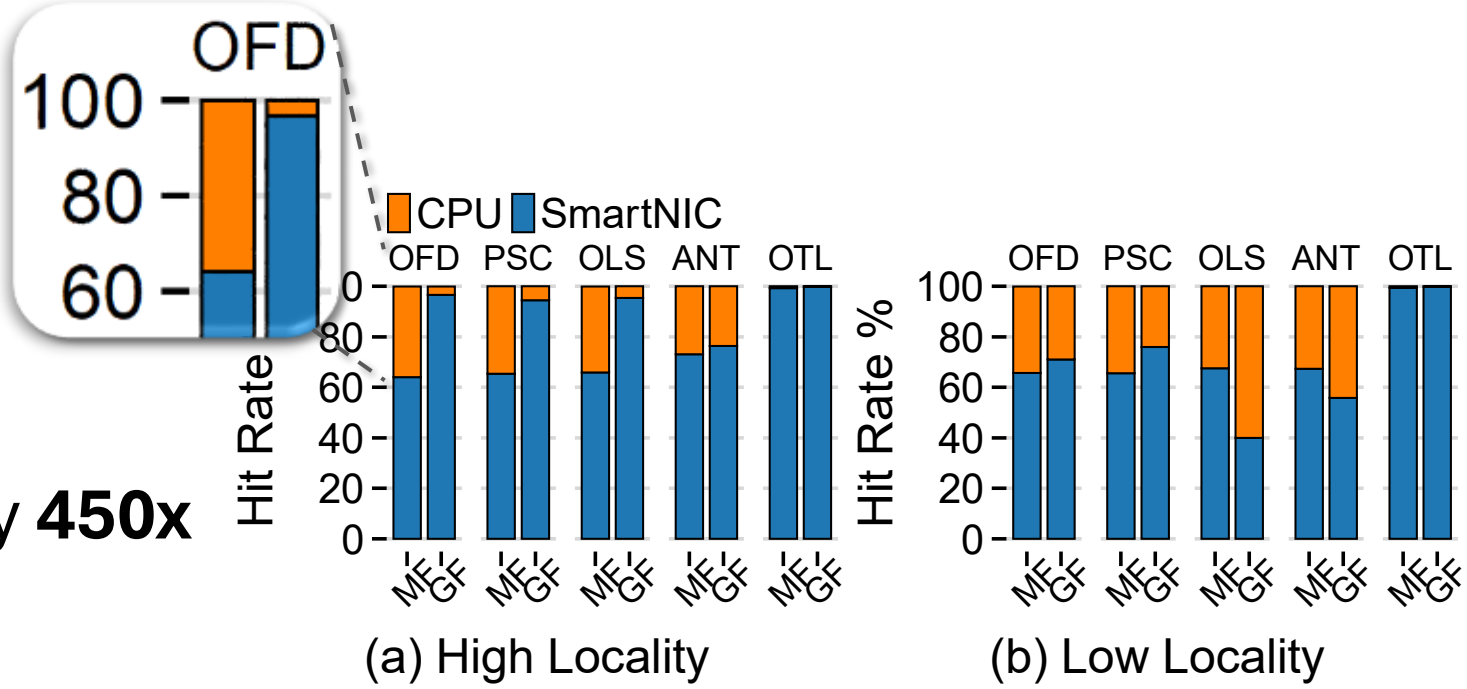
- Real-world pipeline configurations
  - tables, matching fields, traversals
- Classbench[1] rulesets to populate multi-table rules
- CAIDA[2] traces for realistic traffic patterns
- High/low locality environments
  - high/low sub-traversal sharing opportunity

[1] Taylor, David E., and Jonathan S. Turner. Classbench: A Packet Classification Benchmark. In IEEE Transactions on Networking (2004)
[2] The CAIDA UCSD Anonymized Internet Traces. https://www. caida.org/catalog/datasets/passive_dataset/

Performance Summary of Gigaflow

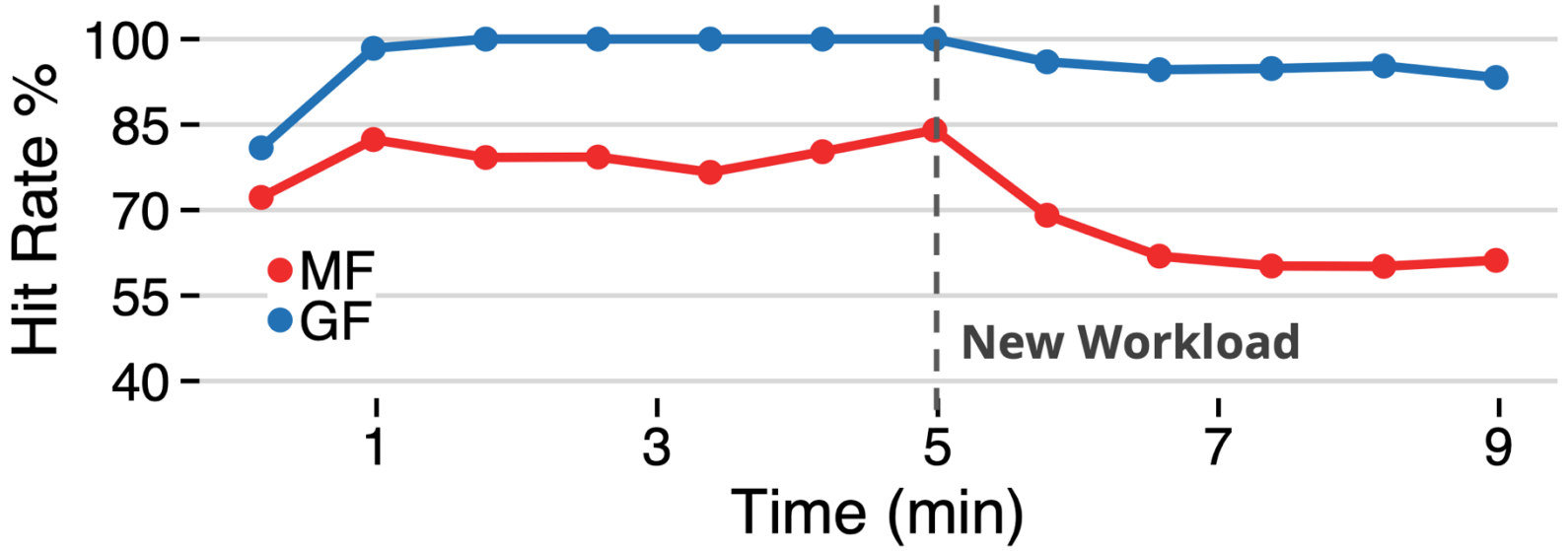
Gigaflow improves

- **#Cache misses** by **90%**
- **Cache hit rate** by **51%**
- **Rule space coverage** by **450x**
- **#Cache entries** by **18%**
- **Forwarding latency** by **30%**



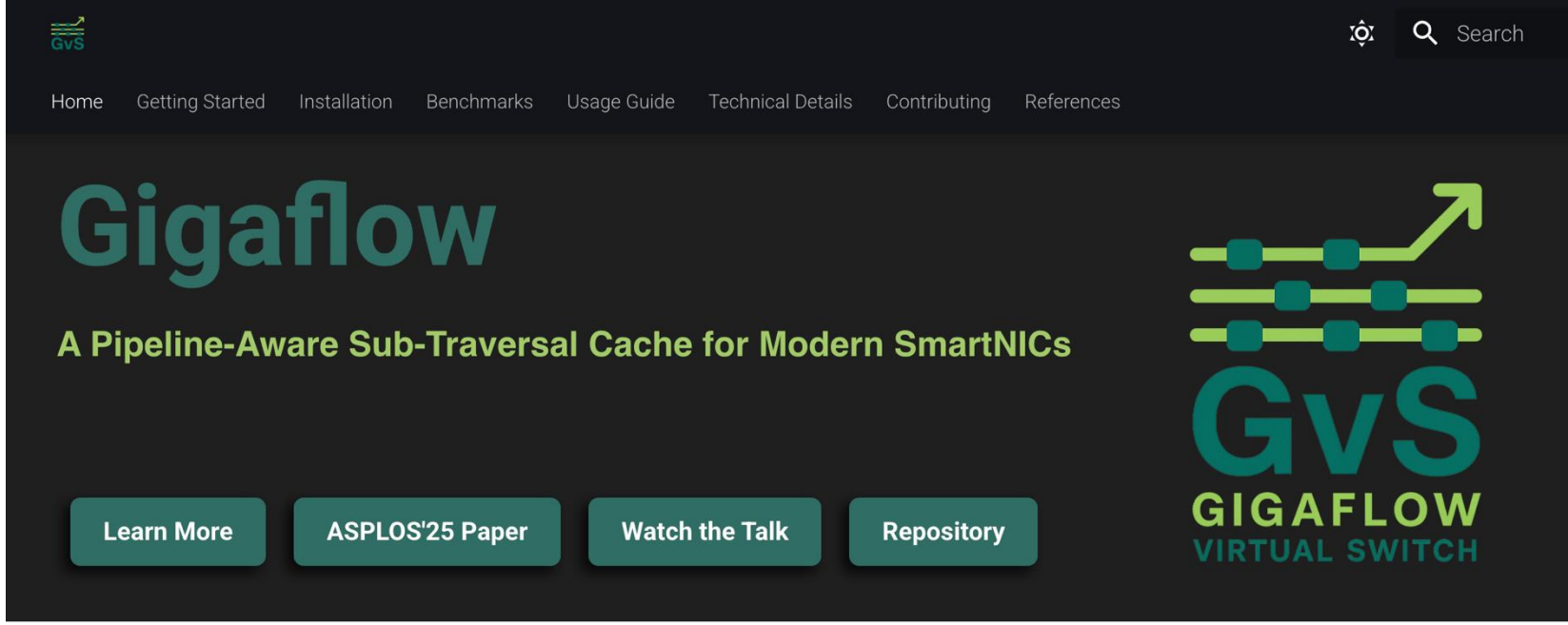
Cache hit rate of Megaflow 32K (MF)
vs Gigaflow 4x8K (GF) for real-world pipelines

Gigaflow With Dynamic Workloads



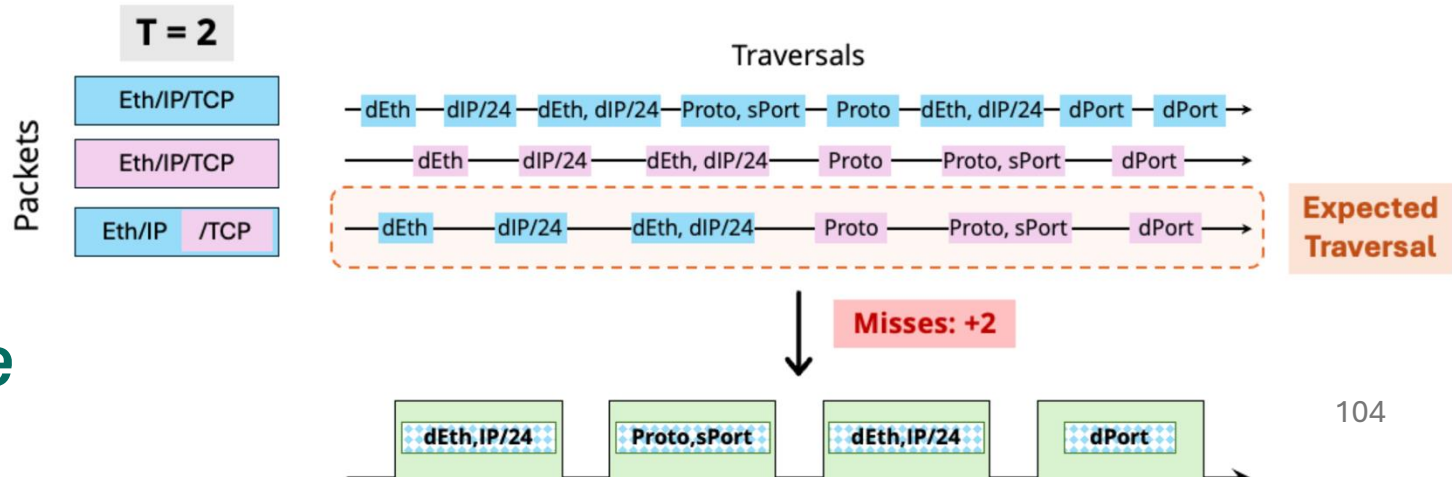
Cache hit rate of Megaflow 32K (MF)
vs Gigaflow 4x8K (GF) under dynamic workloads

Artifact

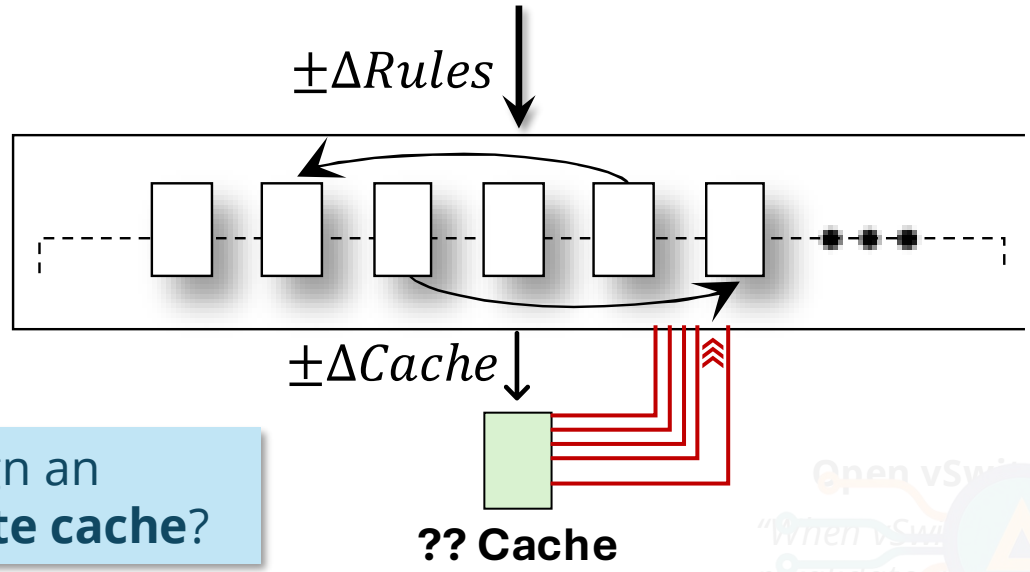


Sub-traversal cache
Pipeline-aware locality
Higher rule space coverage

How does Gigaflow work?

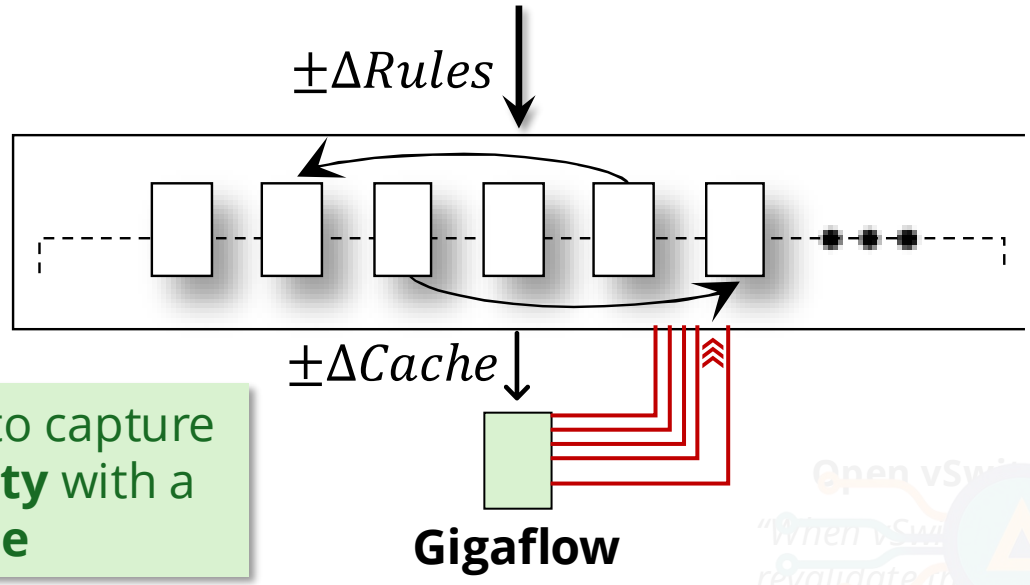


How to Rearchitect the vSwitch for Tbps Era?



Q1: How to design an **efficient, high hit rate cache**?

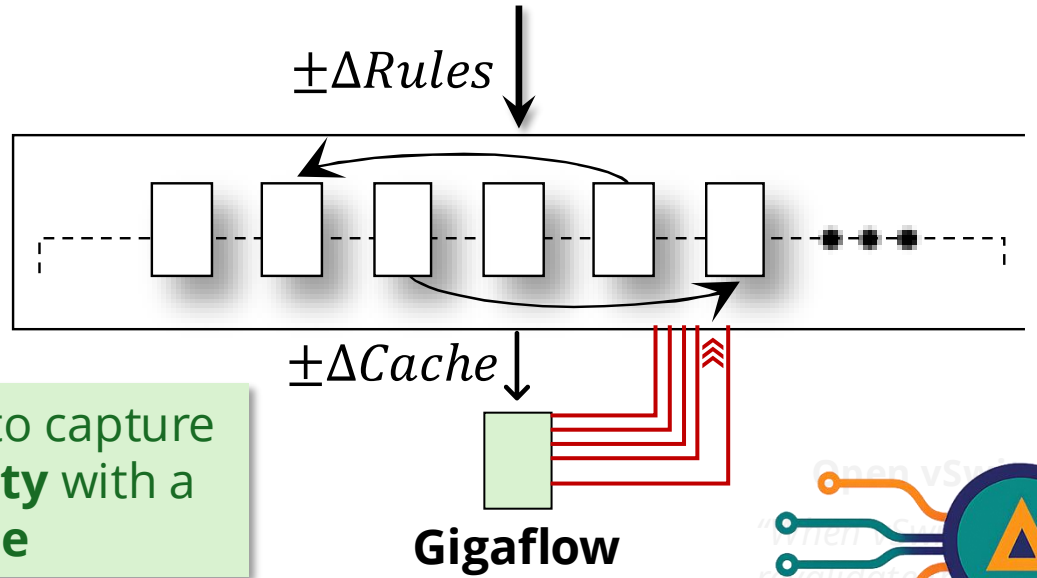
How to Rearchitect the vSwitch for Tbps Era?



Cache **sub-traversals** to capture **pipeline-aware locality** with a **Gigaflow cache**

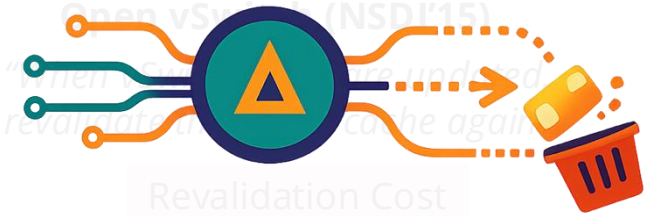
GvS
GIGAFLOW
VIRTUAL SWITCH

How to Rearchitect the vSwitch for Tbps Era?



Cache **sub-traversals** to capture **pipeline-aware locality** with a **Gigaflow cache**

GvS
GIGAFLOW
VIRTUAL SWITCH



Kairo Cache Eviction

Q2: How to quickly **evict stale cache** when **vSwitch rules are updated?**

3MT® THREE MINUTE THESIS
FOUNDED BY THE UNIVERSITY OF QUEENSLAND

SIGCOMM 2026
DENVER
*In Submission

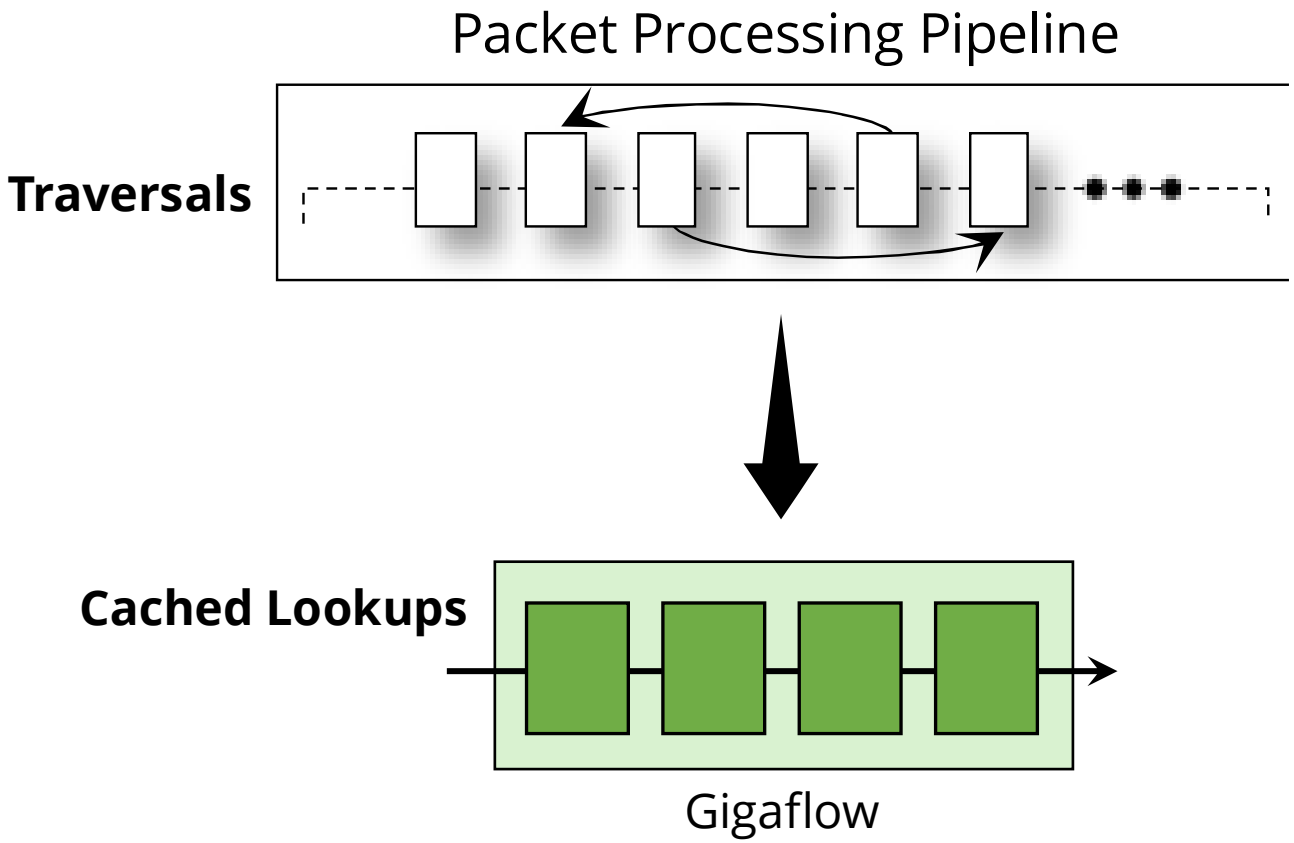
Kairo: Incremental Cache Eviction for Modern vSwitches

Annus Zulfiqar

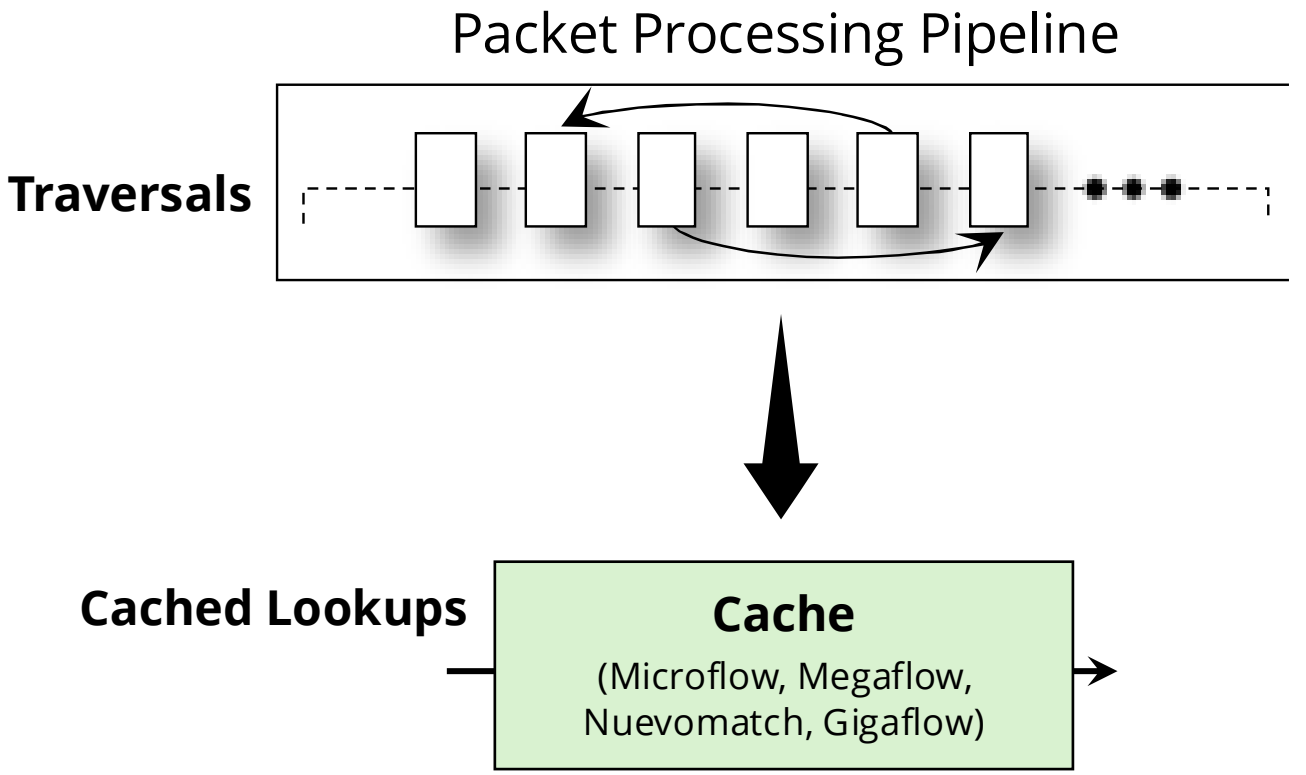
Ben Pfaff, Gianni Antichi, Muhammad Shahbaz

UNIVERSITY OF MICHIGAN
feldera
POLITECNICO MILANO 1863

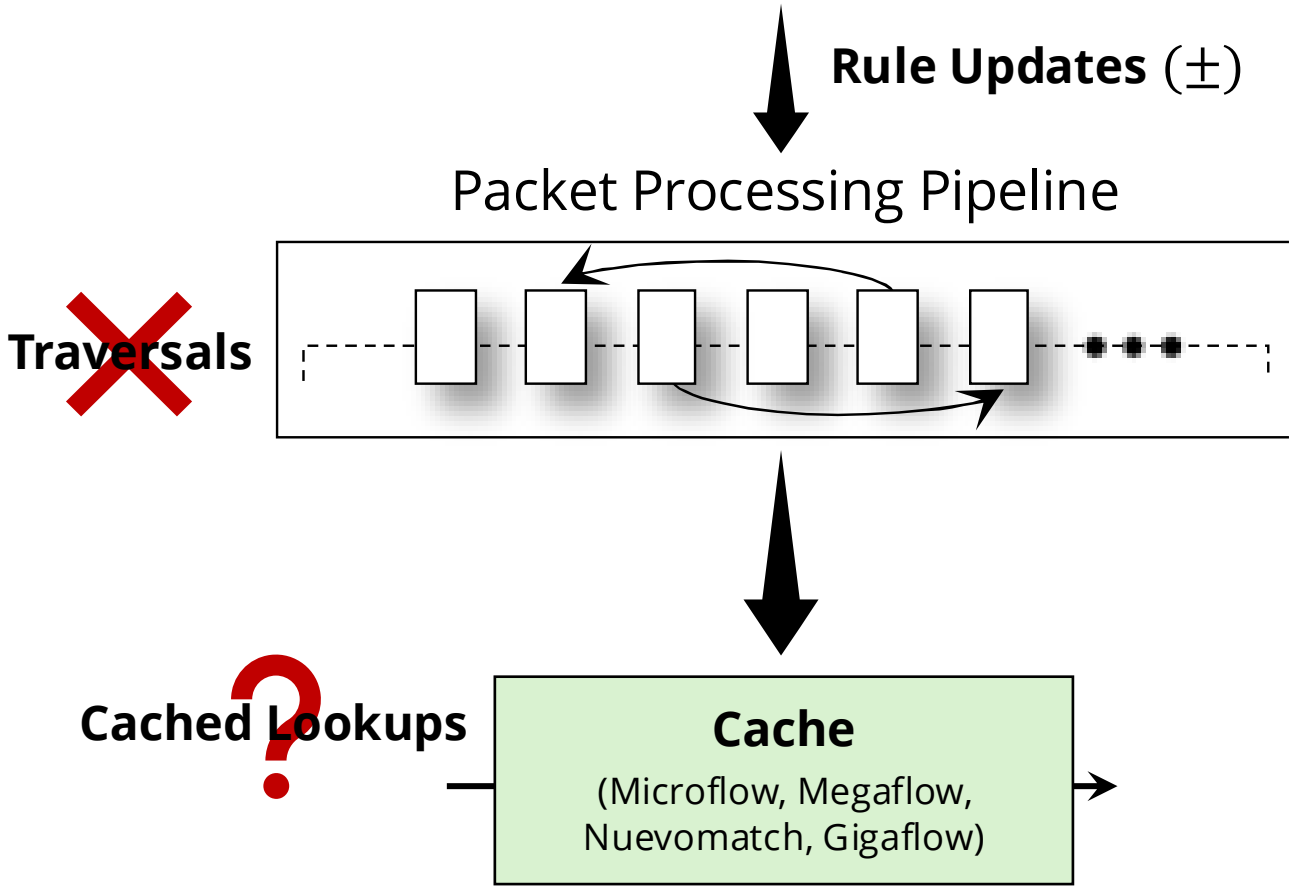
Caching is Key to vSwitch Performance



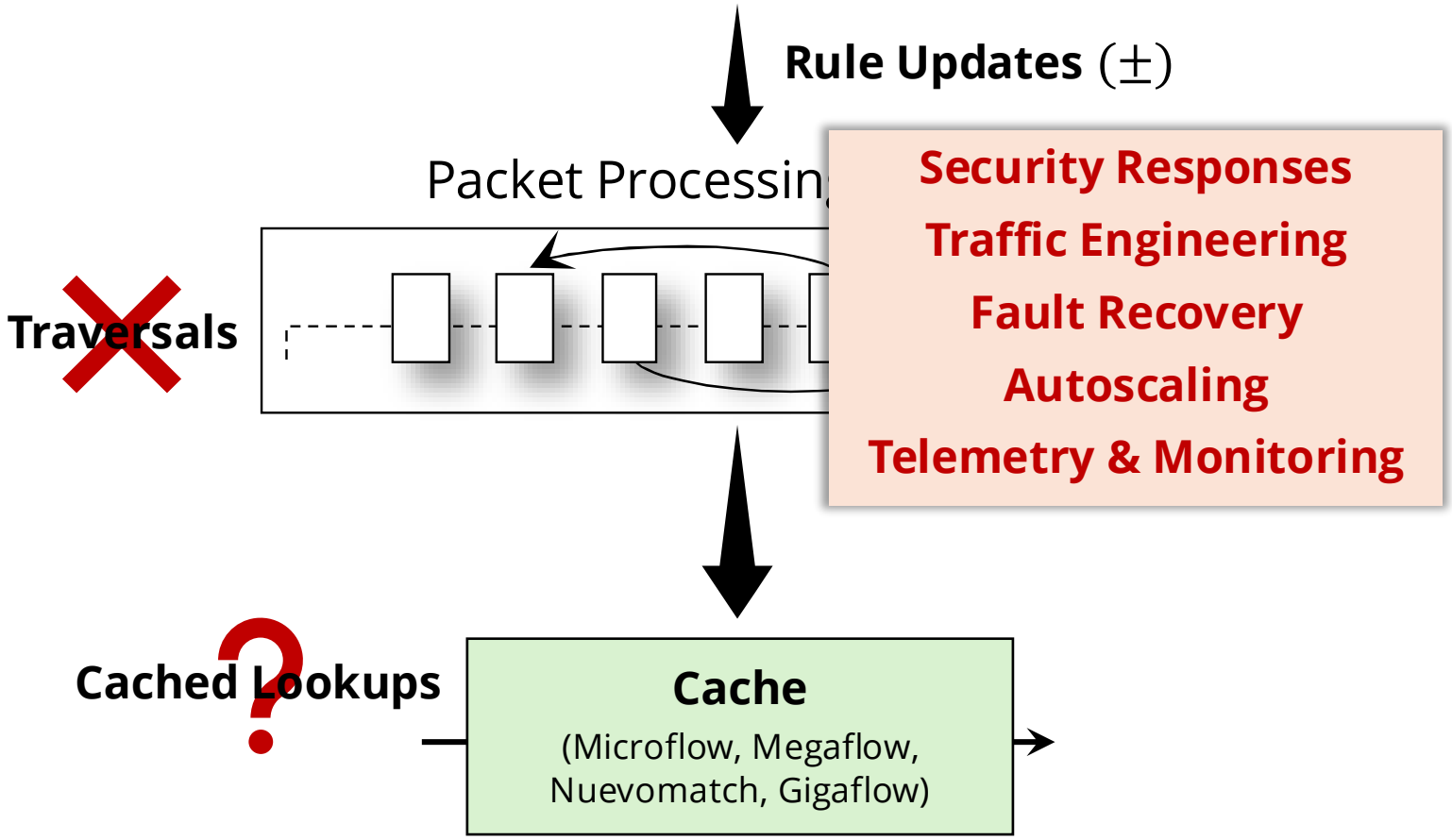
Caching is Key to vSwitch Performance



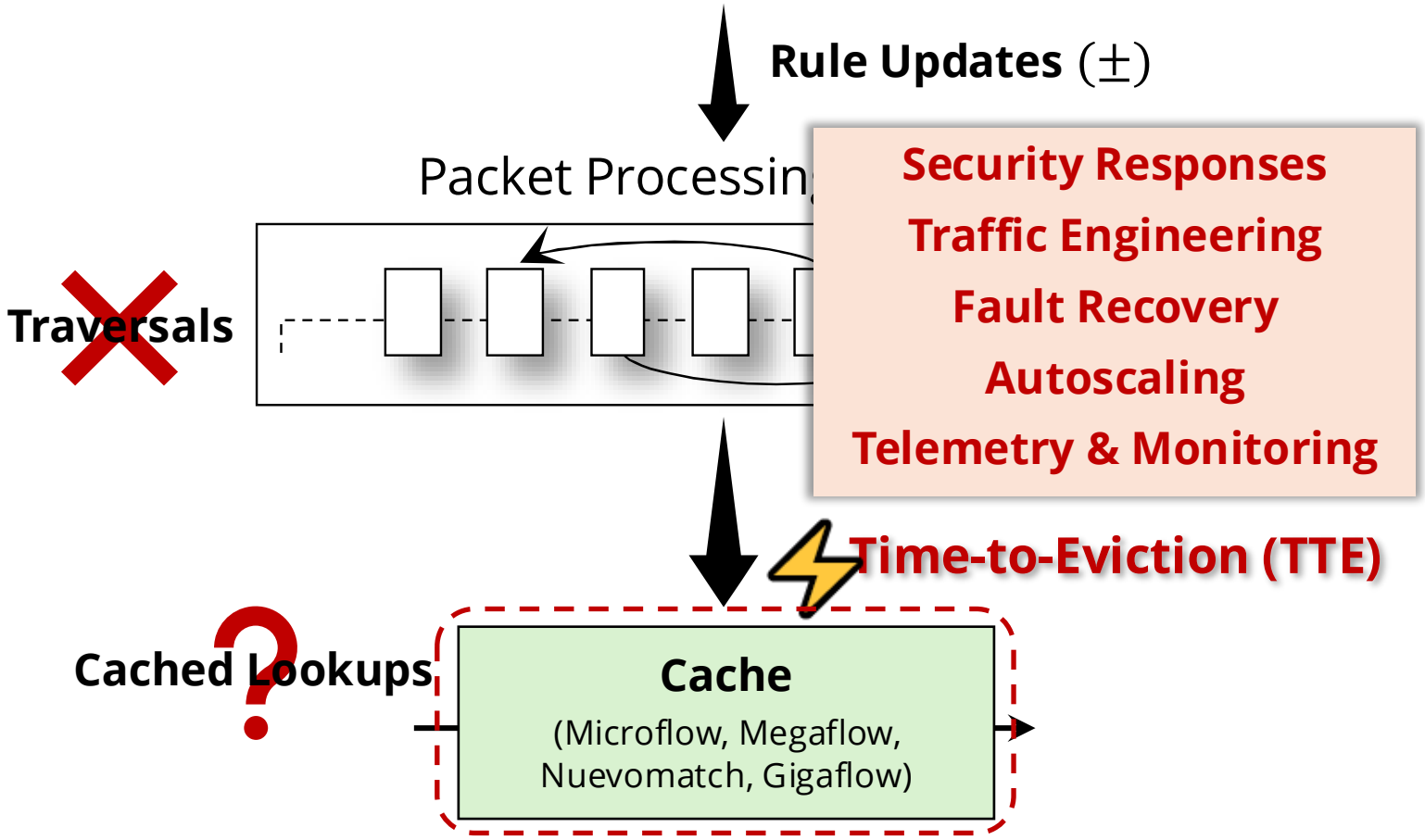
vSwitch Rules are Updated Frequently



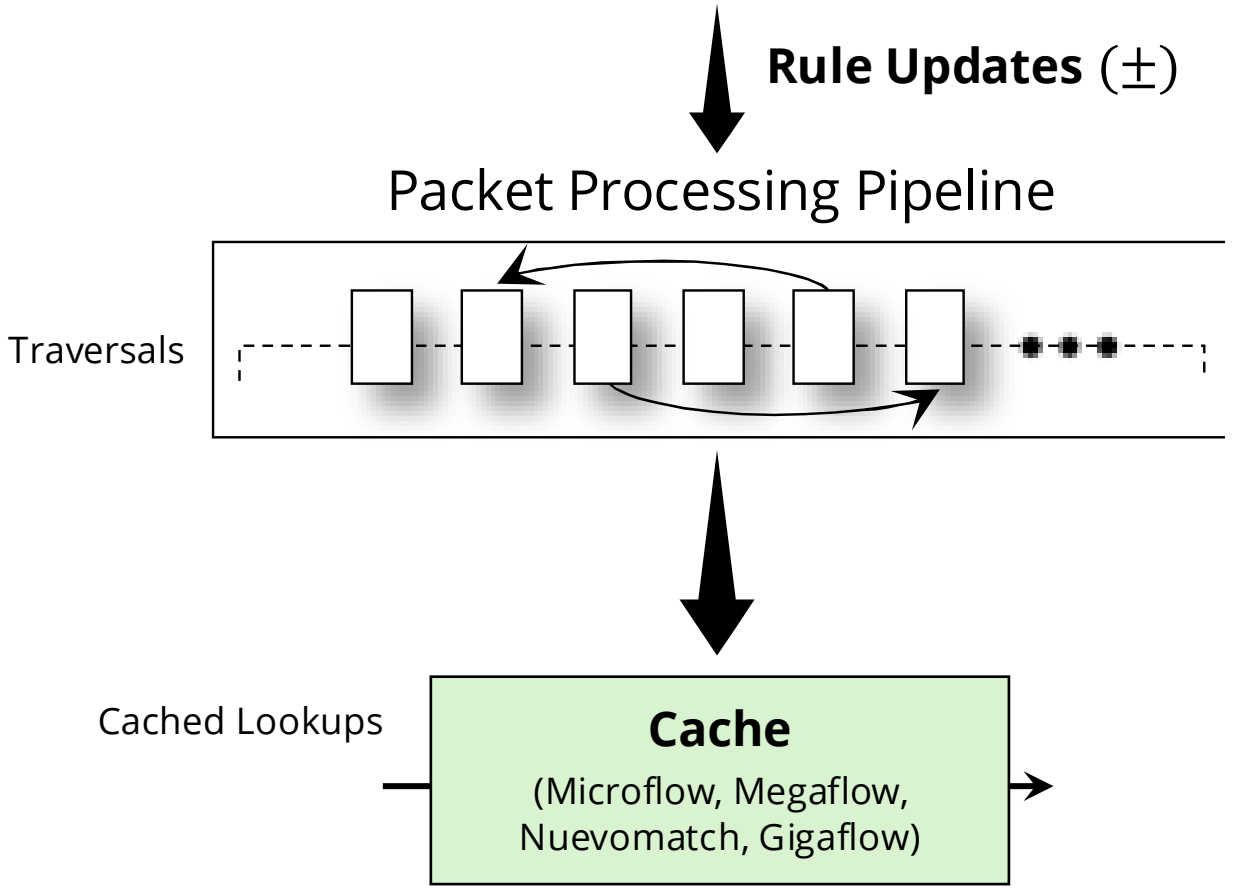
Rule Updates Follow Diverse Patterns



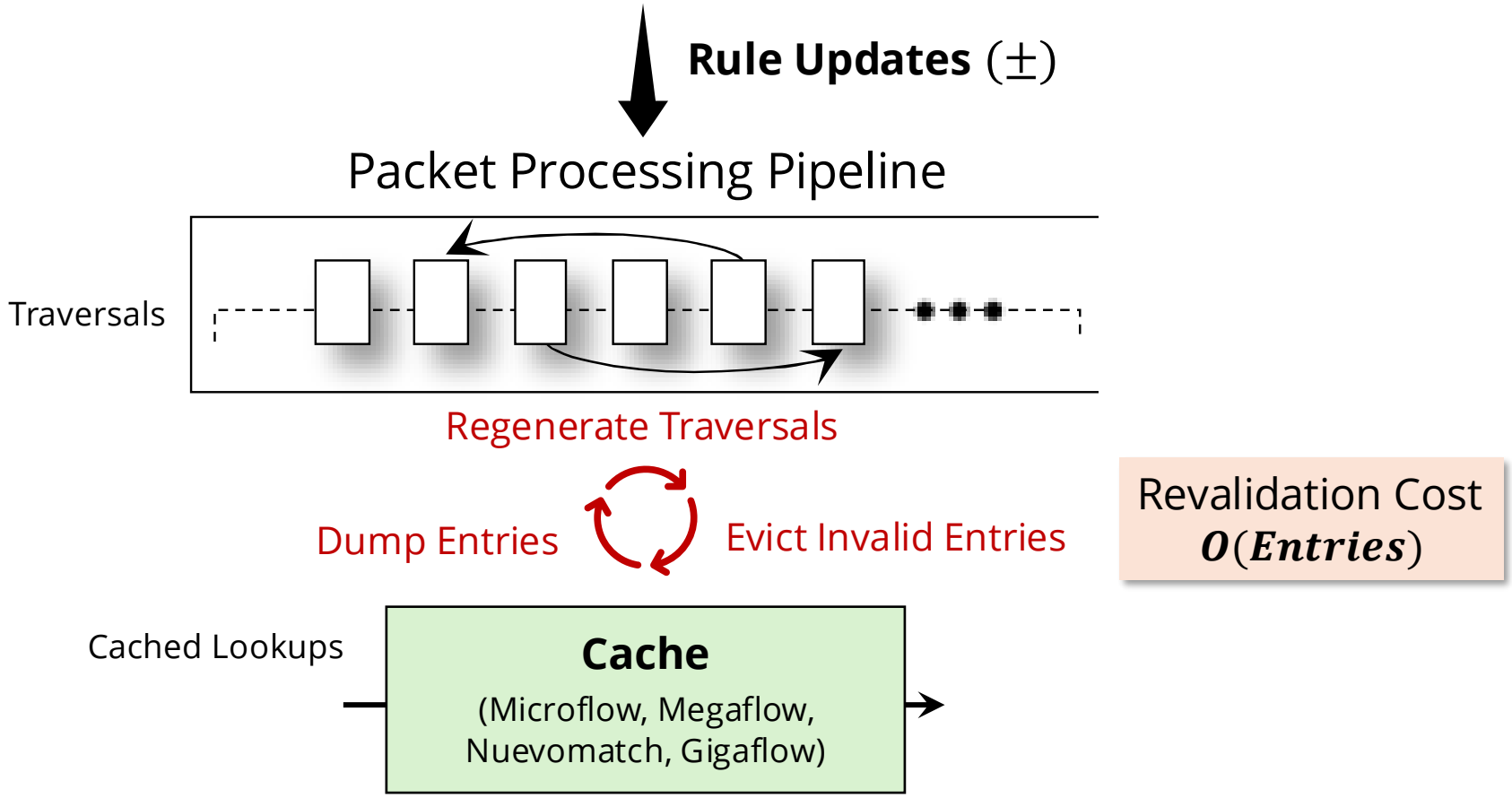
Time-to-Eviction → Stale Cache Processing



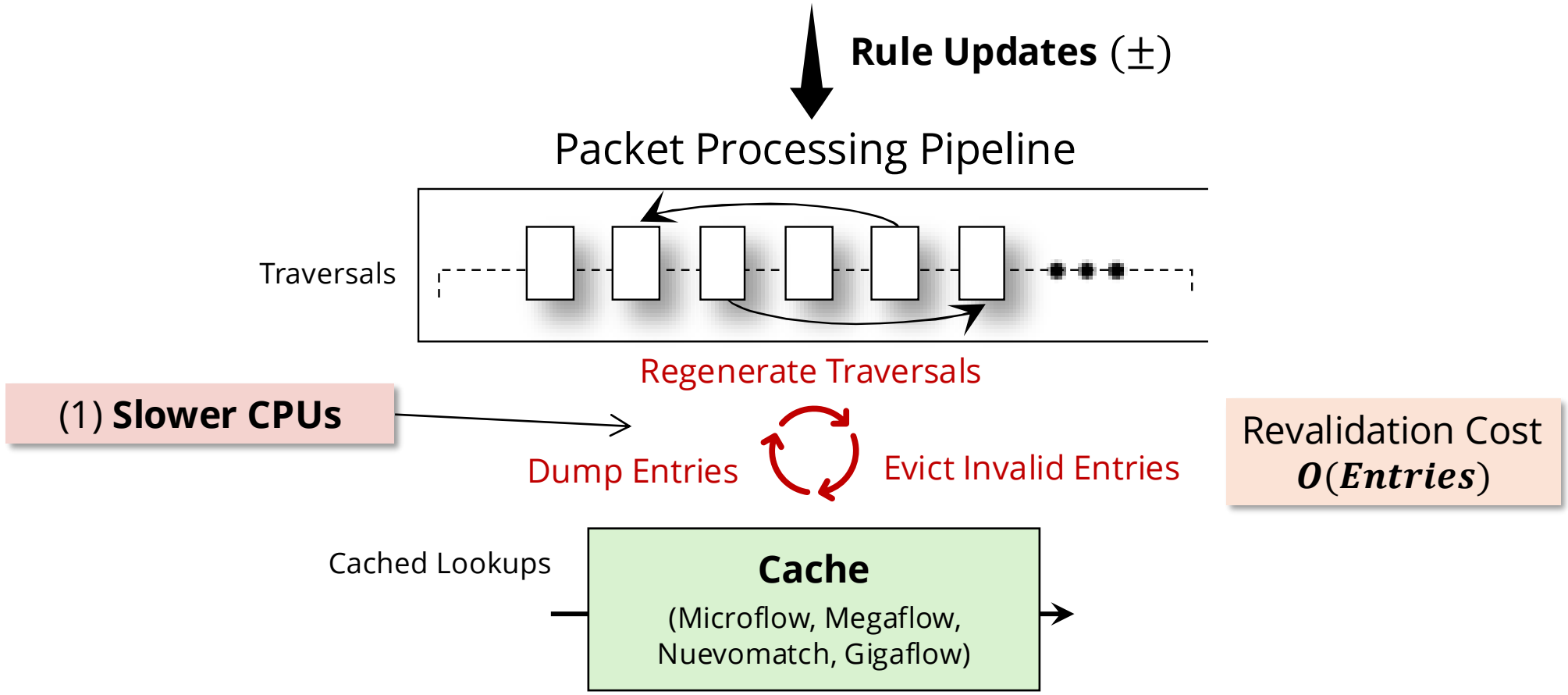
Solution? Revalidate the Whole Cache!!



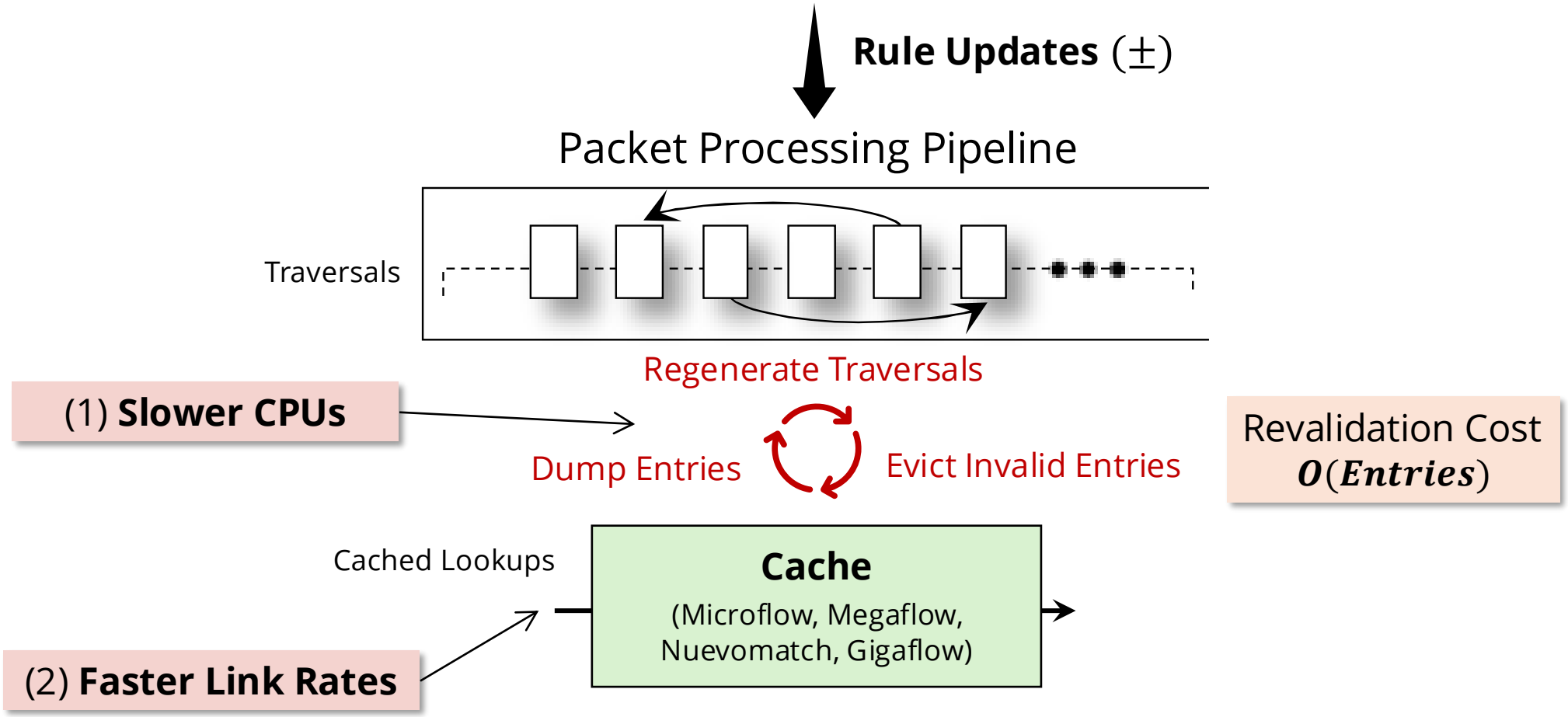
Solution? Revalidate the Whole Cache!!



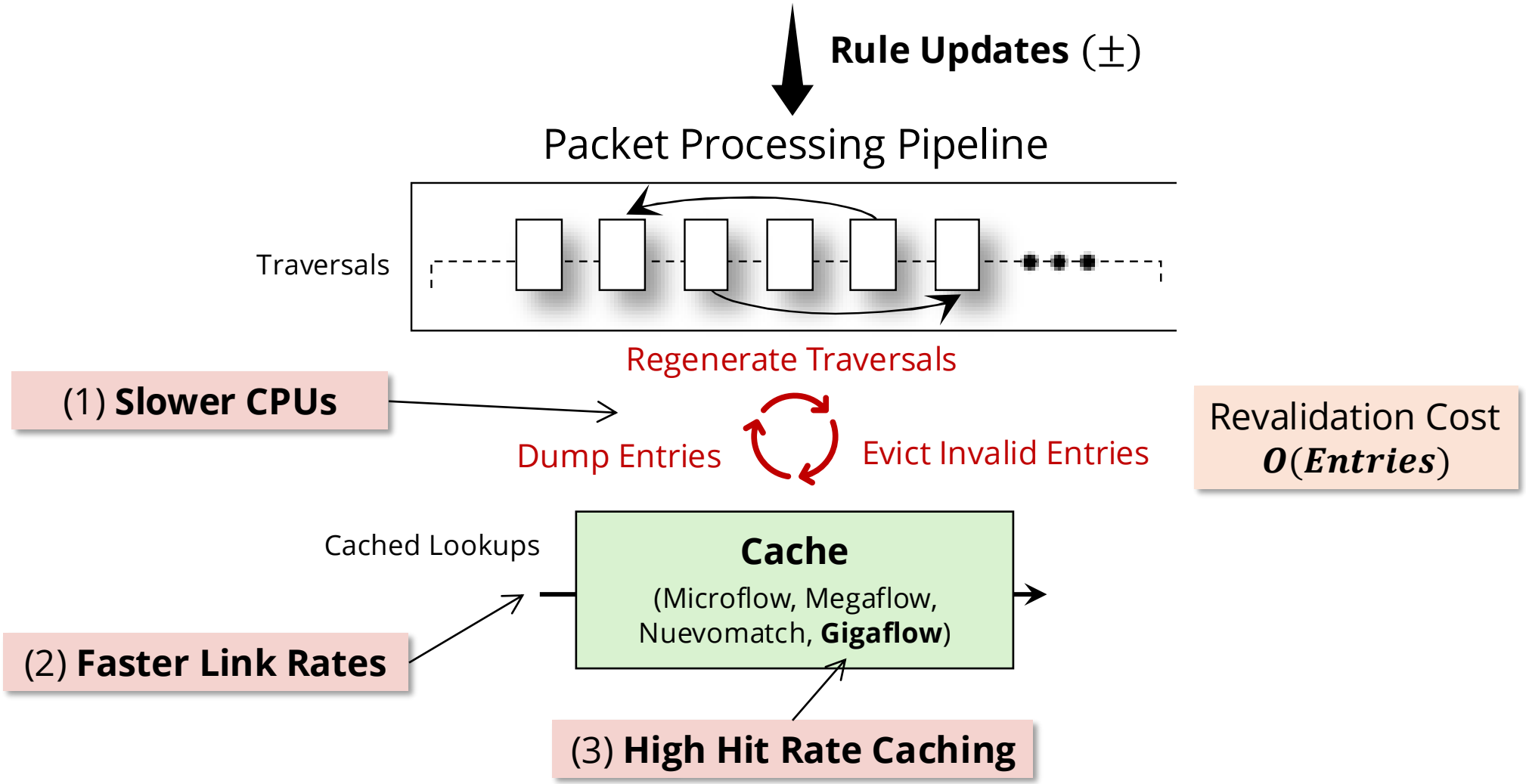
Solution? Revalidate the Whole Cache!!



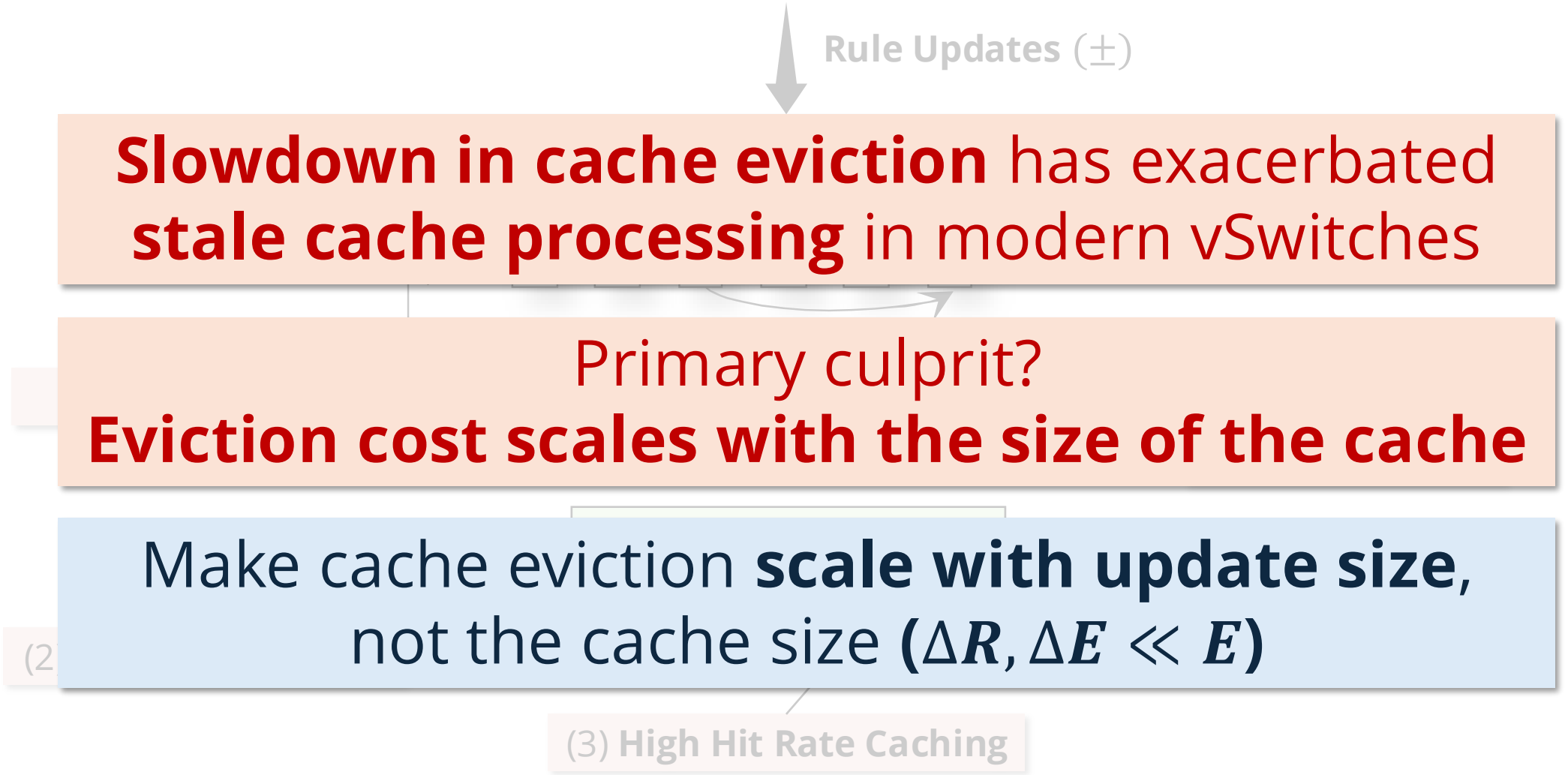
Solution? Revalidate the Whole Cache!!



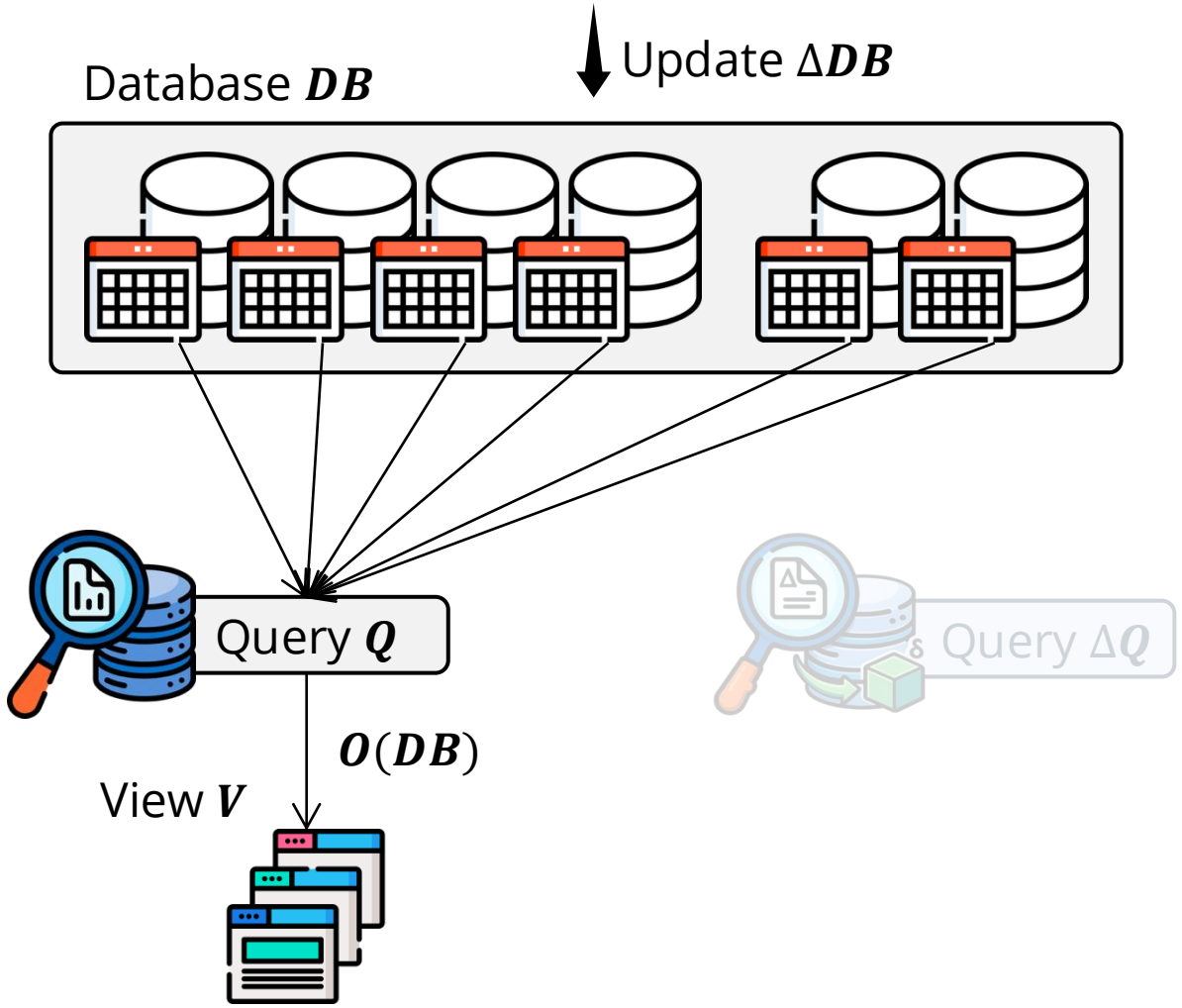
Solution? Revalidate the Whole Cache!!



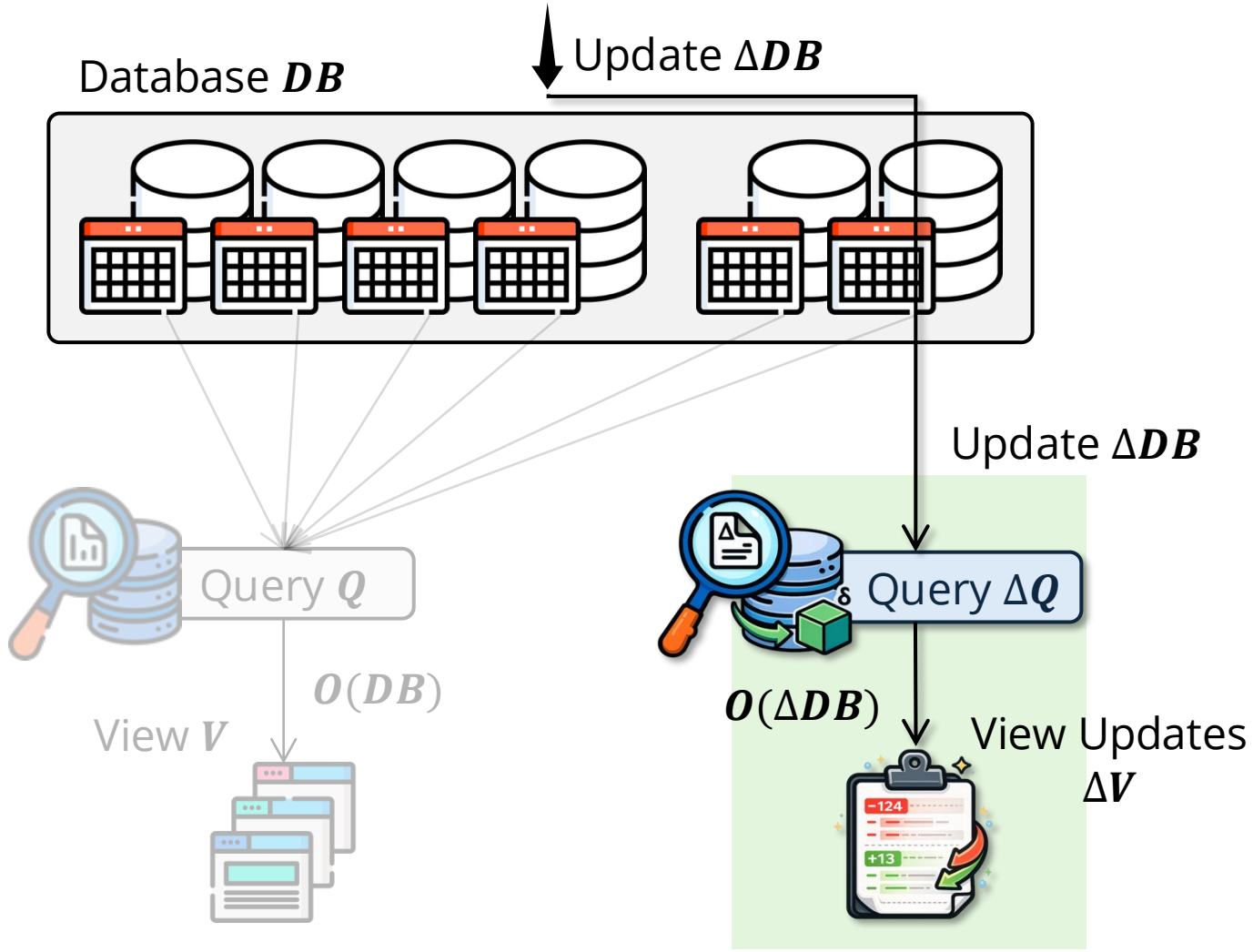
Solution? Revalidate the Whole Cache!!



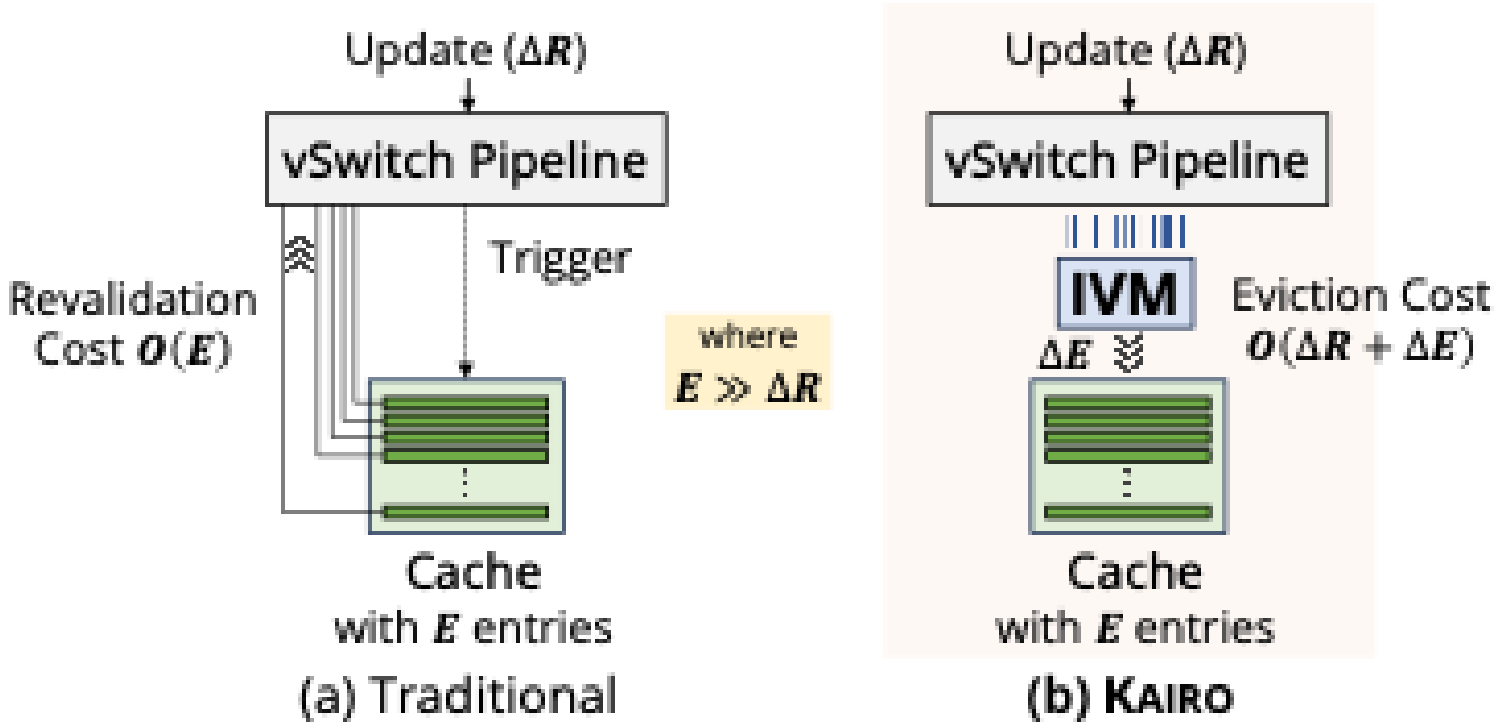
Traditional Database View Maintenance



Incremental View Maintenance (IVM)

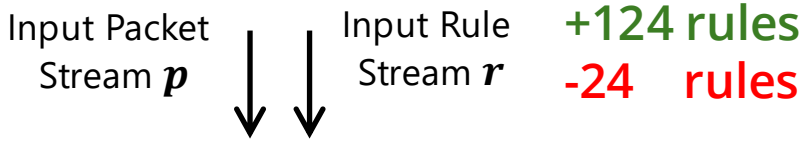


Towards Incremental Cache Eviction



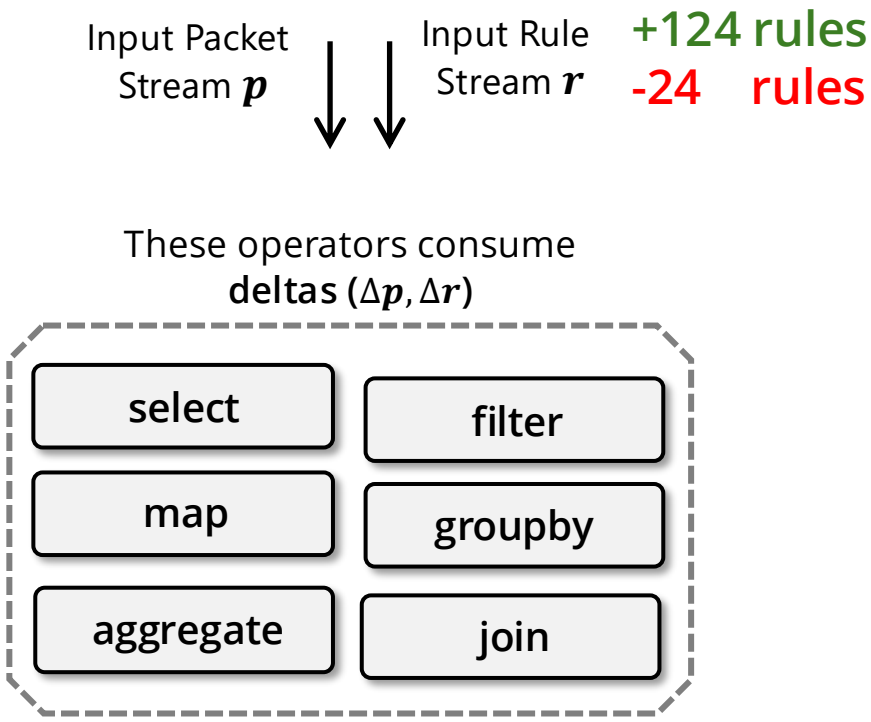
Incremental View Maintenance with DBSP

- A **language** and **execution model**
- Supports automatic *incrementalization*
- Key abstractions:
  - Stream



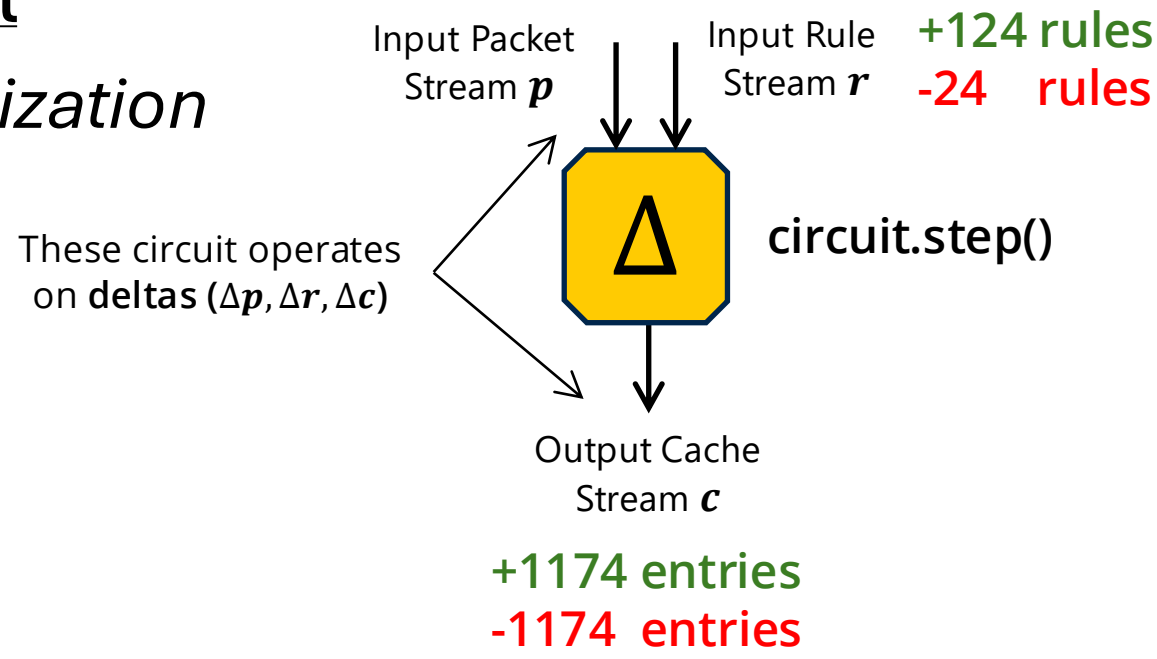
Incremental View Maintenance with DBSP

- A **language** and **execution model**
- Supports automatic *incrementalization*
- Key abstractions:
  - Stream
  - Operators (incremental by design)



Incremental View Maintenance with DBSP

- A **language** and **execution model**
- Supports automatic *incrementalization*
- Key abstractions:
  - Stream
  - Operators (incremental by design)
  - Circuits



Incremental View Maintenance with DBSP

- A lar
- Supp
- Key
  - S
  - O
  - Ci

C1: How to represent the **vSwitch pipeline** as **an incremental circuit?**

C2: How to **control execution** of Kairo circuits to **minimize the TTE?**

C3: How to support **arbitrary recirculation** across vSwitch tables?

C4: How to **update circuits** at runtime when rule updates add **new traversals**?

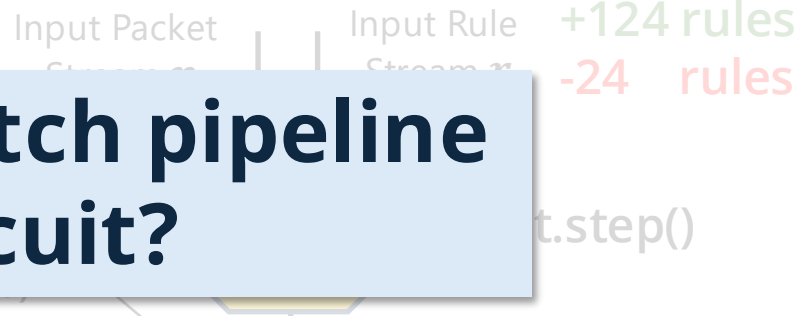
+124 rules
-24 rules

t.step()

Output Cache

Incremental View Maintenance with DBSP

- A language and execution model
- Supp
- Key
  - Stream
  - O
  - C



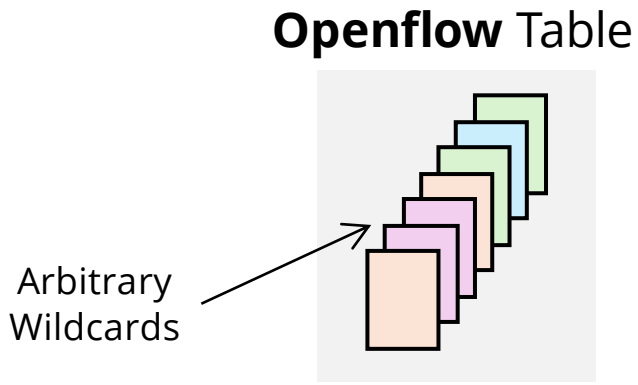
C1: How to represent the **vSwitch pipeline** as **an incremental circuit?**

C2: How to **control execution** of Kairo circuits to **minimize the TTE?**

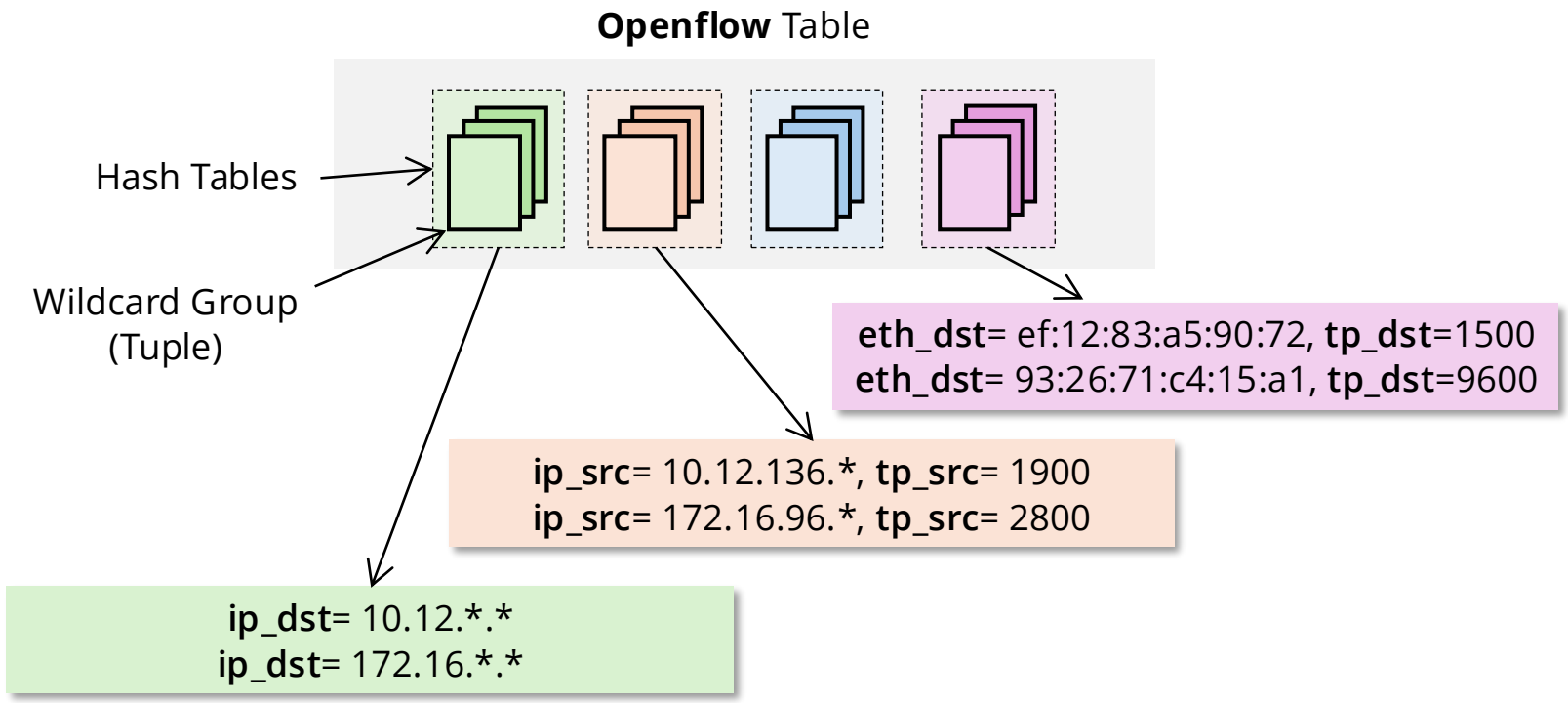
across vSwitch tables?

C4: How to **update circuits** at runtime when rule updates add **new traversals**?

Understanding Tuple Space Search

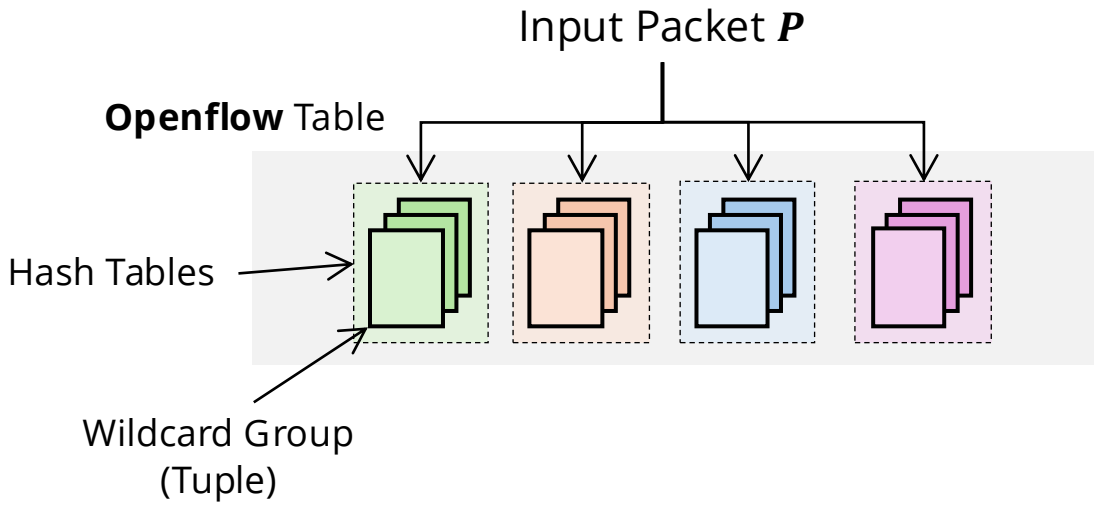


Understanding Tuple Space Search

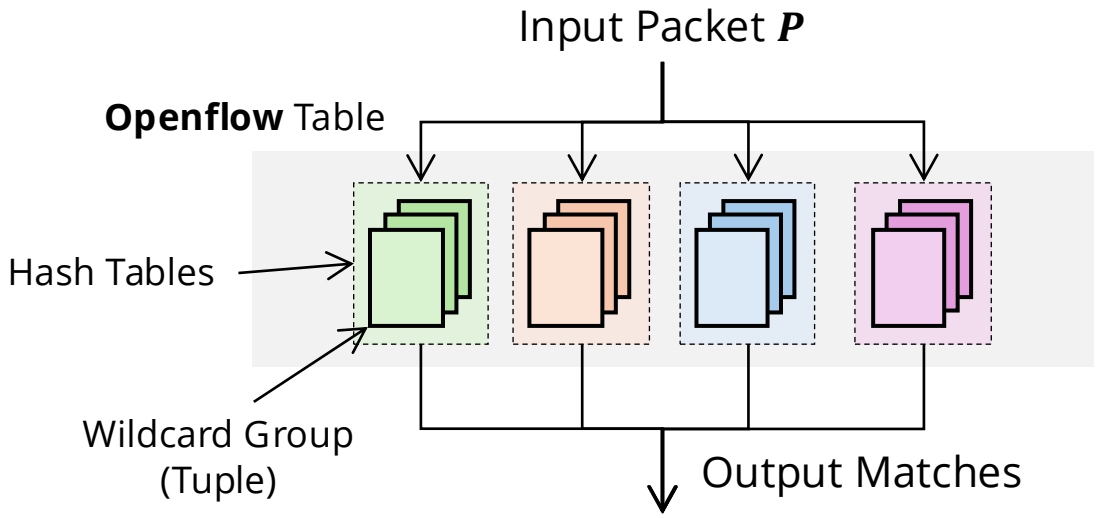


Each **tuple** is just an **exact match!**

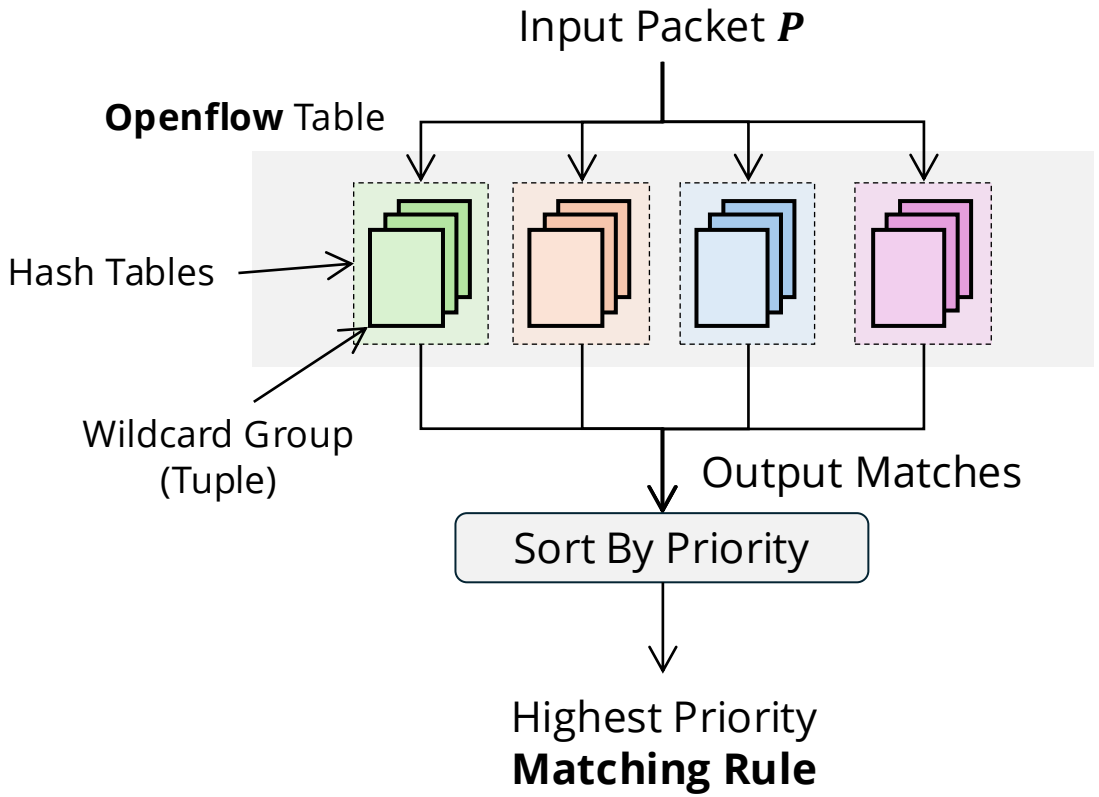
Understanding Tuple Space Search



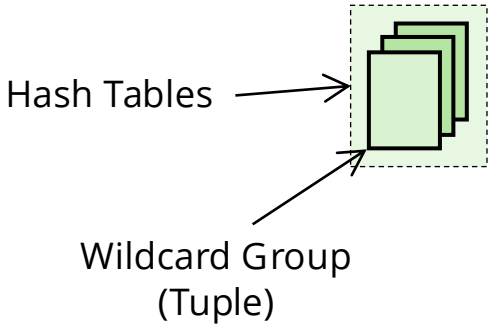
Understanding Tuple Space Search



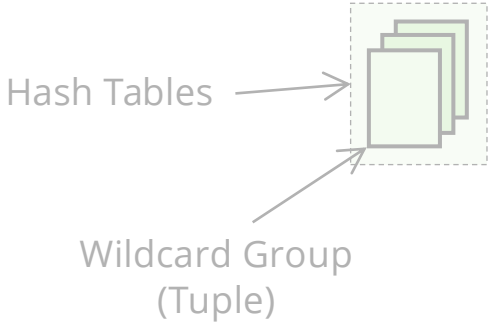
Understanding Tuple Space Search

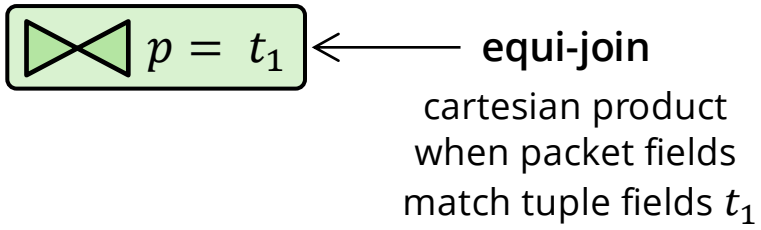


C1: Represent vSwitch Lookups in DBSP

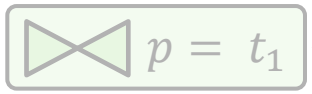


Step 1: Implement a Tuple (Exact Match) in DBSP





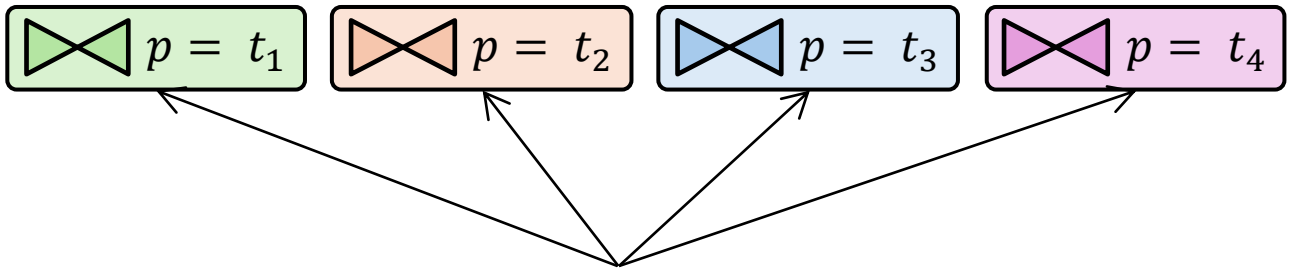
Step 2: Support Arbitrary Wildcards



equi-join

cartesian product
when packet fields
match tuples fields t_1

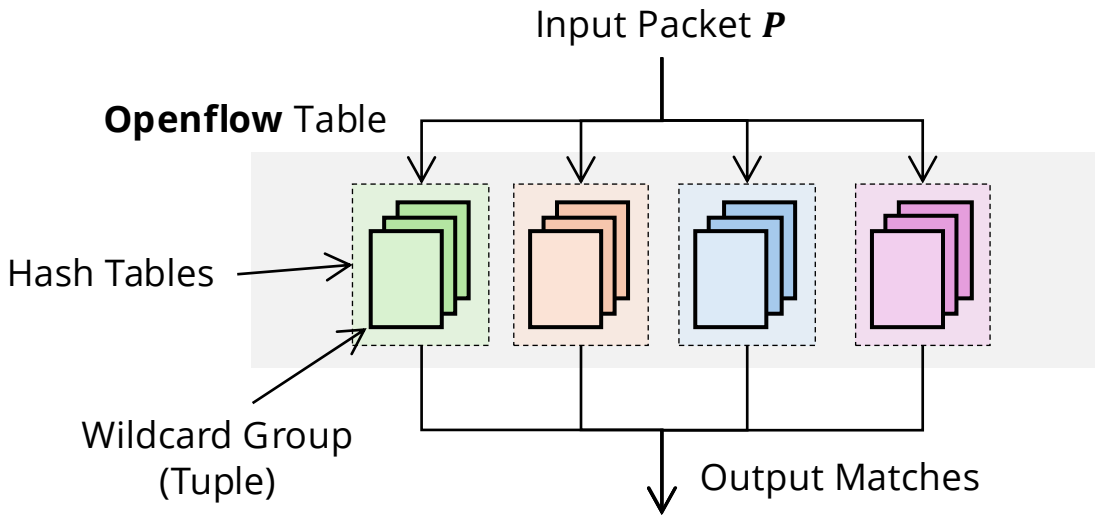
Step 2: Support Arbitrary Wildcards

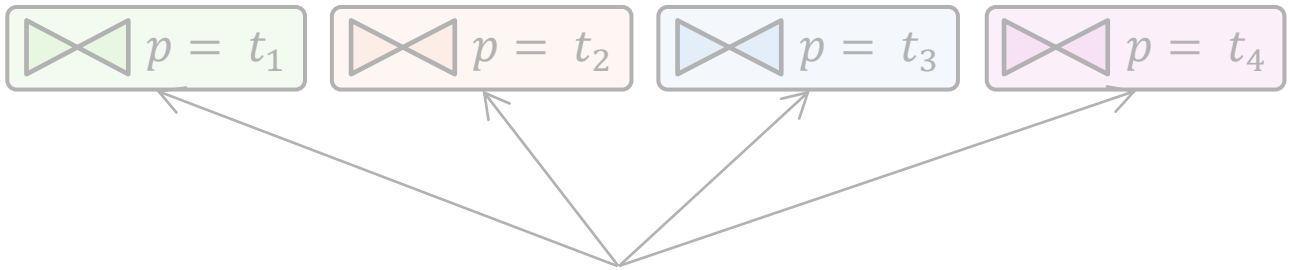


equi-join

cartesian products when
packet fields match
tuple fields t_1, t_2, t_3, t_4

Step 2: Support Arbitrary Wildcards

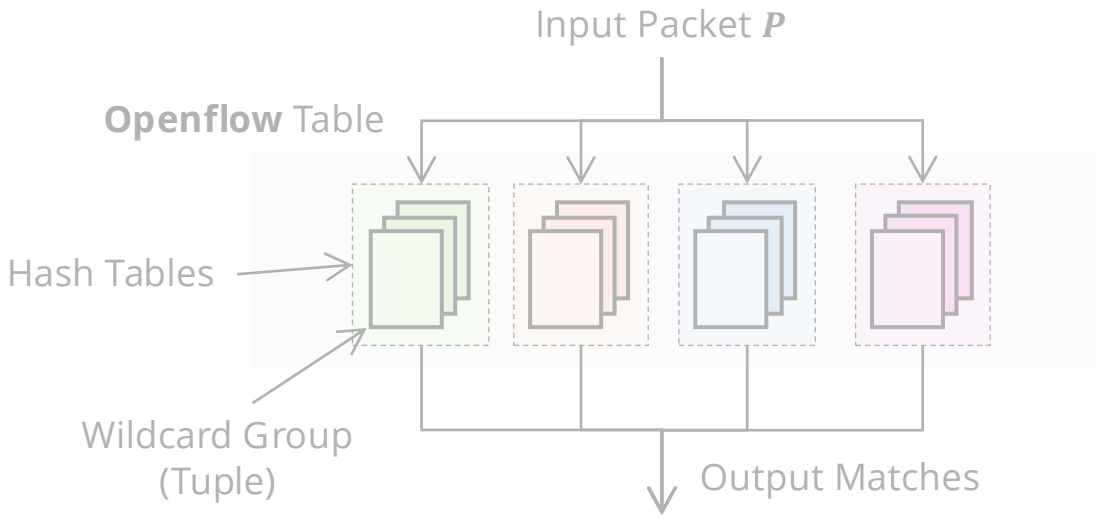


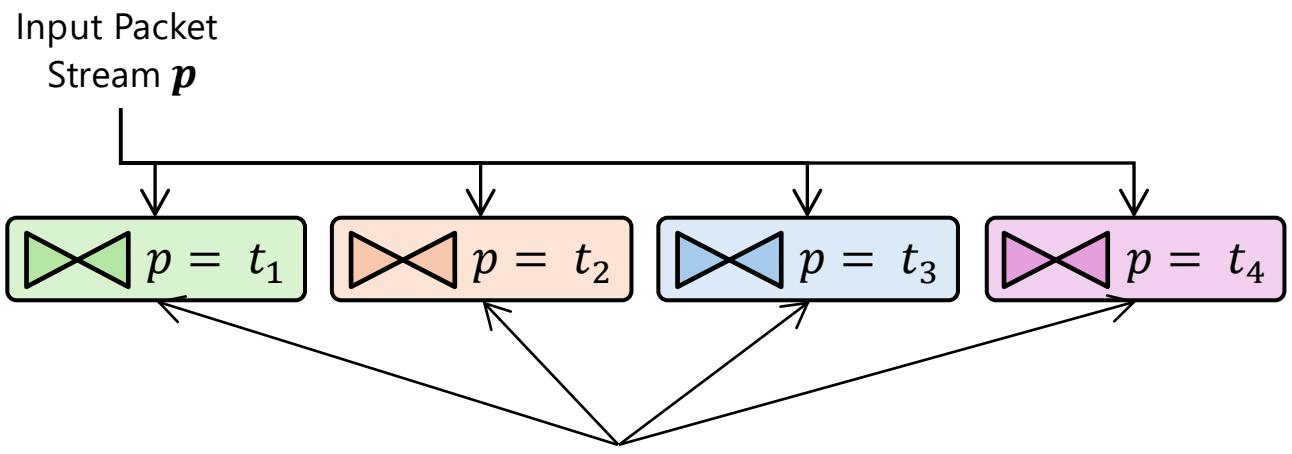


equi-join

cartesian products when
packet fields match
tuple fields t_1, t_2, t_3, t_4

Step 2: Support Arbitrary Wildcards

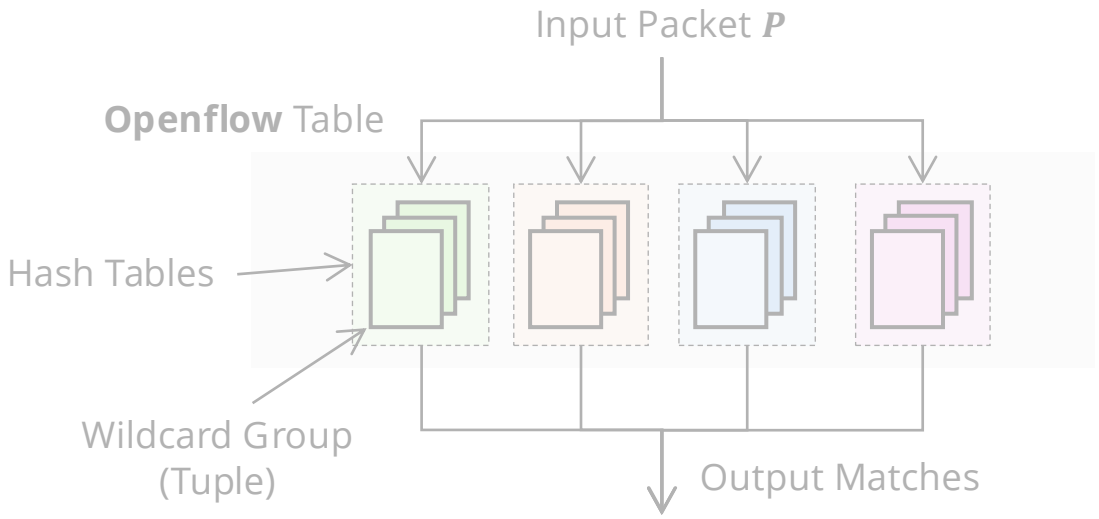


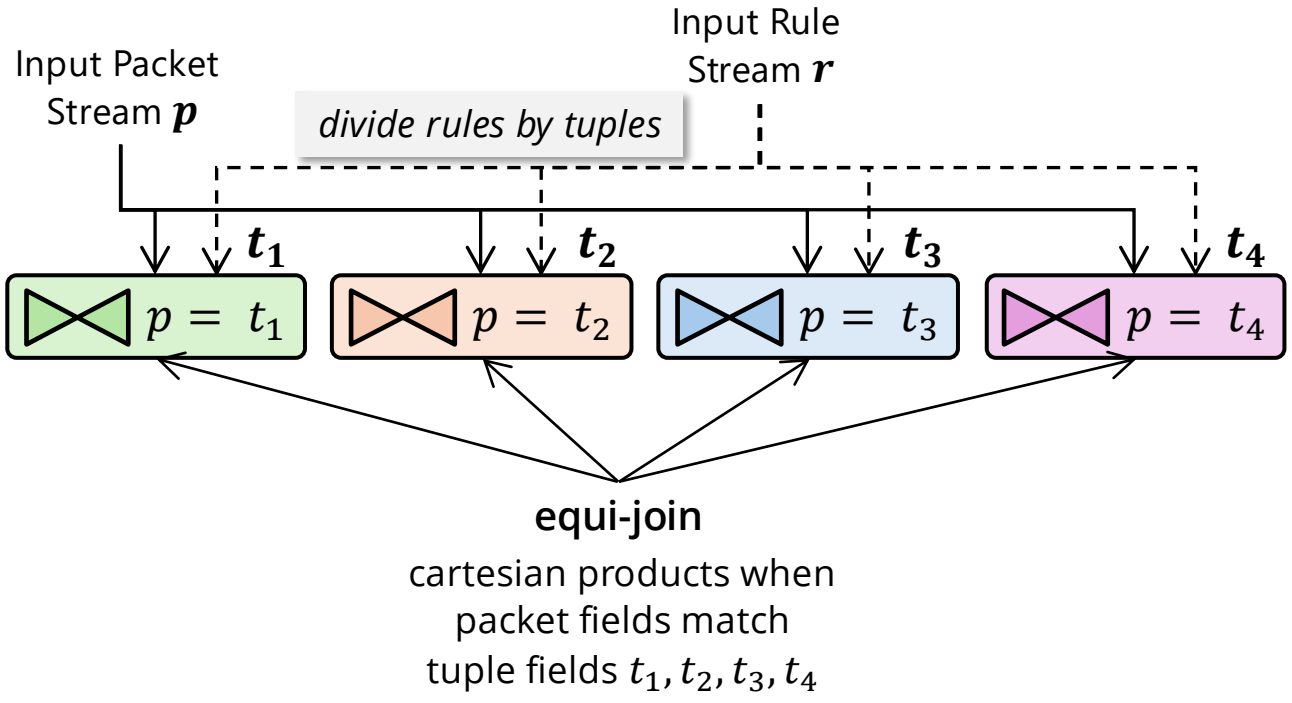


equi-join

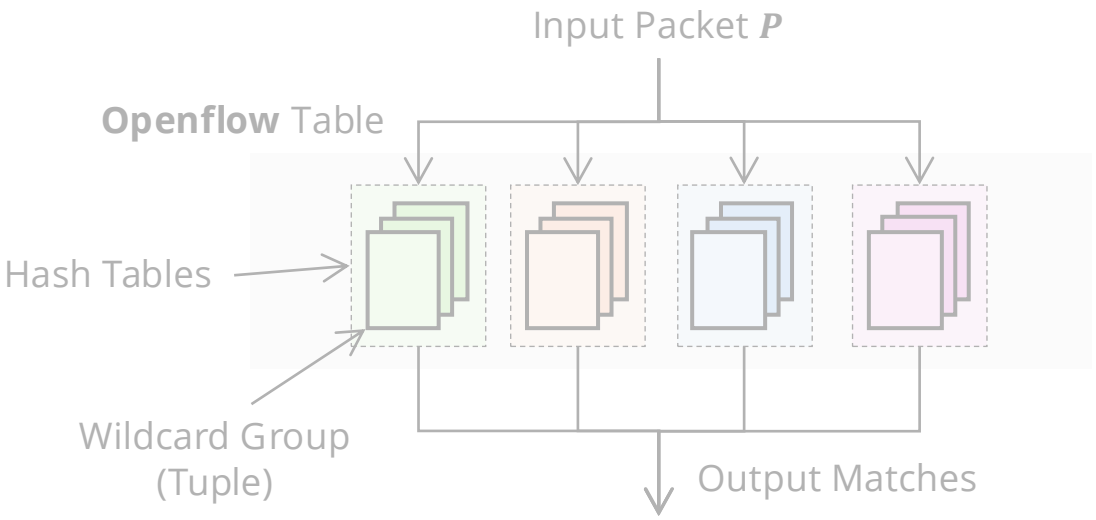
cartesian products when
packet fields match
tuple fields t_1, t_2, t_3, t_4

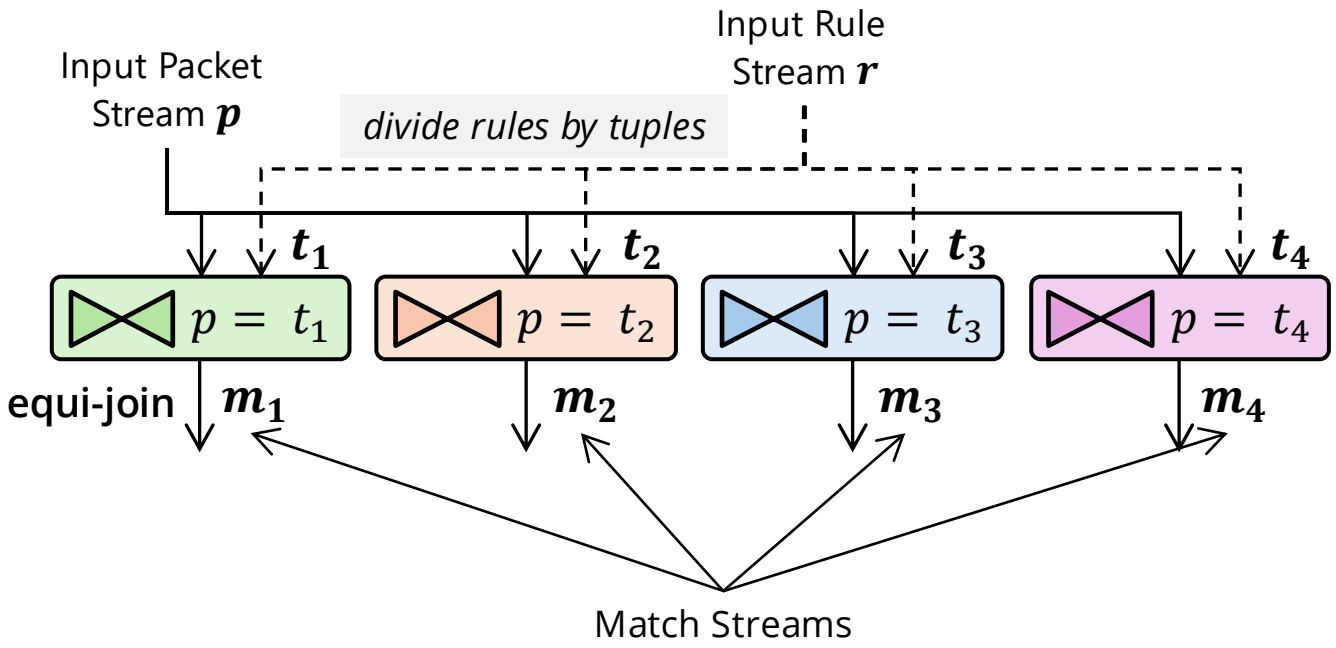
Step 2: Support Arbitrary Wildcards



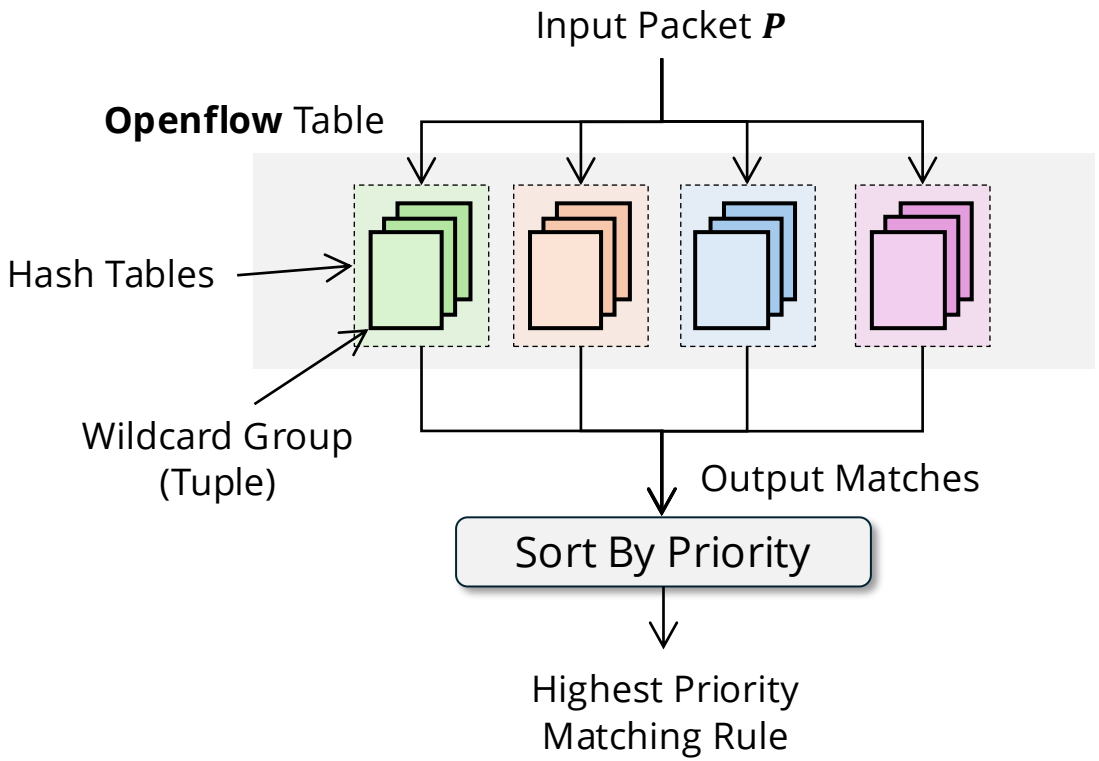


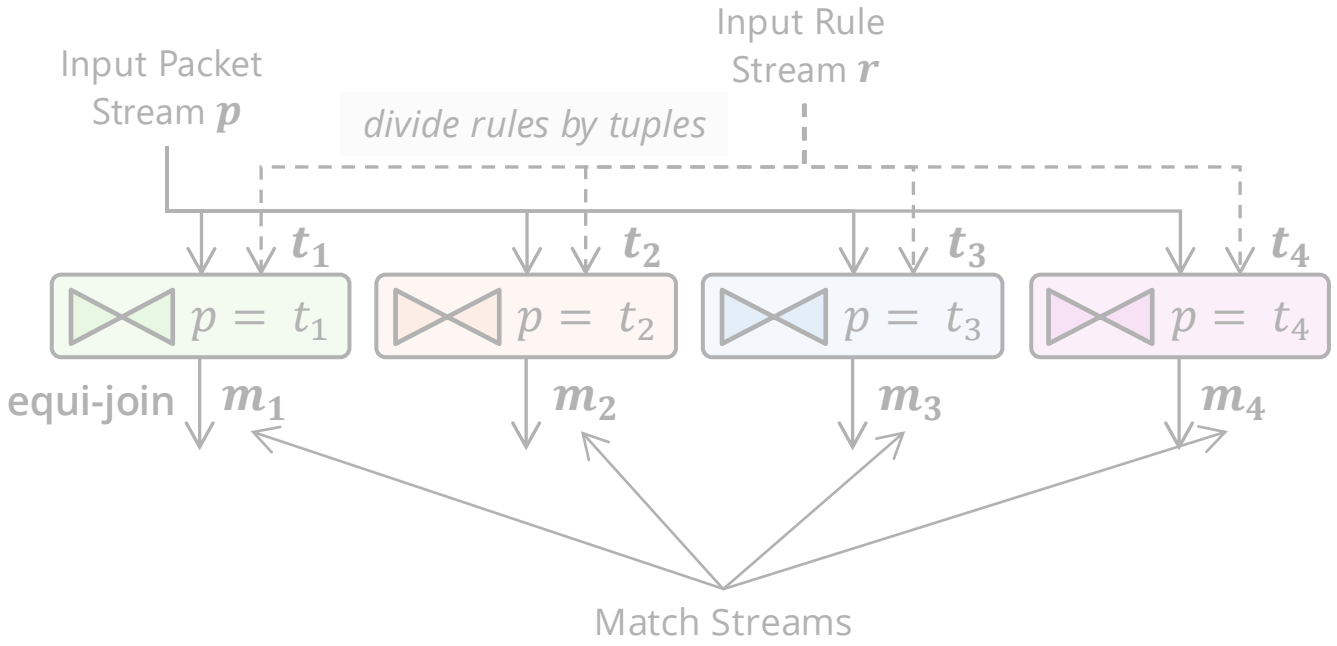
Step 2: Support Arbitrary Wildcards



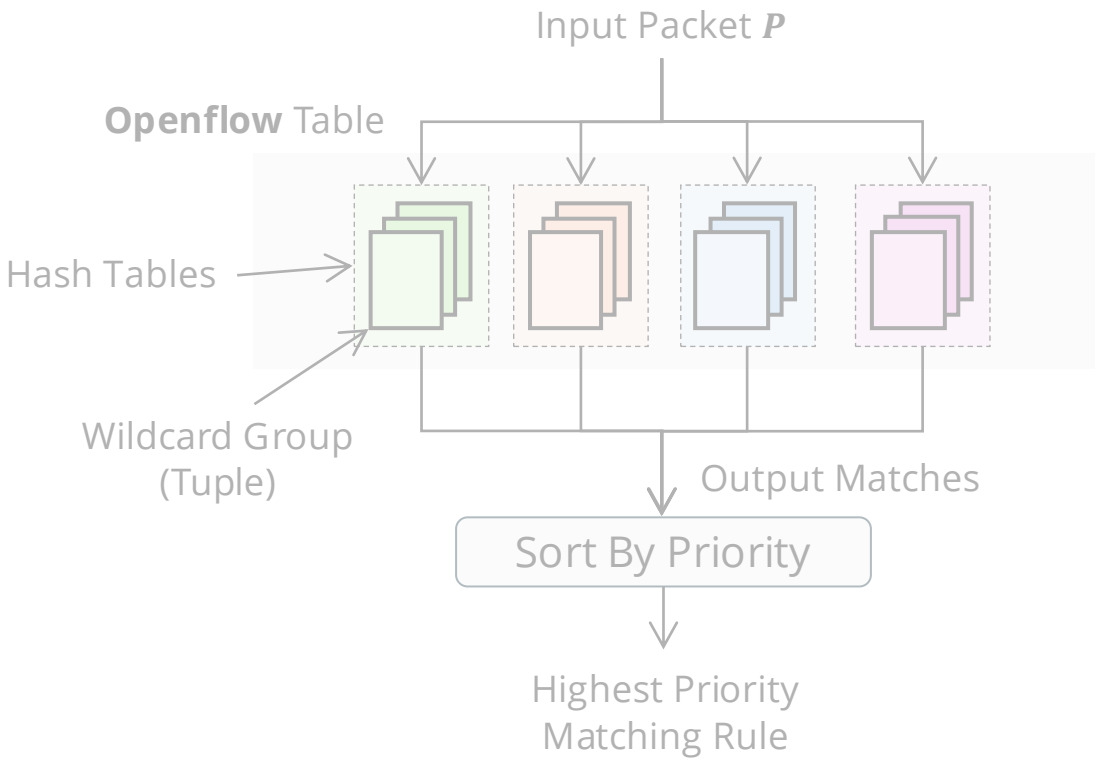


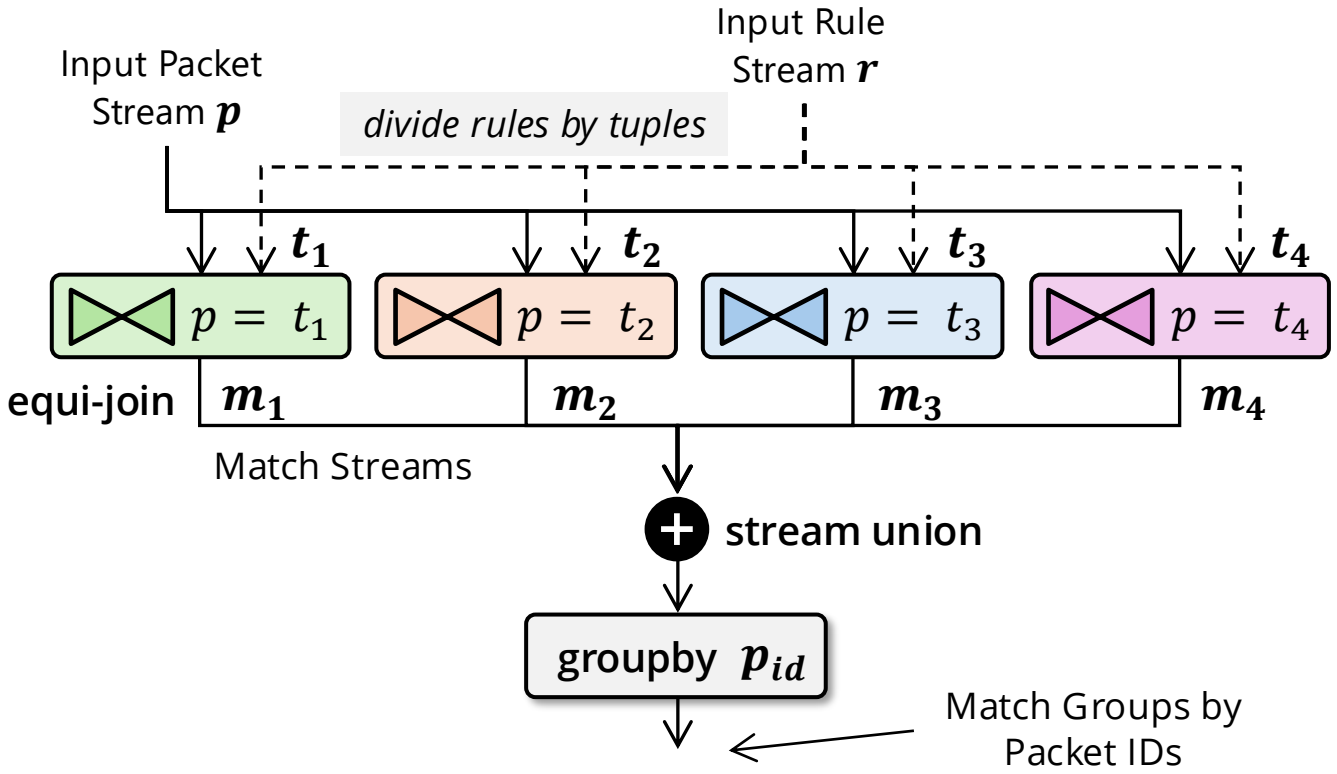
Step 3: Support Rule Priorities



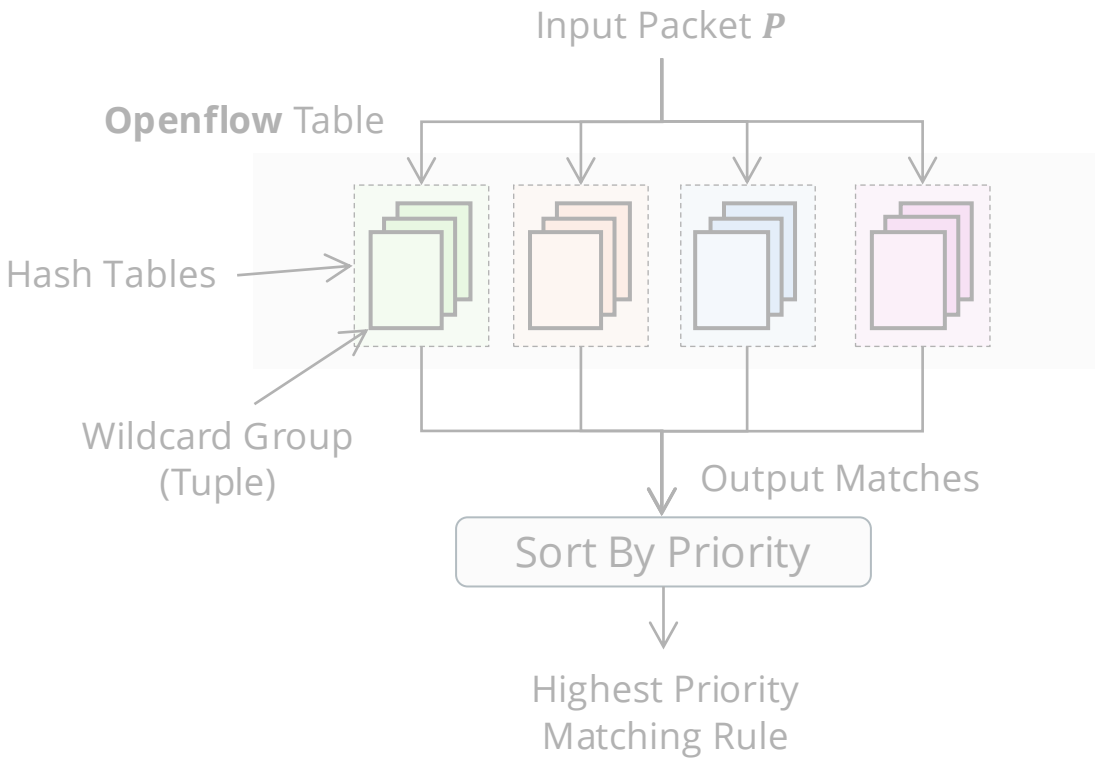


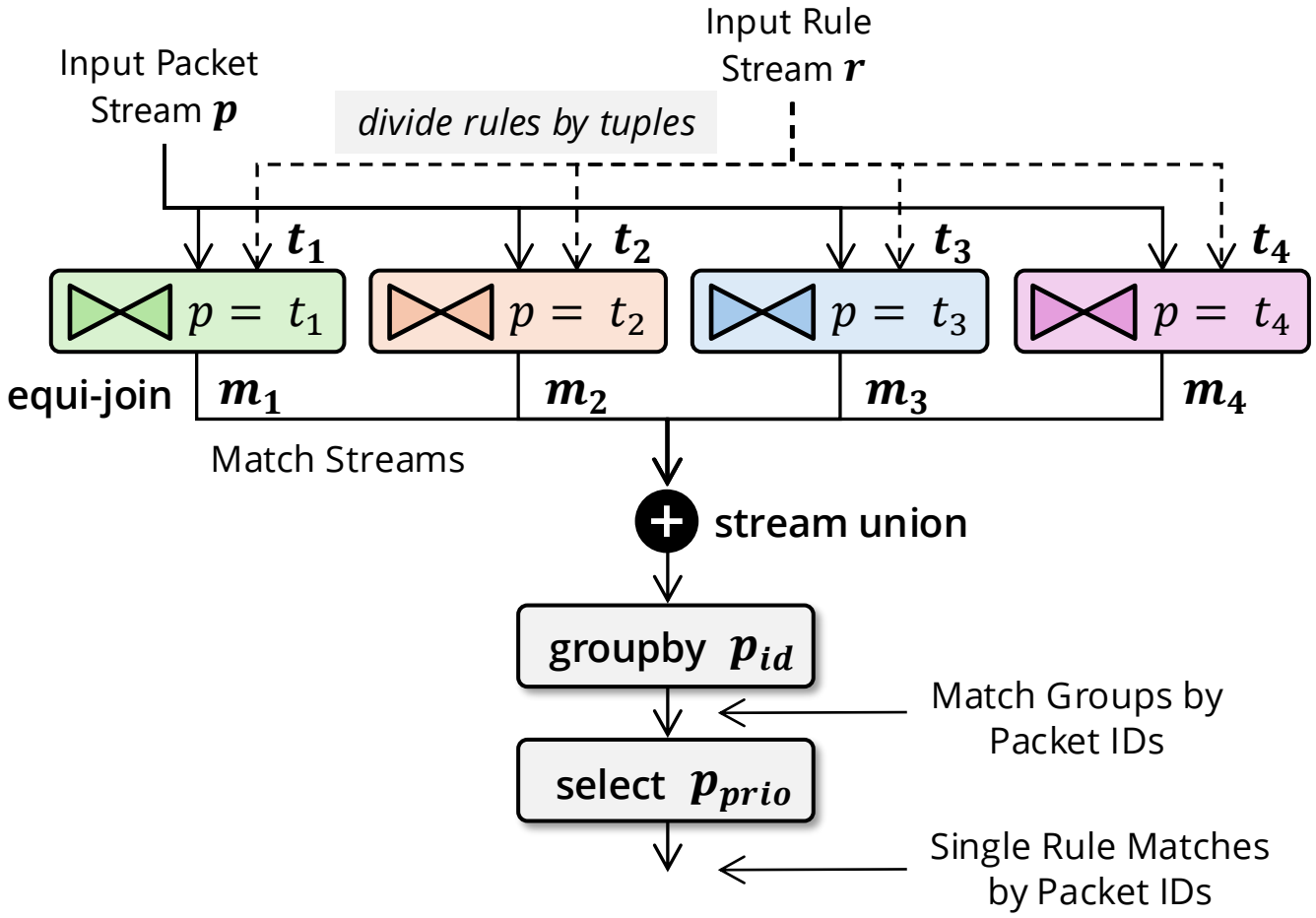
Step 3: Support Rule Priorities



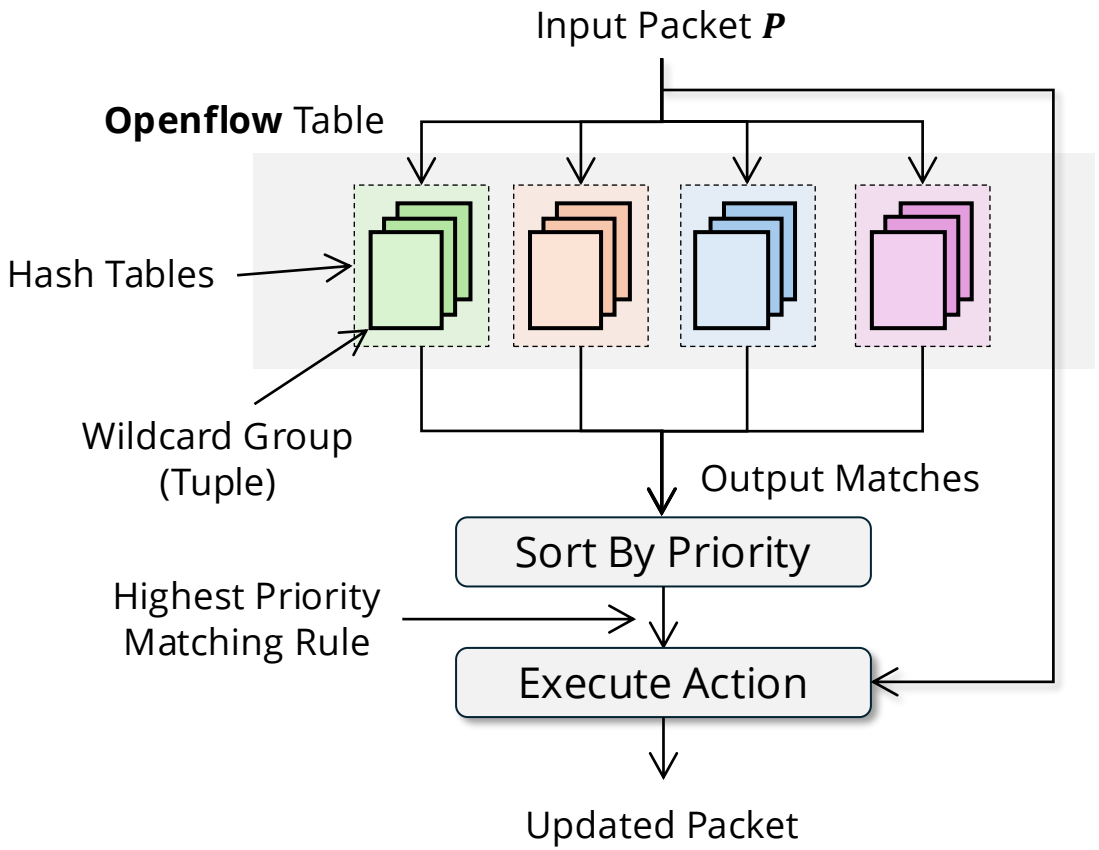


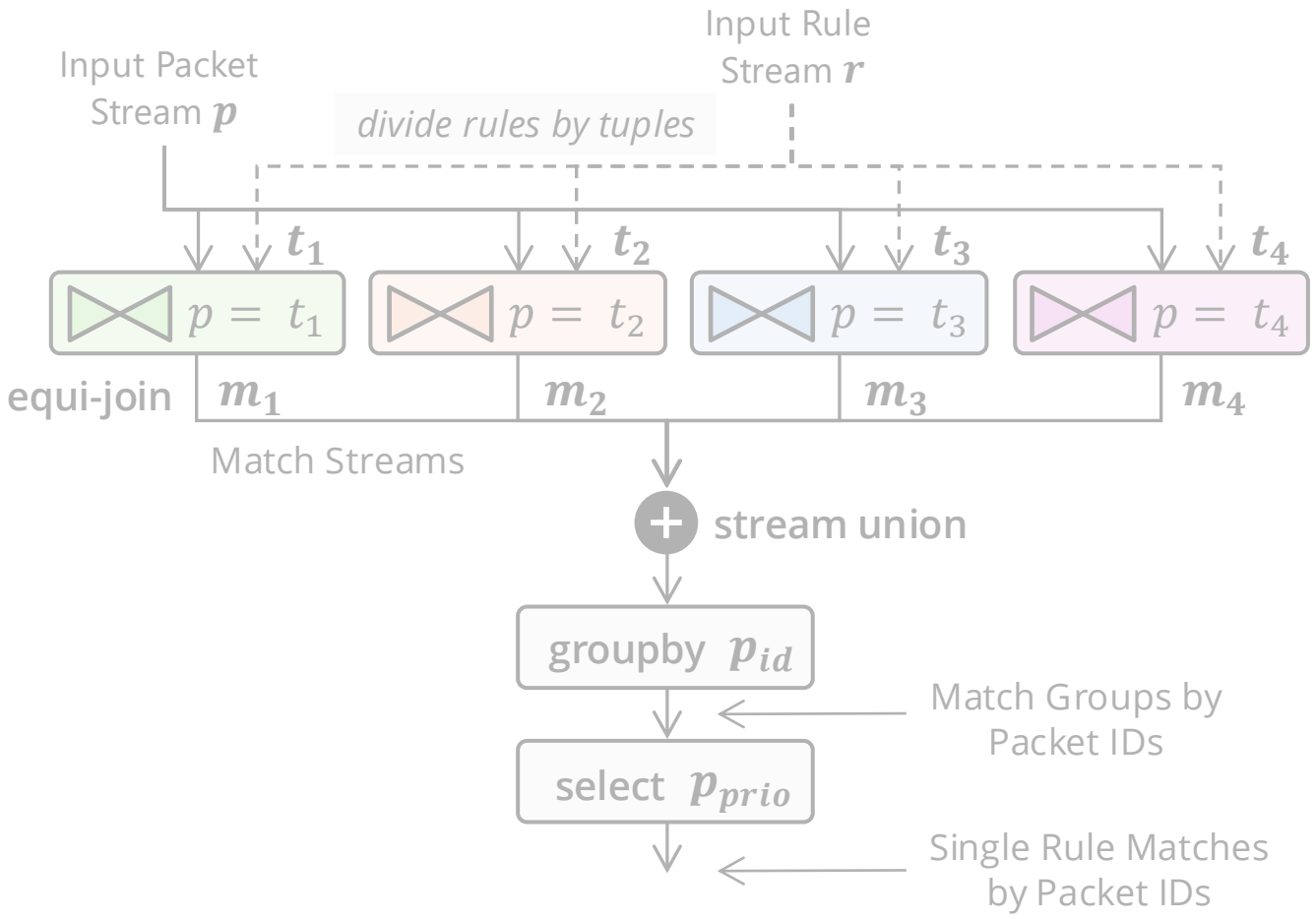
Step 3: Support Rule Priorities



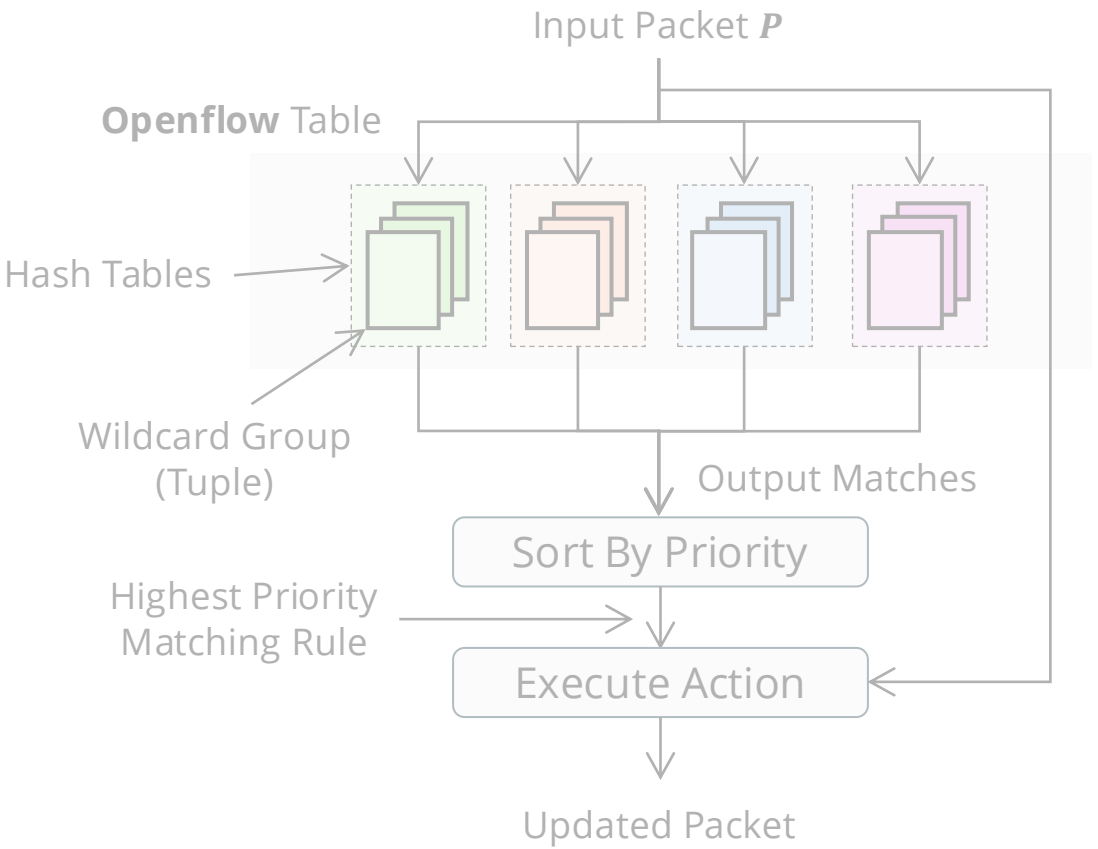


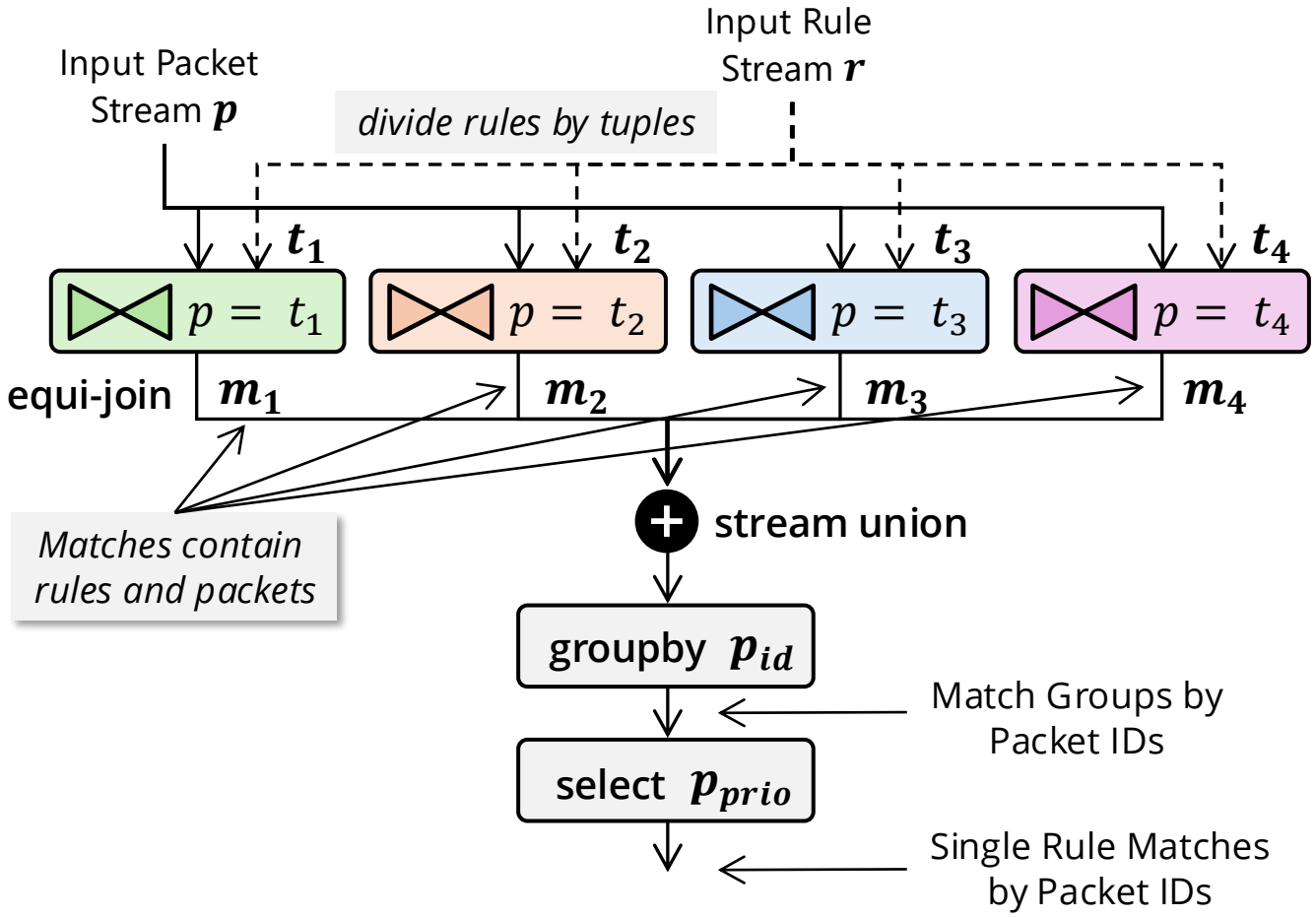
Step 4: Support Actions



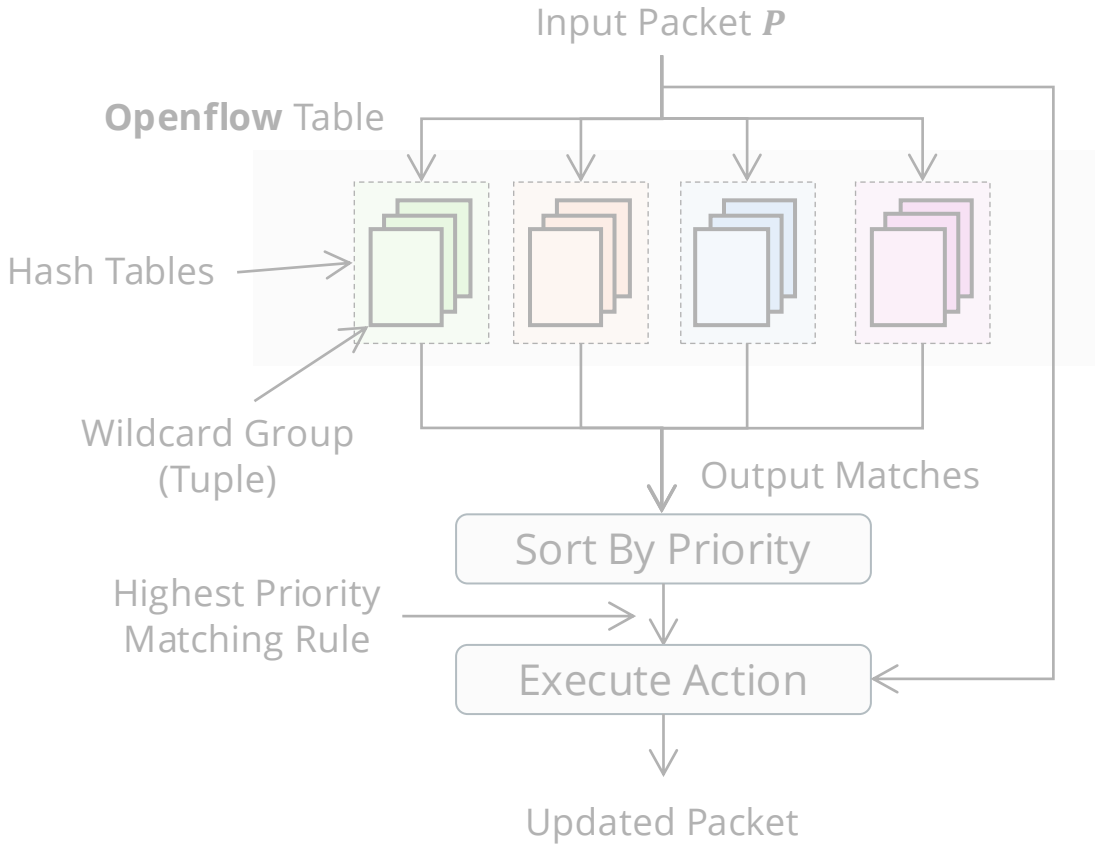


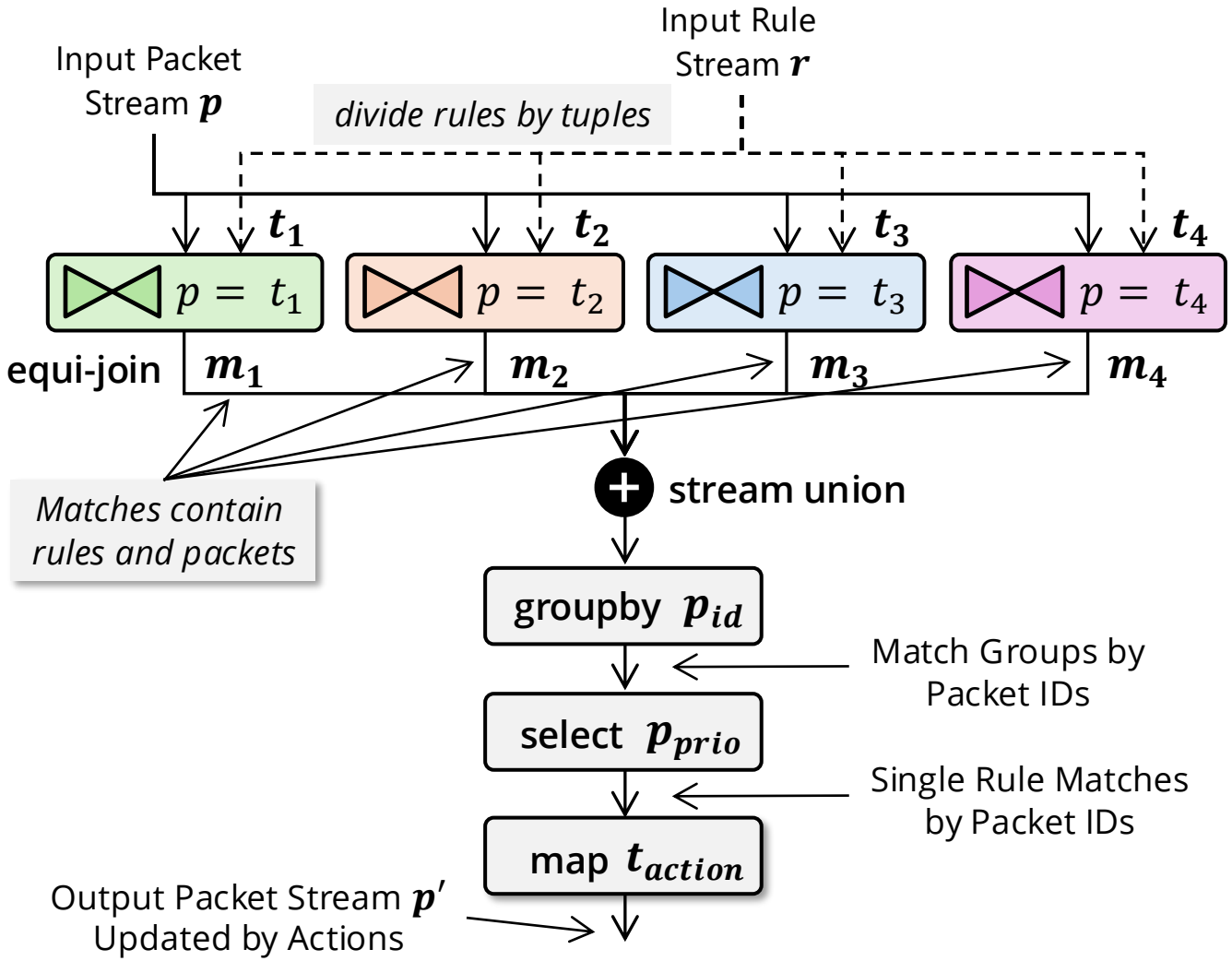
Step 4: Support Actions



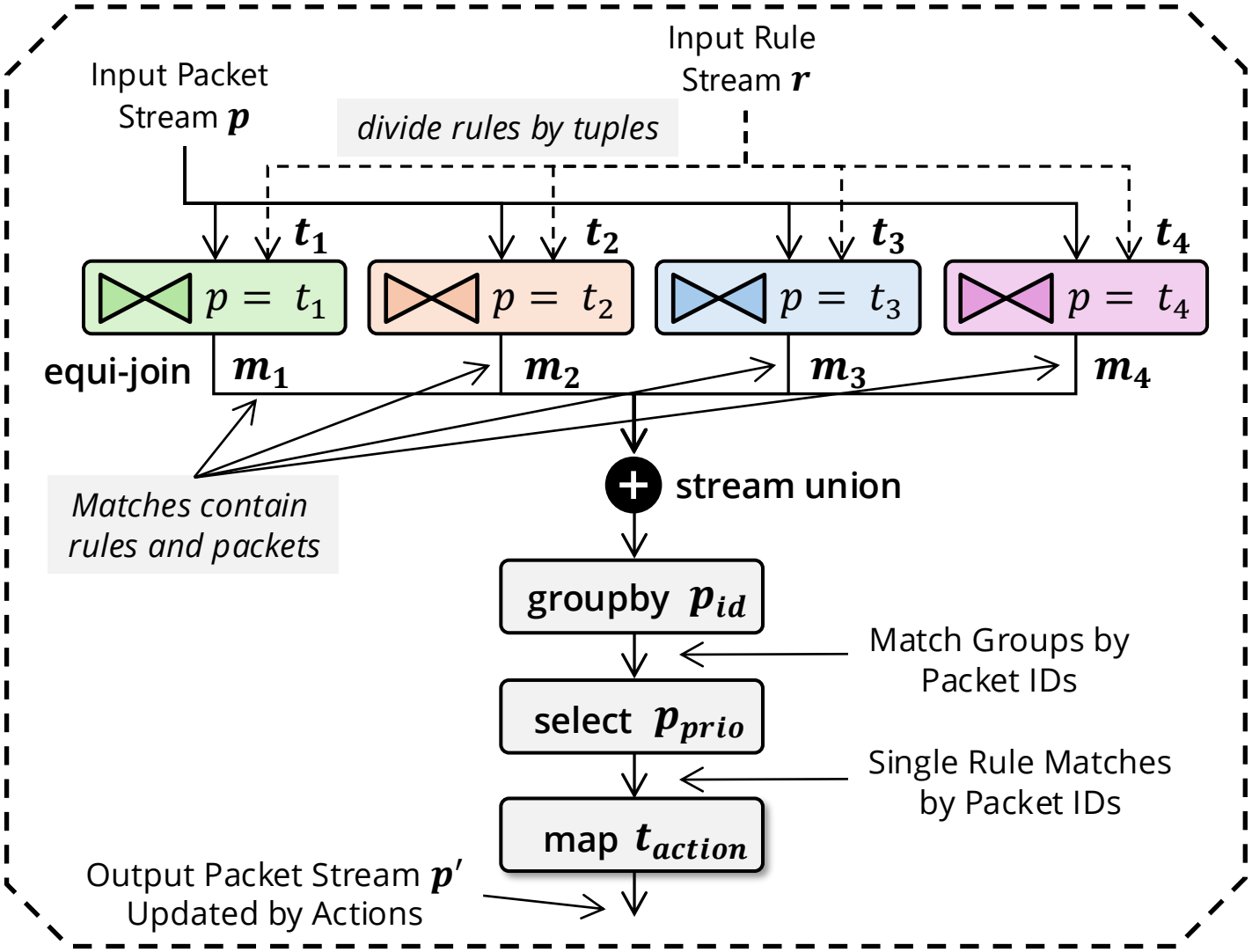


Step 4: Support Actions

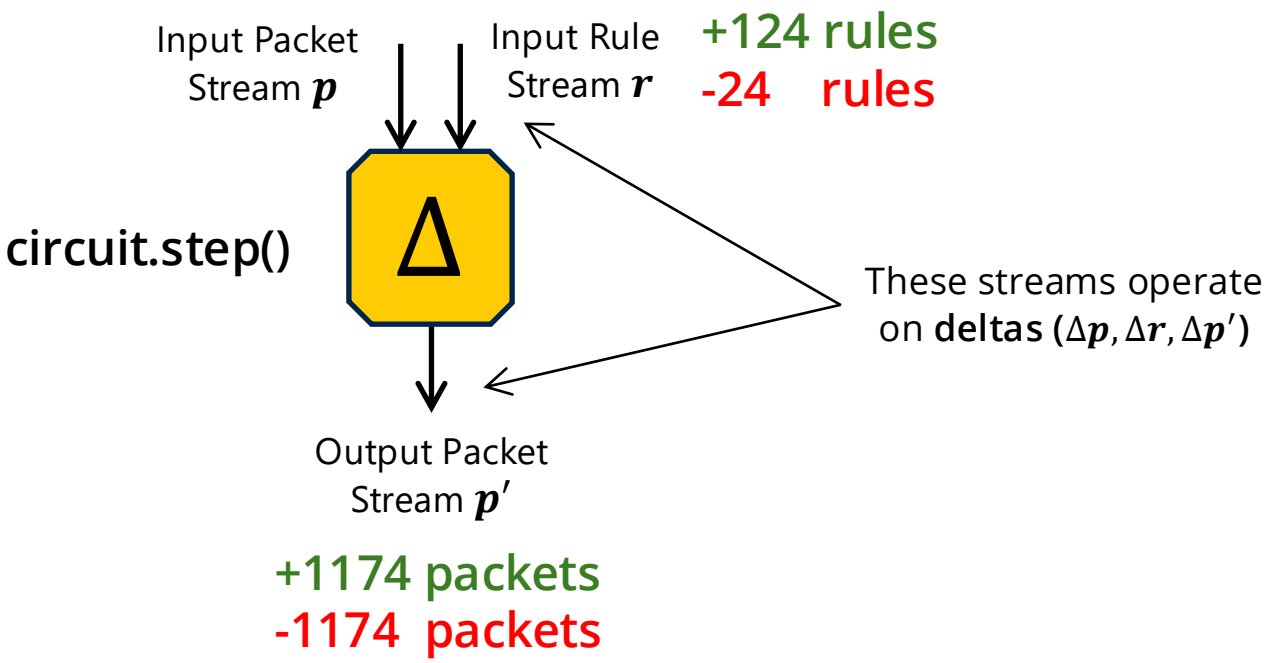




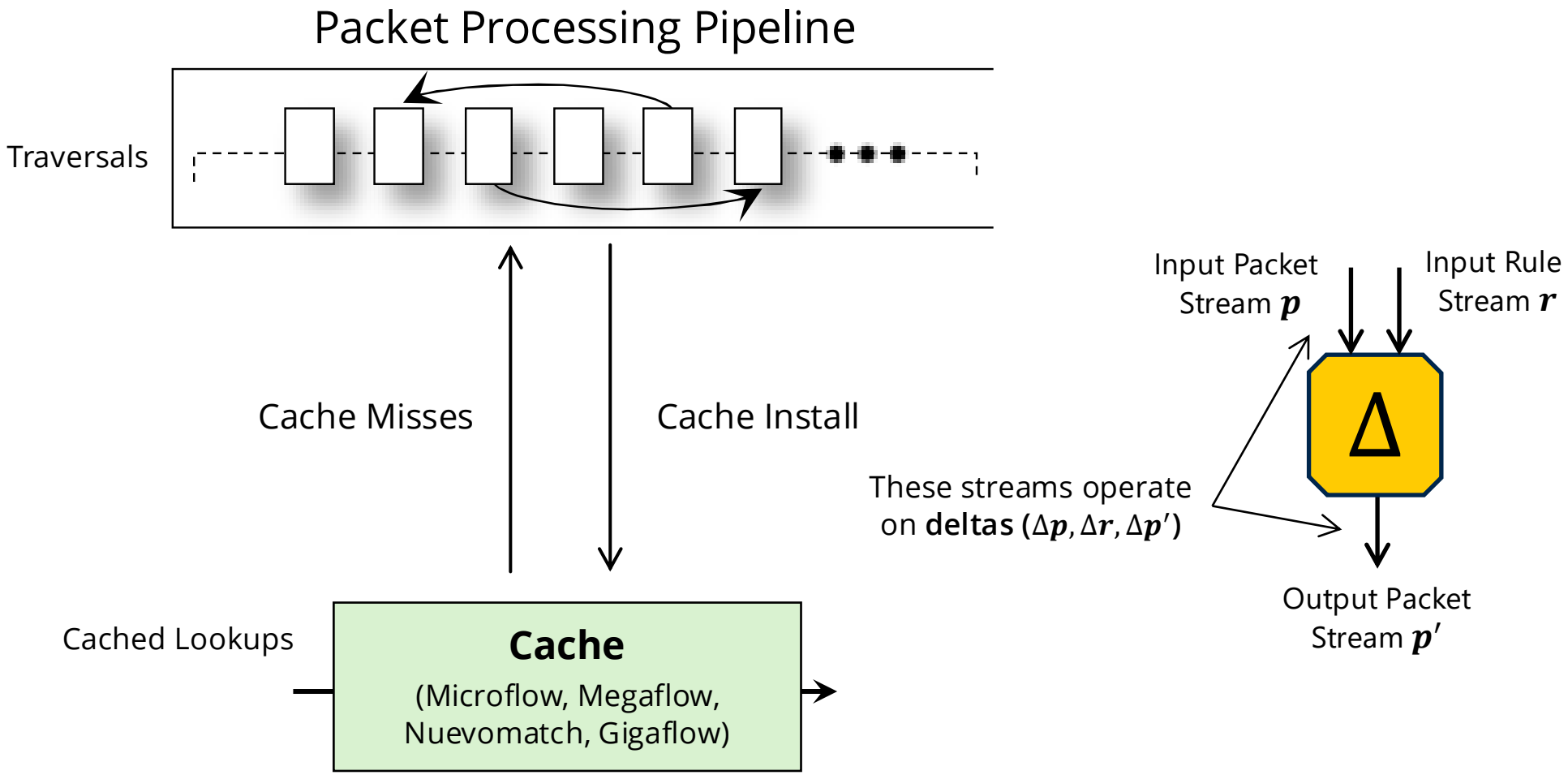
But How Do We Evict Cache Entries?



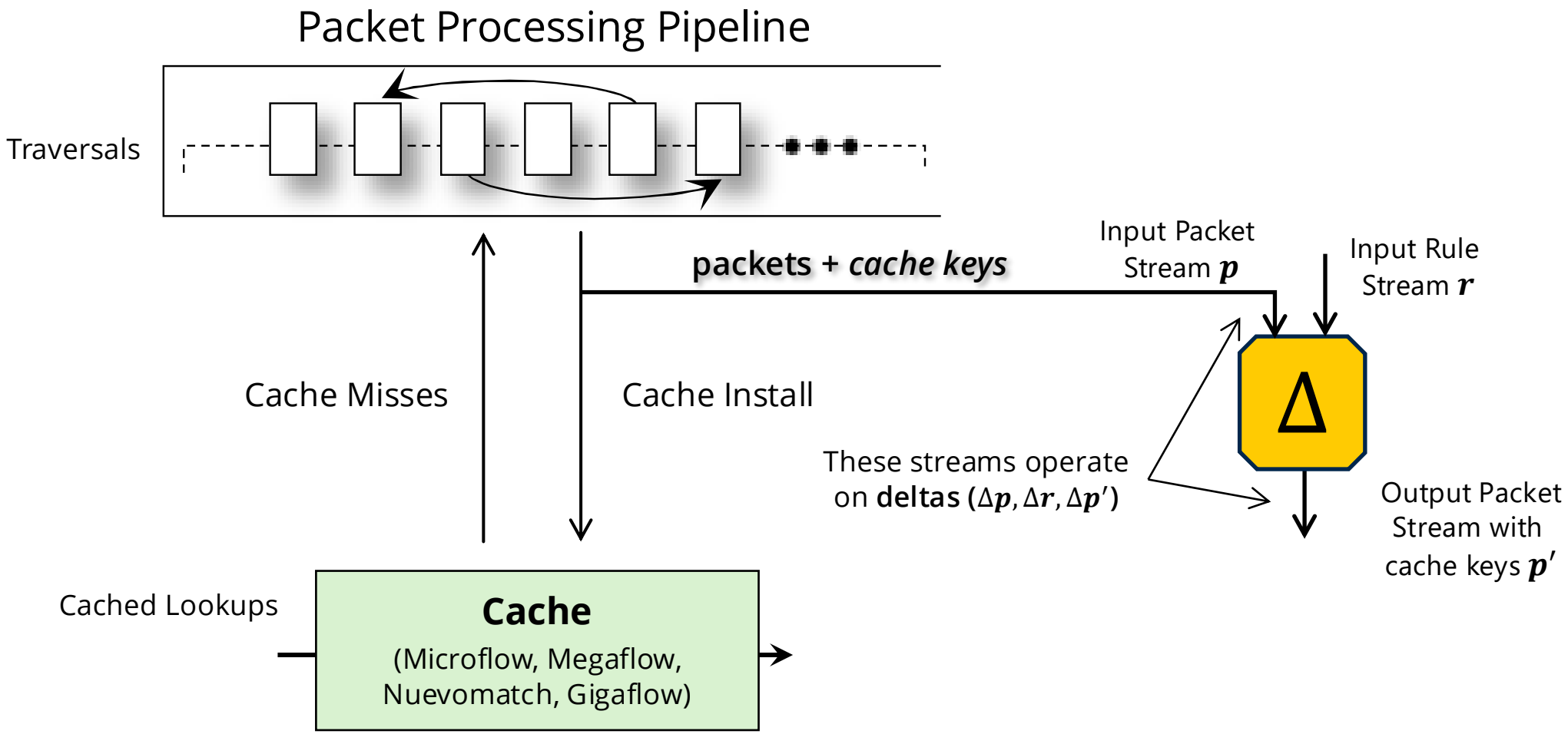
But How Do We Evict Cache Entries?



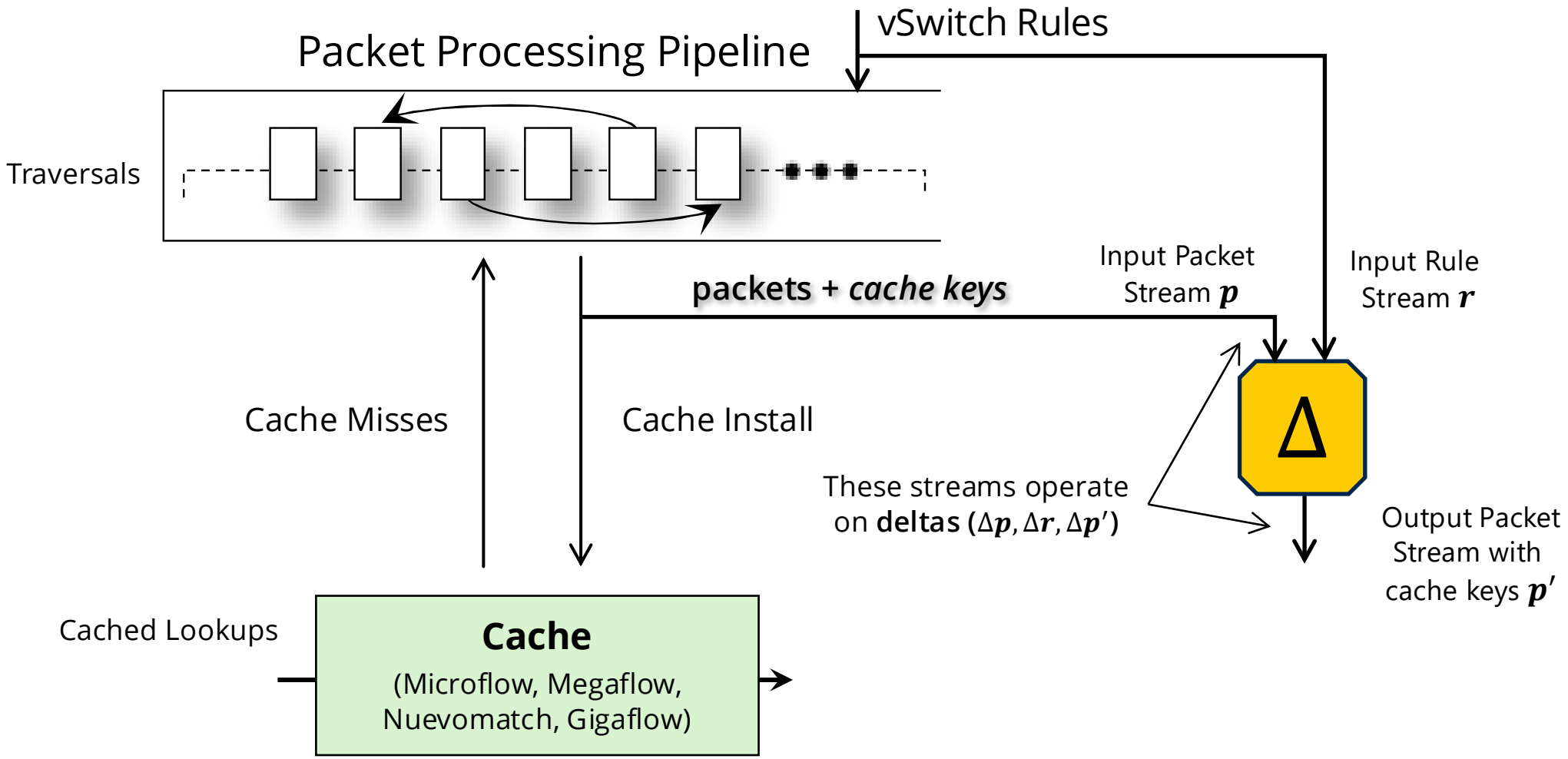
But How Do We Evict Cache Entries?



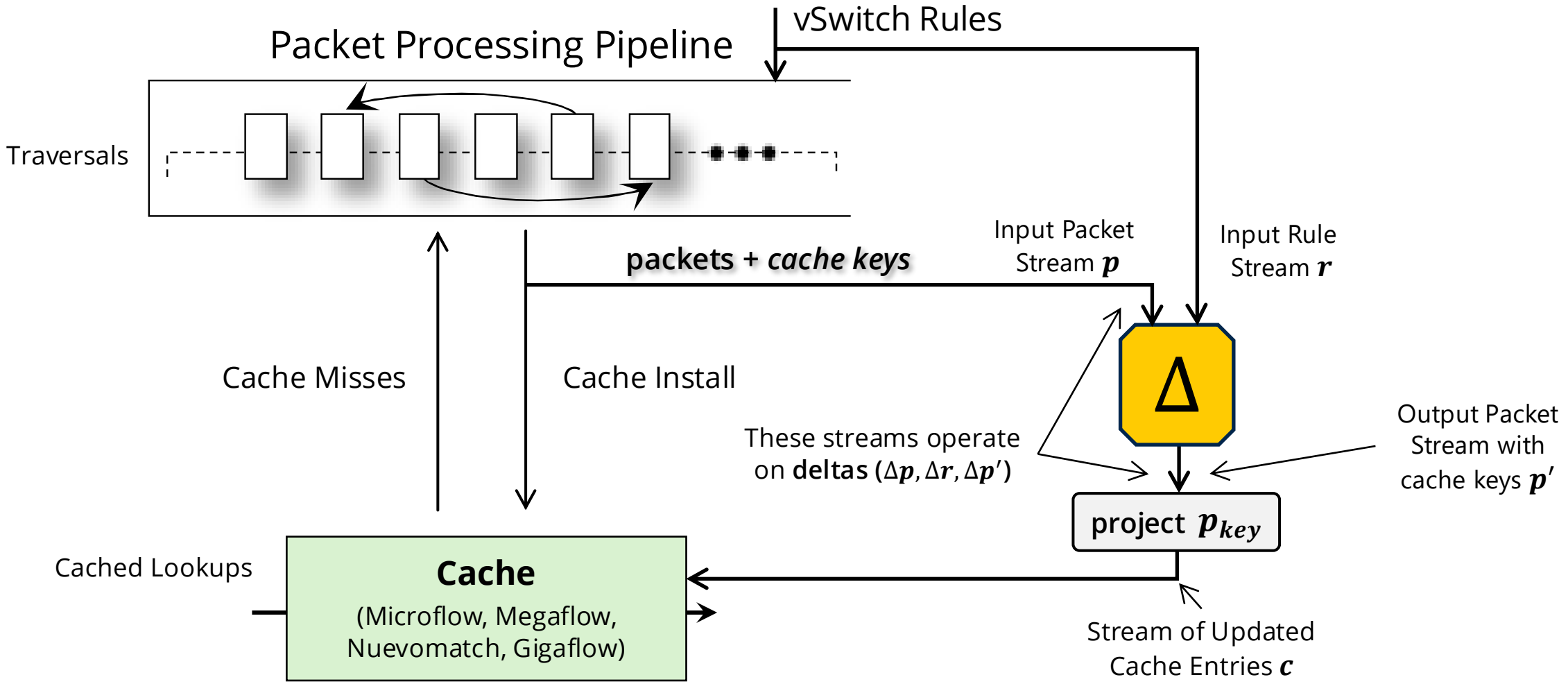
But How Do We Evict Cache Entries?



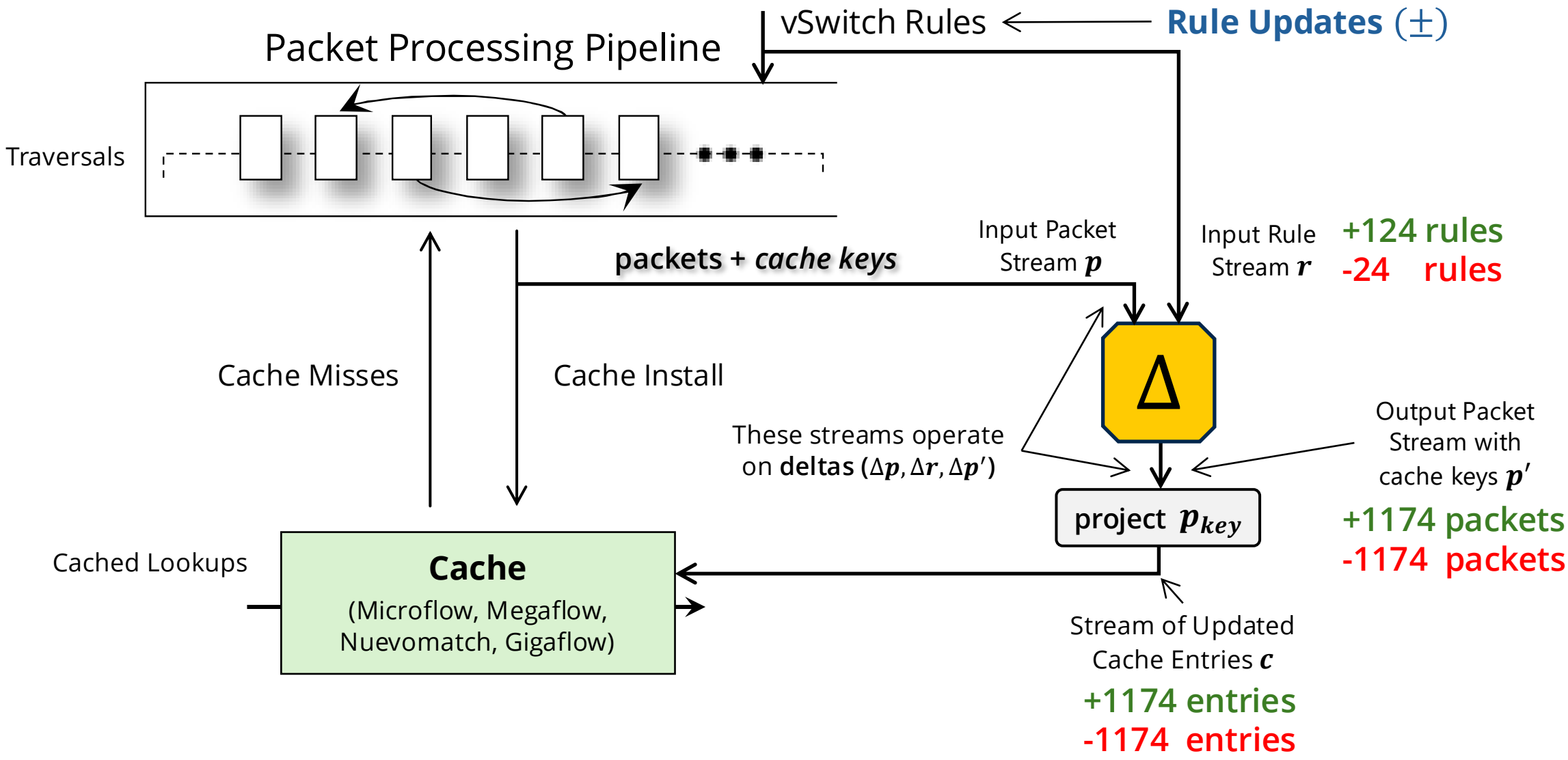
But How Do We Evict Cache Entries?



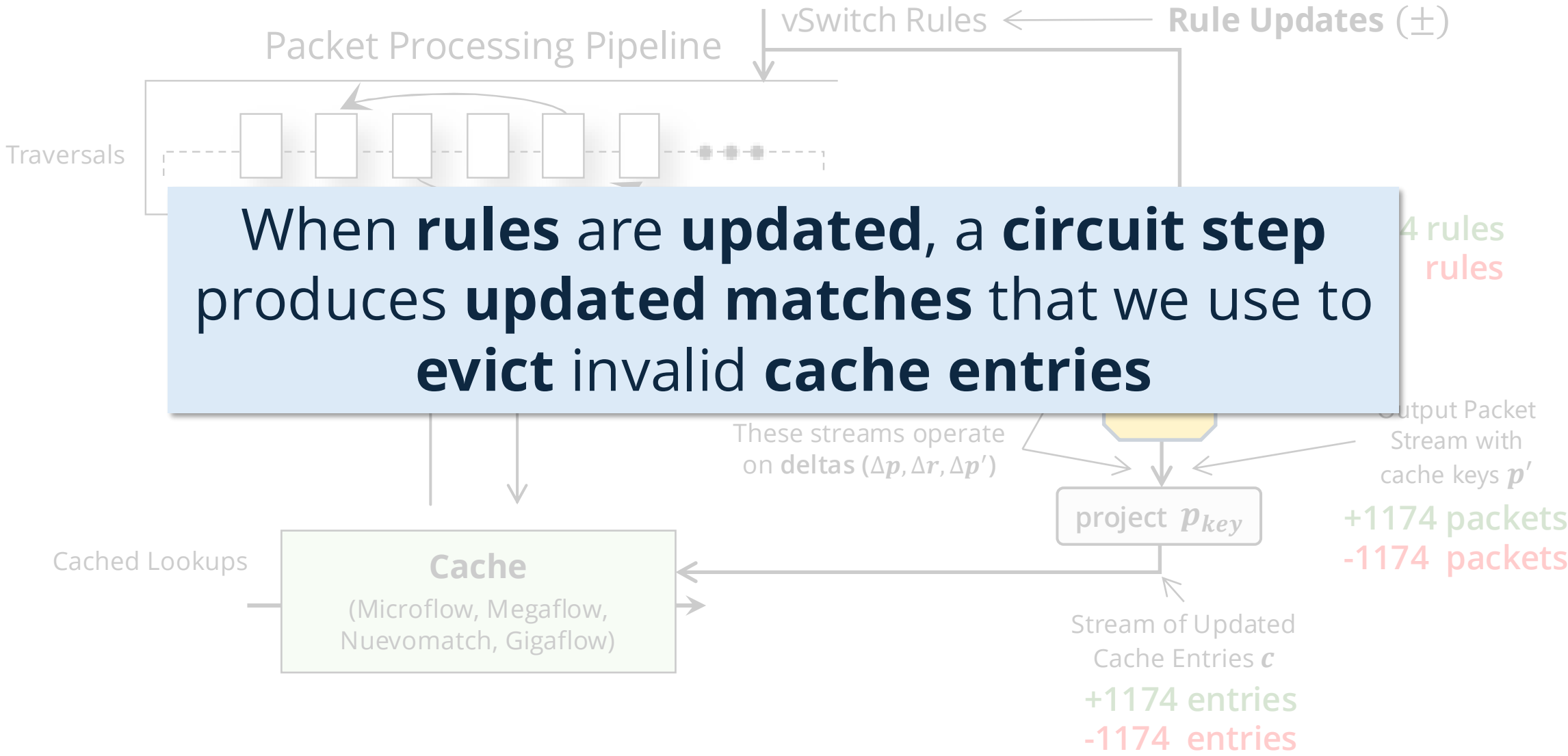
But How Do We Evict Cache Entries?



But How Do We Evict Cache Entries?

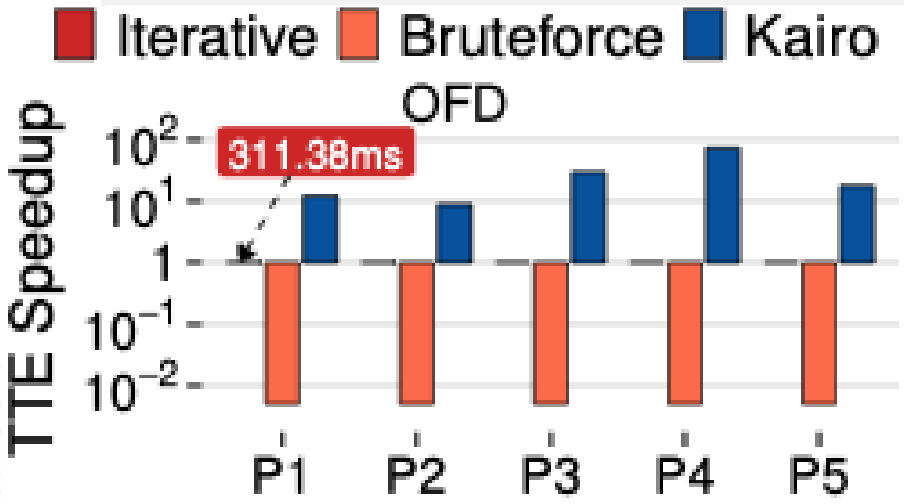


But How Do We Evict Cache Entries?



But How Do We Evict Cache Entries?

When **rules** are **updated**, a **circuit step** produces **updated matches** that we use to **evict** invalid **cache entries**



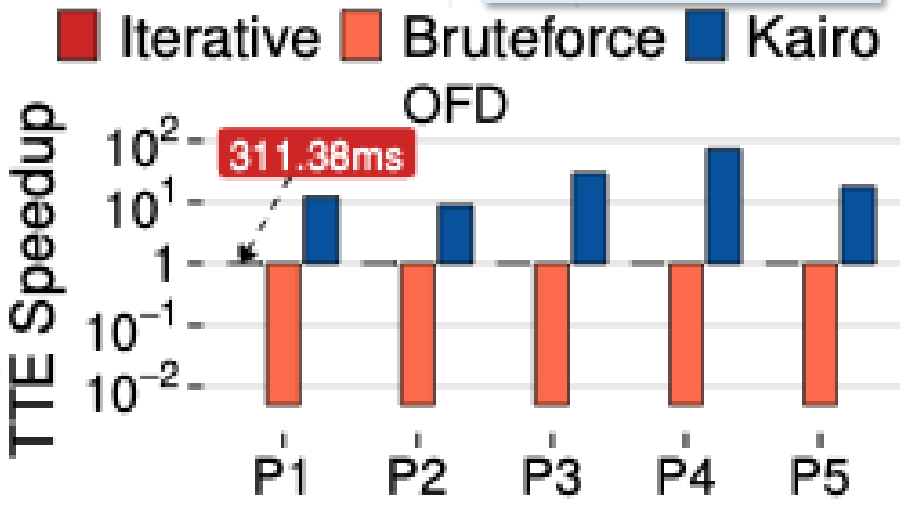
Update Patterns

16.7x faster

But How Do We Evict Cache Entries?

When **rules** are **updated**, a **circuit step** produces **updated matches** that we use to **evict** invalid **cache entries**

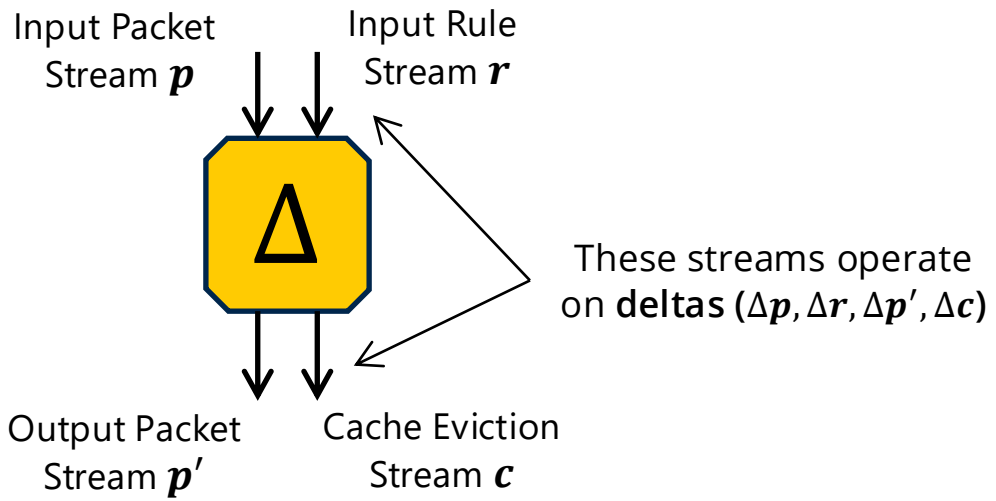
16.7x faster



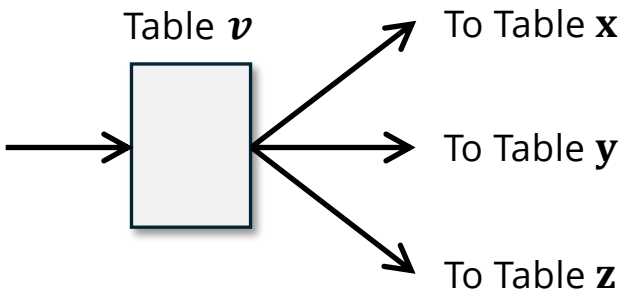
Overhead %		
Setup Time	**CPU Util.**	**Memory**
19.4%	0.27%	43.2%

Despite **additional state**, Kairo incurs **moderate resource** overheads

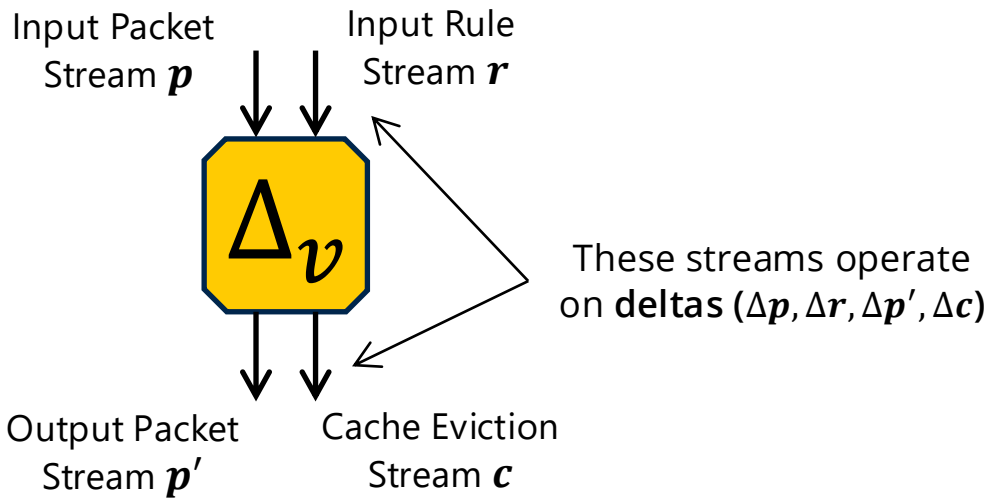
Step 5: Support Control Flows



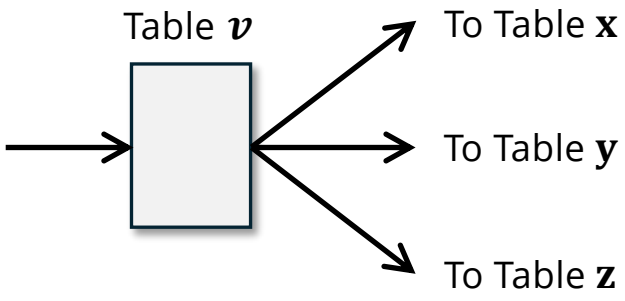
Step 5: Support Control Flows



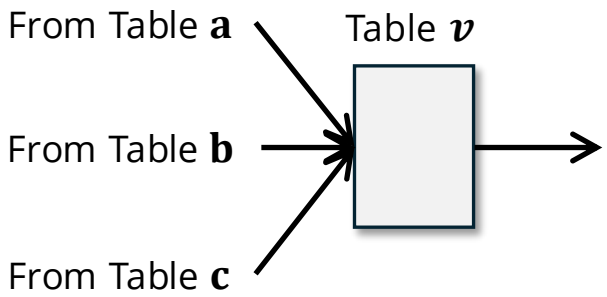
(a) Multi-Table Fan-Out



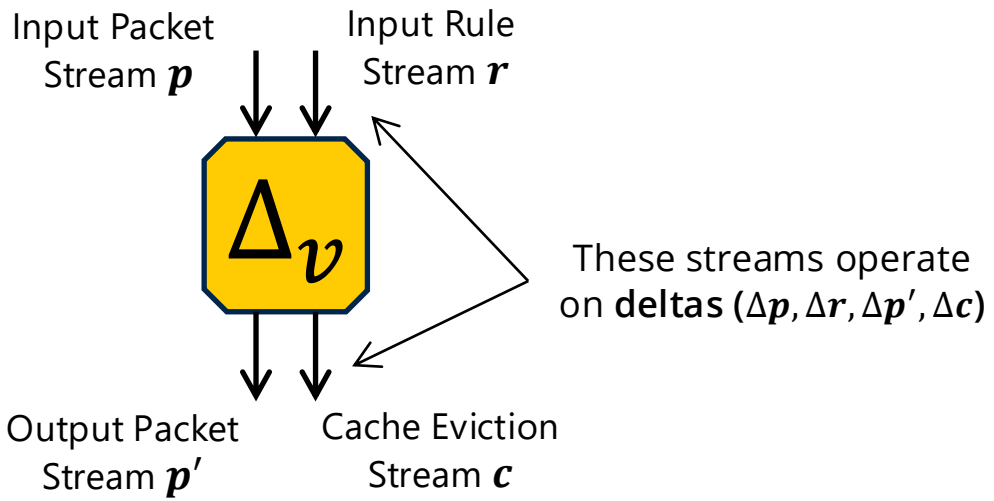
Step 5: Support Control Flows



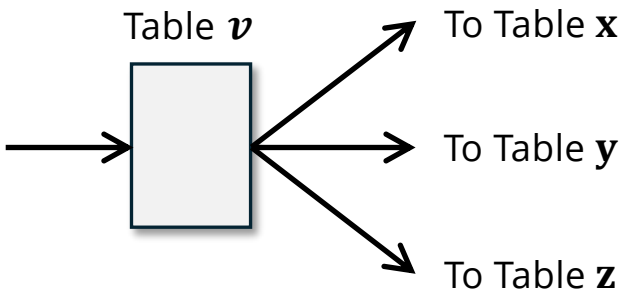
(a) Multi-Table Fan-Out



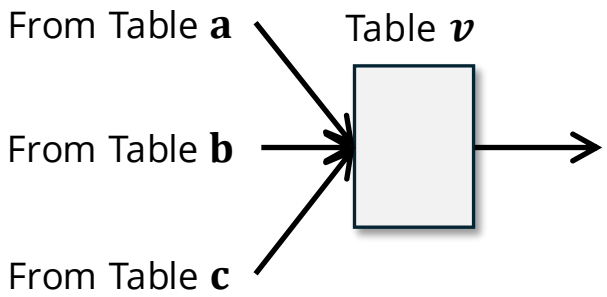
(b) Multi-Table Fan-In



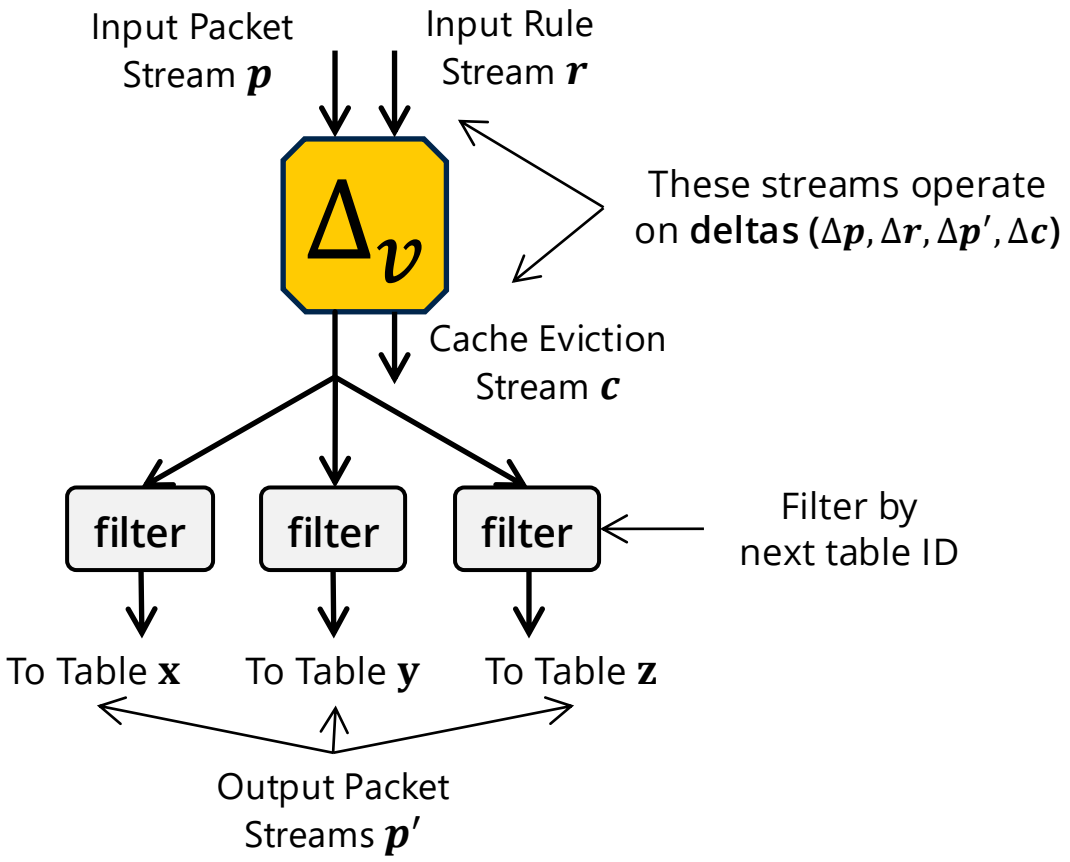
Step 5: Support Control Flows



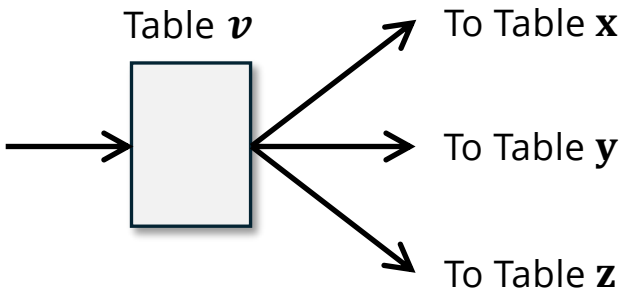
(a) Multi-Table Fan-Out



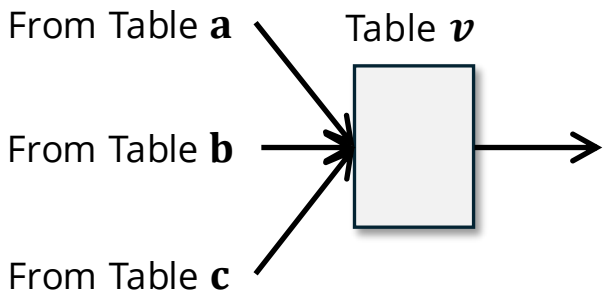
(b) Multi-Table Fan-In



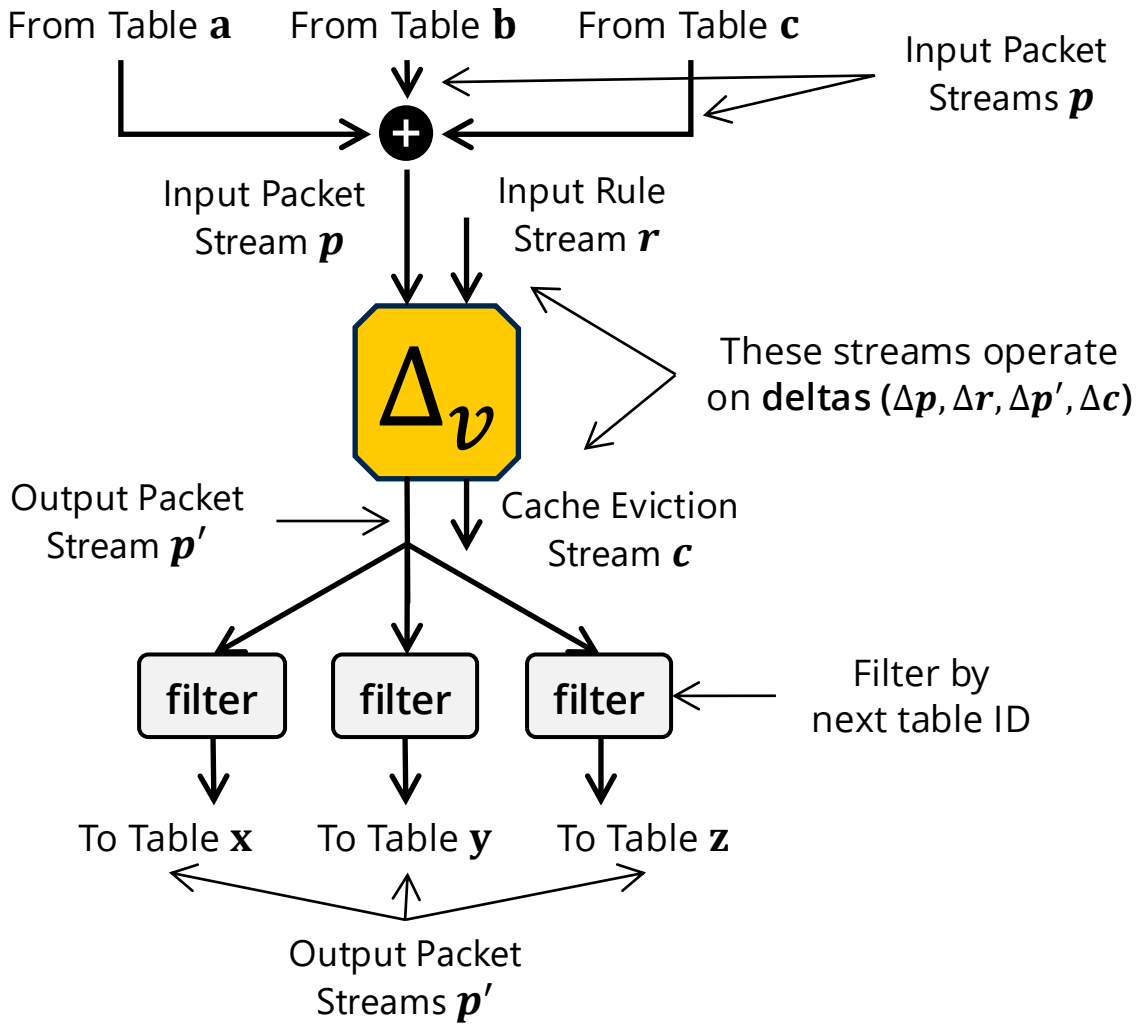
Step 5: Support Control Flows



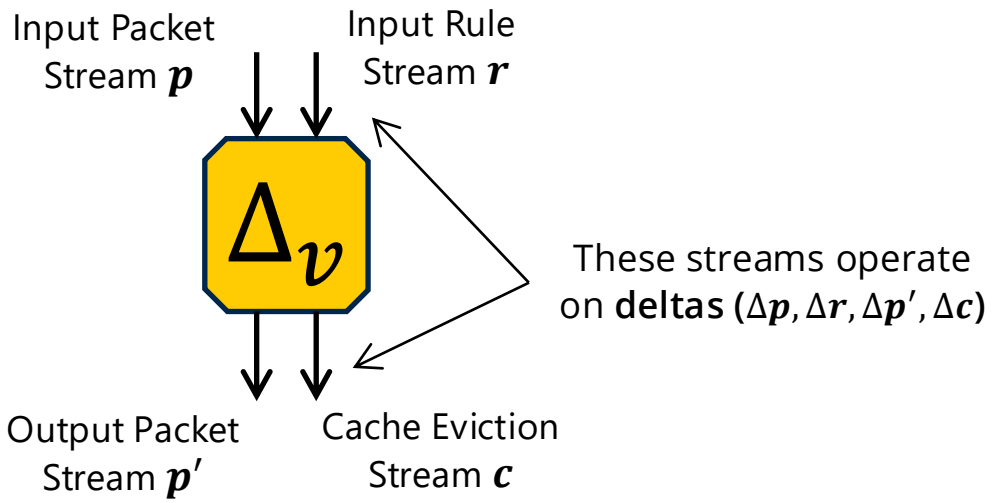
(a) Multi-Table Fan-Out



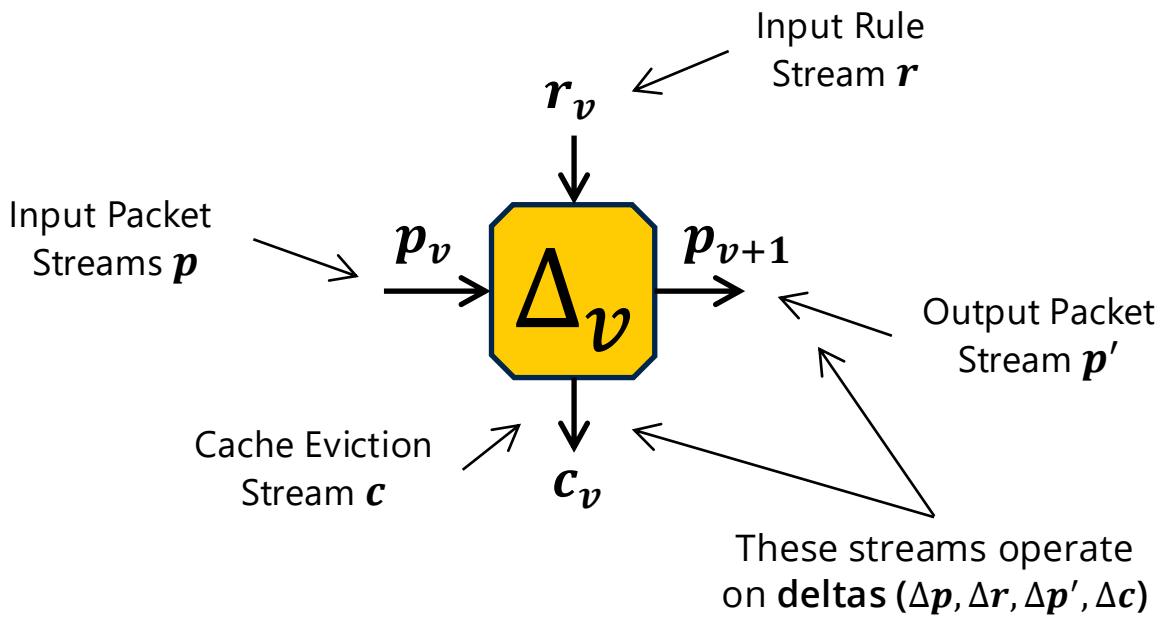
(b) Multi-Table Fan-In



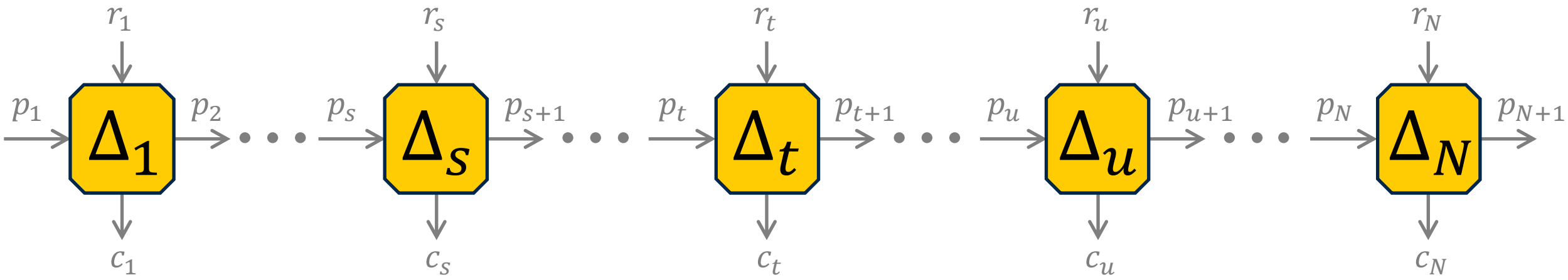
C2: Controlling Kairo Execution



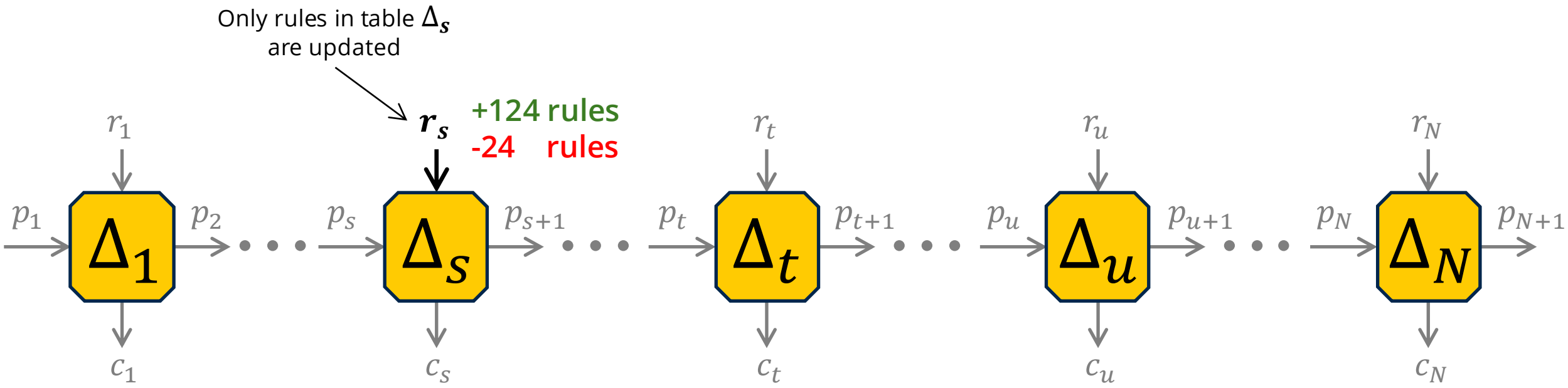
Controlling Kairo Execution



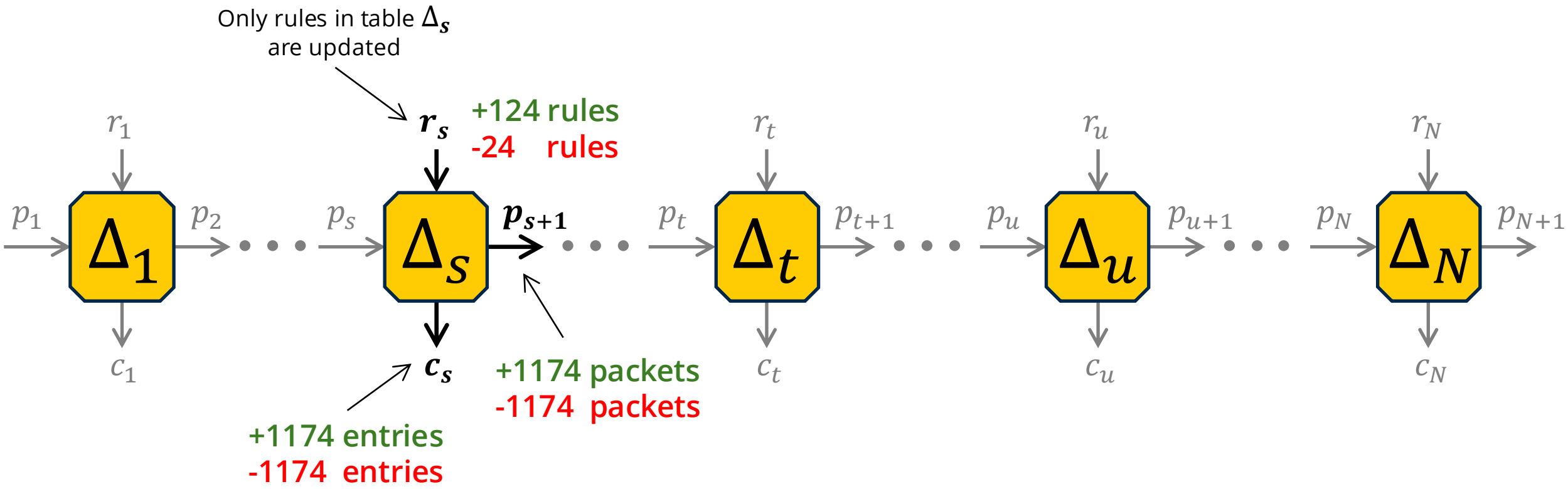
Controlling Kairo Execution



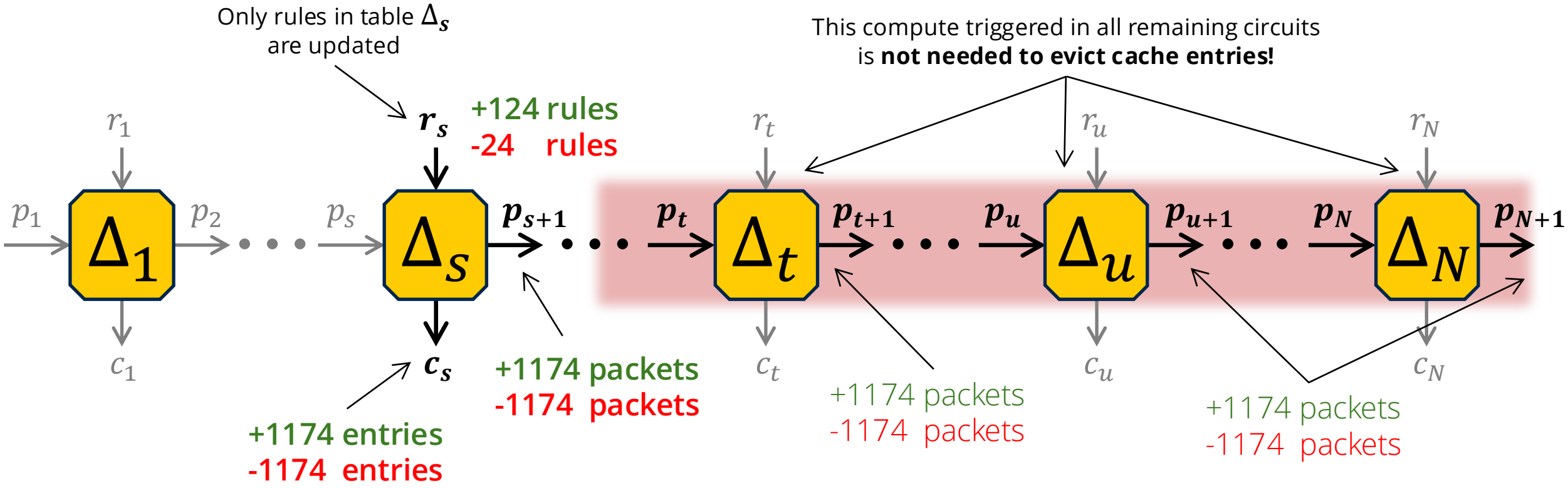
Controlling Kairo Execution



Controlling Kairo Execution



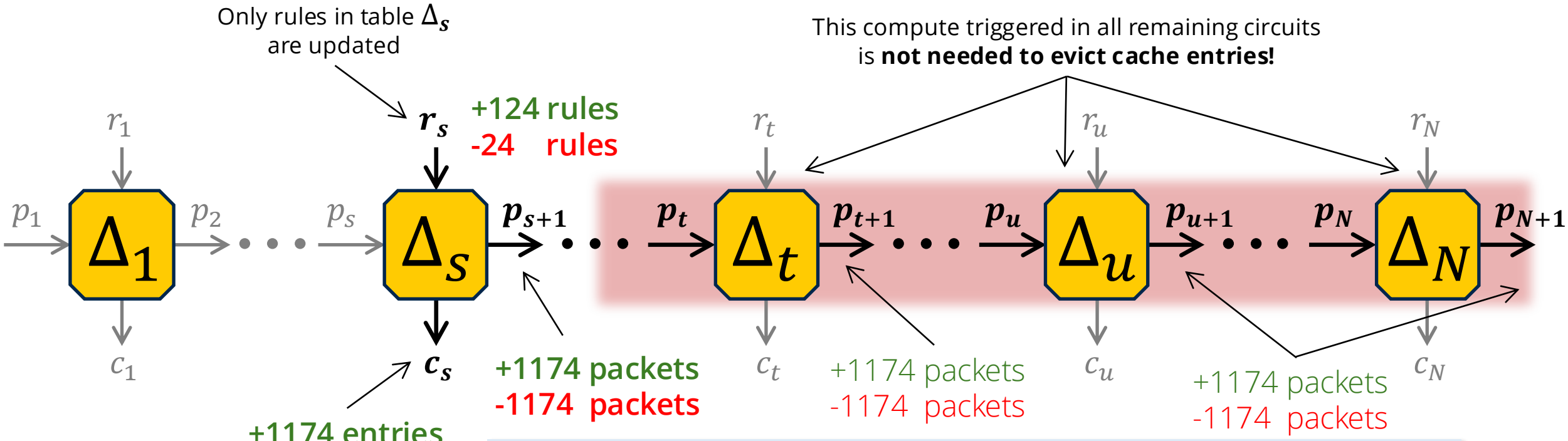
Controlling Kairo Execution



Only **rule updates** trigger **cache evictions**!

Packet updates carry only the **state update** for circuits

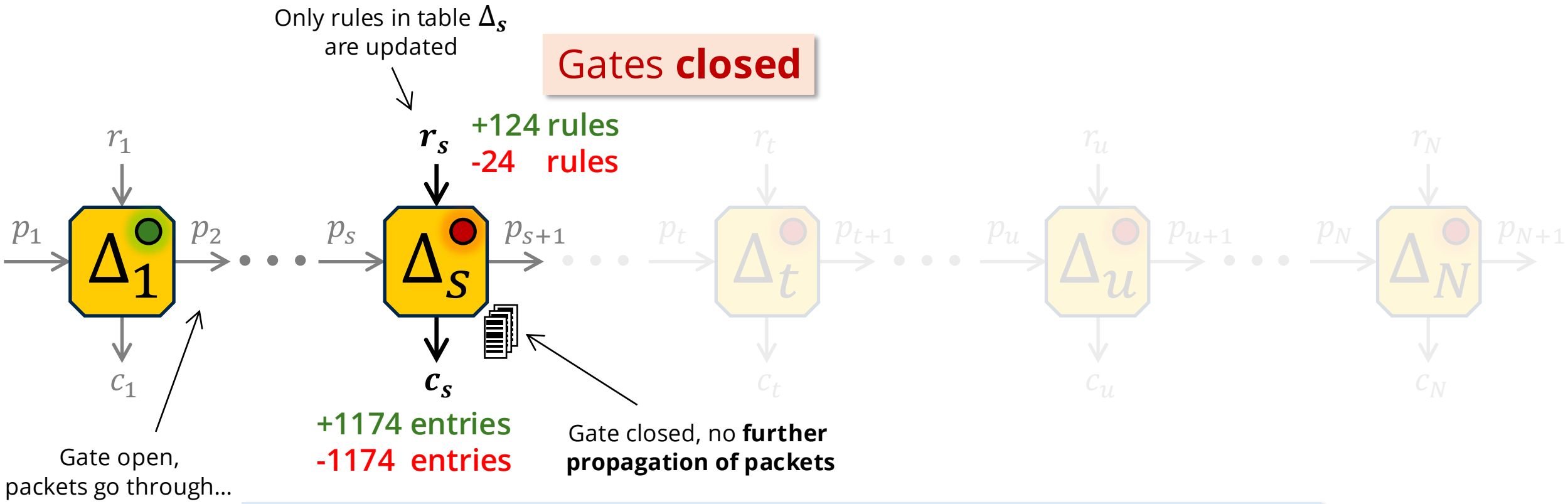
Controlling Kairo Execution



Compute beyond the **last updated table** can be **deferred** to **evict stale cache immediately!**

Need a way to **retain intermediate packet state**

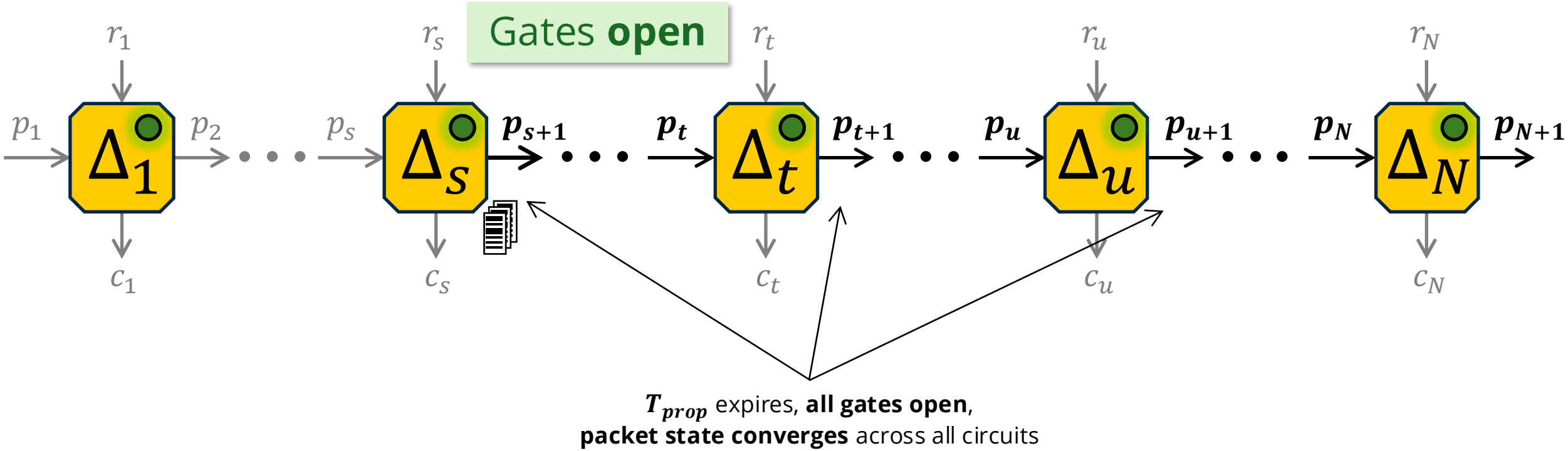
Controlling Kairo Execution with **Gates** and T_{prop}



Lightweight **gate operators** are **embedded in circuits** to **evict cache entries** and **retain packet state**

Controlling Kairo Execution with **Gates** and T_{prop}

A configurable **timer T_{prop}** triggers the **release of retained packet state** to allow **state convergence** in **idle periods**



Rule Update Patterns, Pipelines, and Traffic

Pattern	Description	Reported Updates (#/%)
P1: Security	Firewall/ACL periodic rule changes, security responses (e.g., to DDoS)	1005 / **6.0%**
P2: Pruning	Redundant/under-utilized ruleset cleanup from enterprise firewalls	1668 / **10.0%**
P3: Recovery	Traffic engineering, fault recovery (link/node failures)	**300** / 2.1%
P4: Elasticity	Server load balancing, auto-scaling backends/pool membership changes	**110** / 0.8%
P5: Telemetry	Traffic monitoring by ISPs, temporary rules for debug/exceptions	**500** / 3.5%

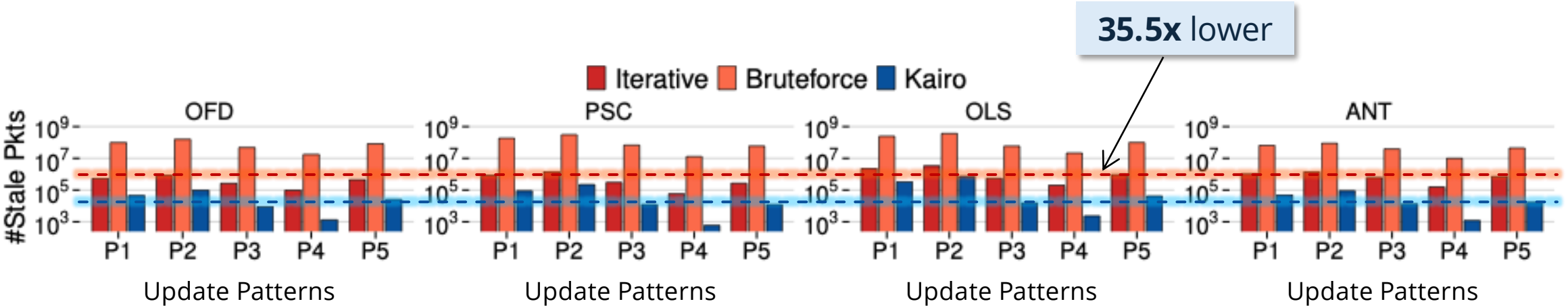
- Gigaflow Pipelines and Traffic
  - Real-world pipeline configurations
- Update patterns based on reported numbers
  - Either as rule set % or number of rules

Performance Summary of Kairo

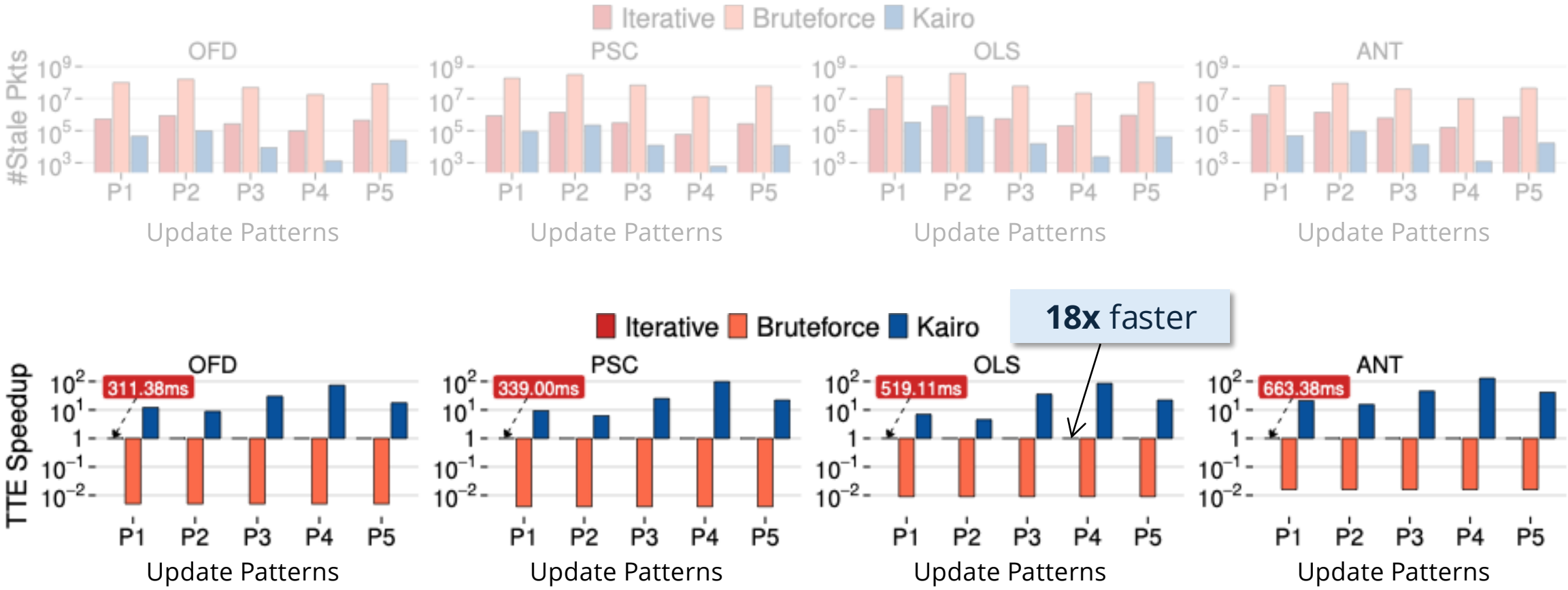
Kairo in Open vSwitch:

- reduces **time-to-eviction (TTE)** by **18x** (max **30x**)
- reduces **stale cache processing** by **35.5x** (max **130x**)
- supports **~7x** more **cache entries** at 400Gbps
- has **no impact** on **end-to-end** performance

Stale Cache Processing with Kairo

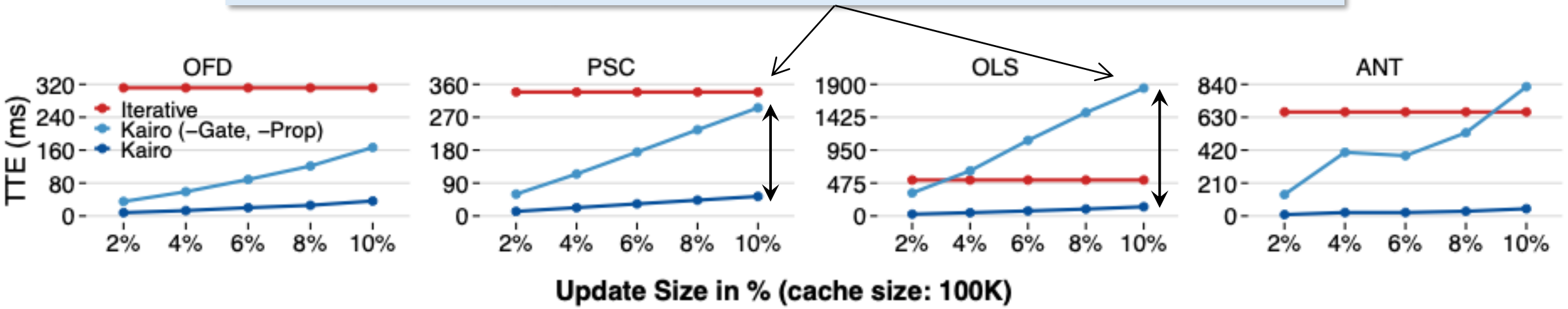


Speedup in Time-to-Eviction (TTE)



Time-to-Eviction (TTE) vs Update Size

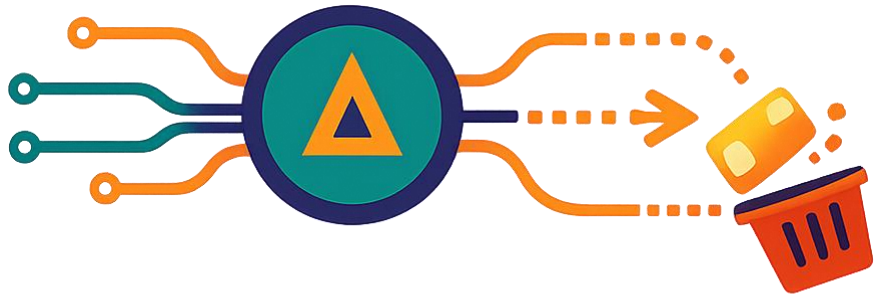
Gates and T_{Prop} help tremendously at **bigger updates**



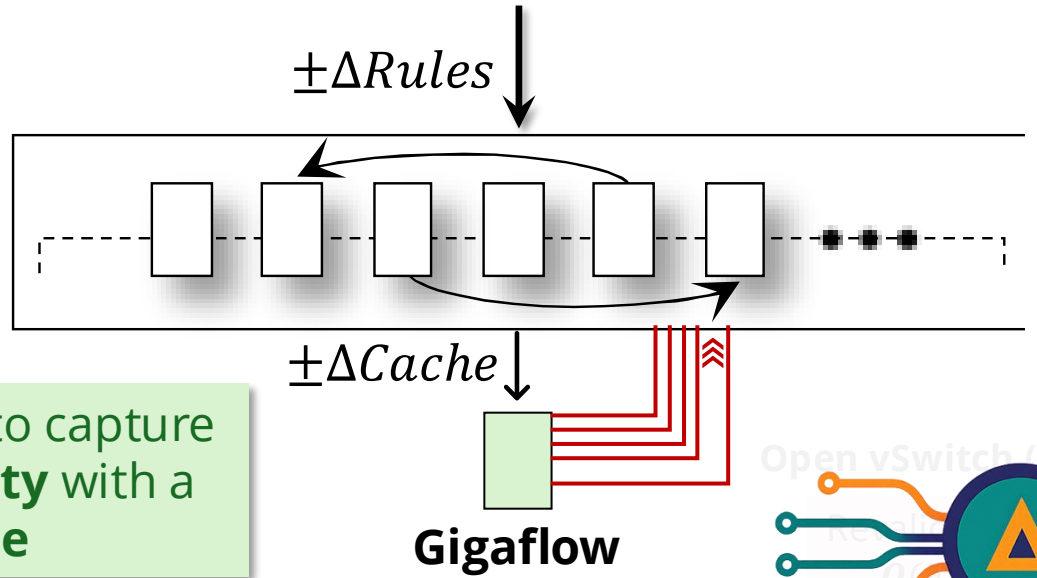
Kairo's TTE scales with the size of the update!

Kairo Summary

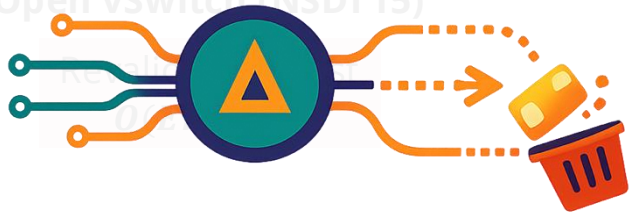
- Uses **Incremental View Maintenance**
- Evicts stale cache entries **incrementally**
- Scales with **update size**, not cache size



How to Rearchitect the vSwitch for Tbps Era?



Cache **sub-traversals** to capture **pipeline-aware locality** with a **Gigaflow cache**

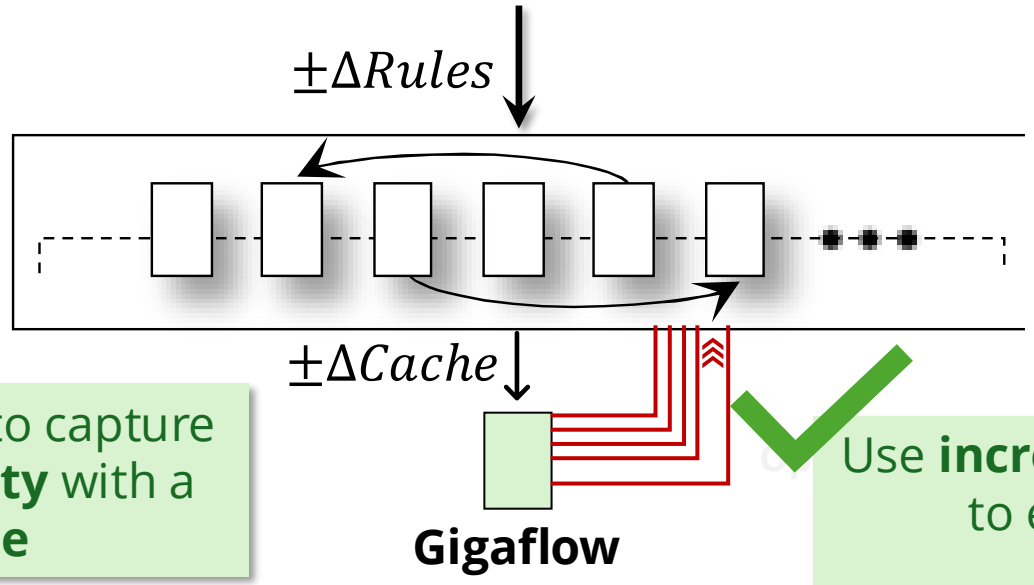


Kairo Cache Eviction

Q2: How to quickly **evict stale cache** when **vSwitch rules are updated?**

GvS
GIGAFLOW
VIRTUAL SWITCH

How to Rearchitect the vSwitch for Tbps Era?



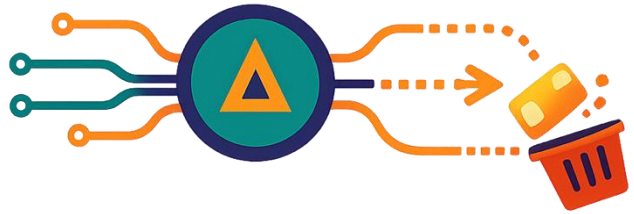
Cache **sub-traversals** to capture **pipeline-aware locality** with a **Gigaflow cache**

Use **incremental view maintenance** to evict stale cache entries in $\boldsymbol{O(\Delta R + \Delta E)}$ **time**

NuevomatchUP (NSDI'22) Elixir (NSDI'22)

"Do packet classificatio… offload subset of Megaflow to on dozens of CPUs for … SmartNIC TCAM memory"

GvS
GIGAFLOW
VIRTUAL SWITCH



Kairo Cache Eviction

List of Publications

Towards Network-Efficient Cross-Regional Inference via Learned Activation Compression
↳ **APNet 2026** (Under Review) *Co-author*

Incremental Cache Eviction for Modern vSwitches
↳ **SIGCOMM 2026** (Under Review) *First author*

Celeris: A Resilient and Tail-Optimal RDMA NIC for Distributed ML Workloads
↳ **OSDI 2026** (Under Review) *Co-author*

SpliDT: Partitioned Decision Trees for Scalable Stateful Inference at Line Rate
↳ **NSDI 2026** *Co-first author*

Reimagining RDMA Through the Lens of ML
↳ **CAL 2025** *Co-author* ***Broadcom Research Award***

NetSparse: Hardware Acceleration for Distributed Sparse Kernels
↳ **Micro 2025** *Co-author*

Gigaflow: Pipeline-Aware Sub-Traversal Caching for Modern SmartNICs
↳ **ASPLOS 2025** *First author*

The Slow-Path Needs an Accelerator Too!
↳ **SIGCOMM CCR 2023** *First author*

Homunculus: Auto-Generating Efficient Data-Plane ML Pipelines for Datacenter Networks
↳ **ASPLOS 2023** *Co-author* ***Distinguished Artifact Award***

List of Awards

Ross fellowship at **Purdue University**

Selected as **Mentor** for **P4 Language Consortium**, **Google Summer of Code 2025**

Research Award for **Celeris CAL'25**

Distinguished Artifact Award for **Homunculus**, **ASPLOS'23**

Travel Awards

ASPLOS'22, SIGCOMM'22, NSDI'25, SIGCOMM'25






PURDUE

Mango Mango
Dessert
mango



DON'T
TRASH
IT!


Thank you!