FlowValve: Packet Scheduling Offloaded on NP-based SmartNICs

ICDCS 2022

Shaoke Xi, Fuliang Li, XingWei Wang

Northeastern University

Zhejiang University

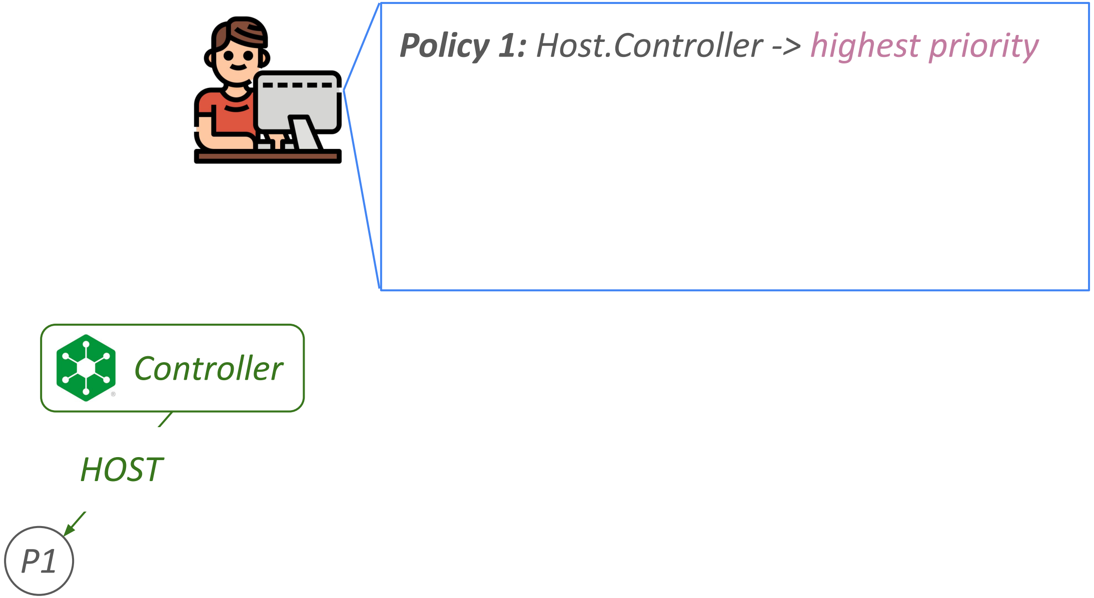
FlowValve is a parallel packet scheduler for Network Processor (NP)-based SmartNICs that offloads critical network functions of Linux TC, including classifying and scheduling.

- What are the requirements of end-host scheduling?
- The offloading idea
- Why using NP-based SmartNICs?

What are the requirements of packet scheduling?

Complex & Flexible Network Policies

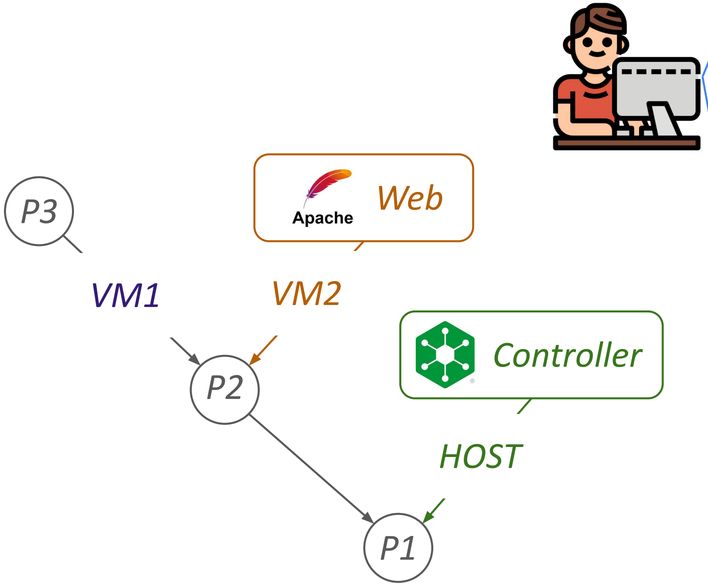
End-host enforces complex network policies to meet SLAs for applications and tenants.



Policy 1: *Host.Controller -> highest priority*

Complex & Flexible Network Policies

End-host enforces complex network policies to meet SLAs for applications and tenants.

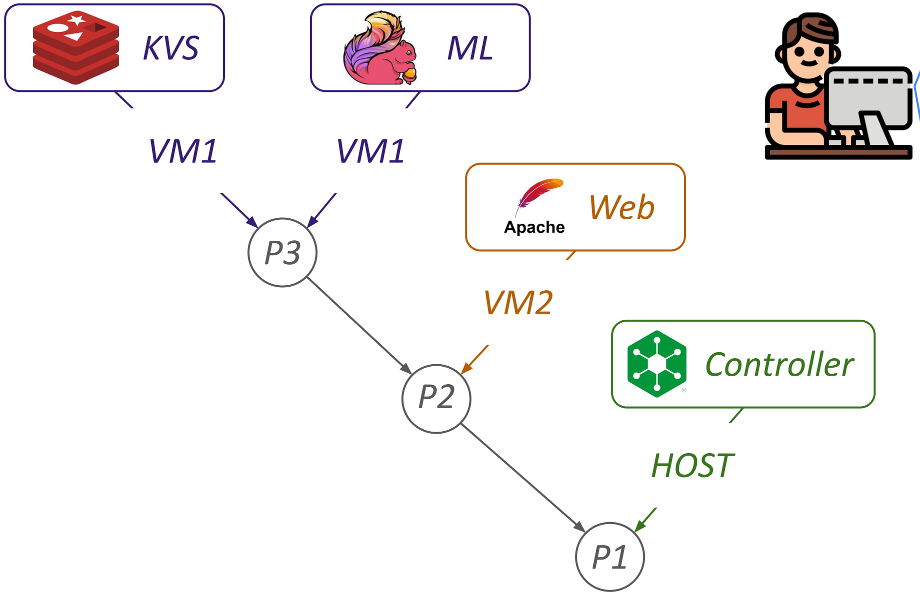


Policy 1: *Host.Controller -> highest priority*

Policy 2: *VM1:VM2 -> weighted sharing*

Complex & Flexible Network Policies

End-host enforces complex network policies to meet SLAs for applications and tenants.



Policy 1: *Host.Controller -> highest priority*

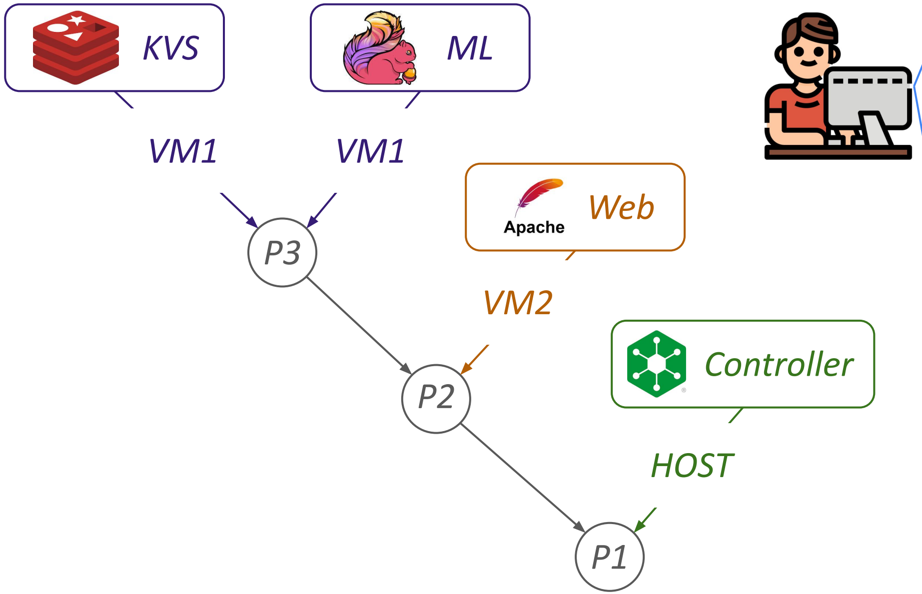
Policy 2: *VM1:VM2 -> weighted sharing*

Policy 3.1: *VM1.KVS prior than VM1.ML*

Policy 3.2: *VM1.ML guaranteed 2Gbps*

Complex & Flexible Network Policies

End-host enforces complex network policies to meet SLAs for applications and tenants.



Policy 1: *Host.Controller -> highest priority*

Policy 2: *VM1:VM2 -> weighted sharing*

Policy 3.1: *VM1.KVS prior than VM1.ML*

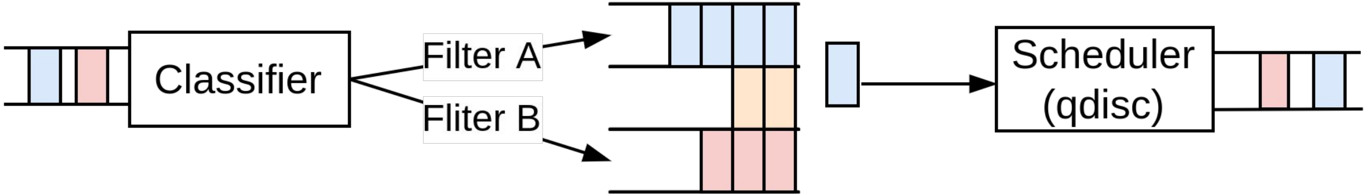
Policy 3.2: *VM1.ML guaranteed 2Gbps*

- Fine-grained traffic control
- Complex network policies
- Flexible support of new algorithms

Efficient Enforcement

Single core scheduler works fine under low packet rate.

Scheduling accuracy drops under heavy traffic workloads (e.g., >10Gbps).



Root cause: Single-core scheduling hits the performance bottleneck.

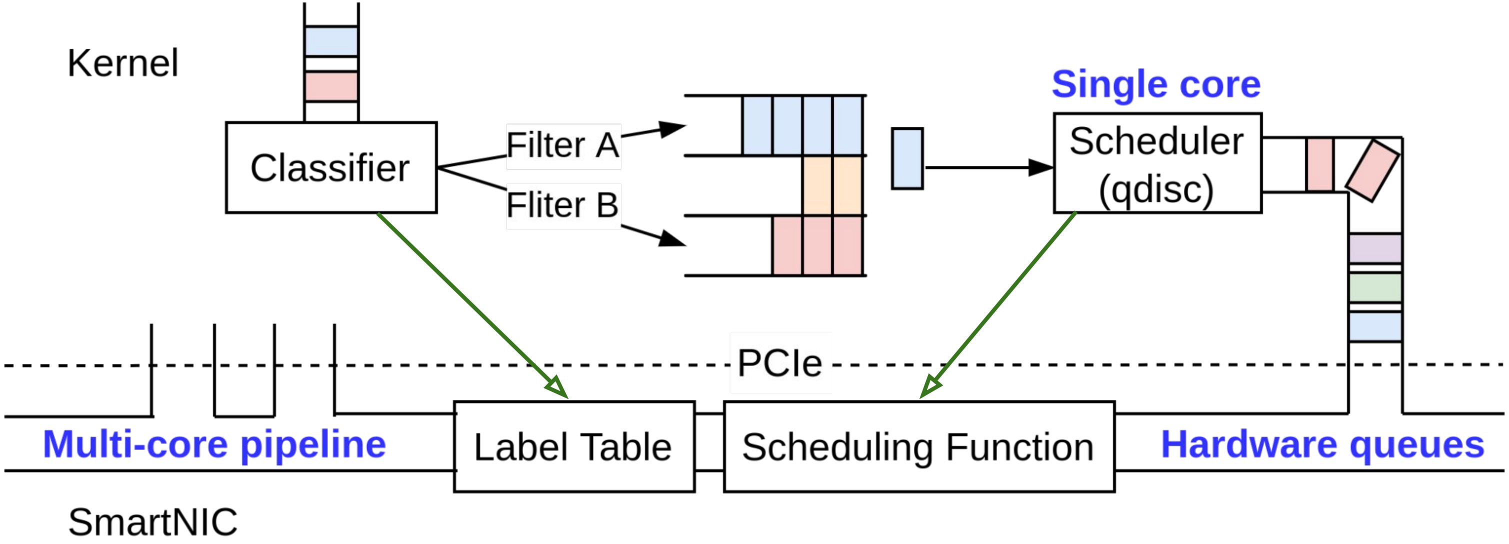
Single core scheduler can easily maintain consistent queue status.

Multi-core scheduling needs inter-core coordination, which is challenging.

The Offloading Idea

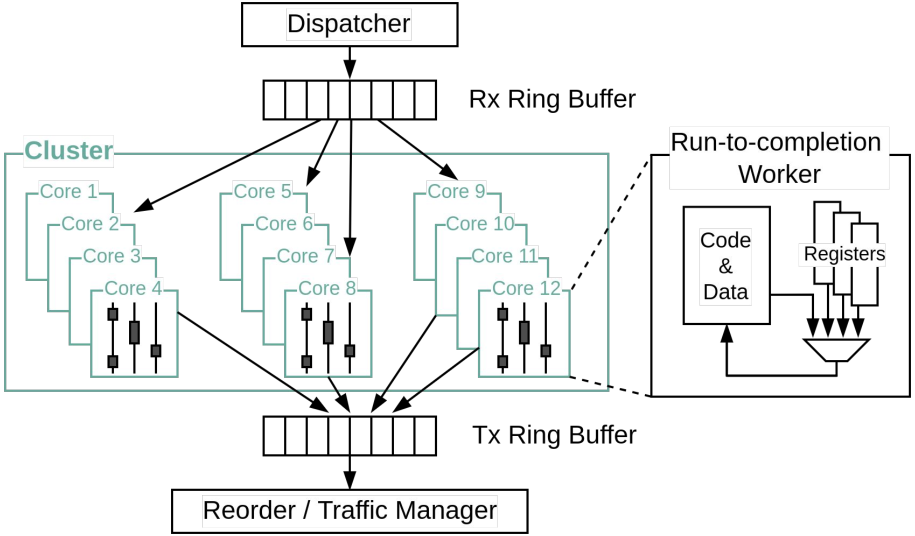
Utilize multi-core hardware to accelerate packet scheduling.

Offload classifying and scheduling functions to save CPU cores.



Why using Network Processor-based SmartNICs?

Parallel Packet Processing

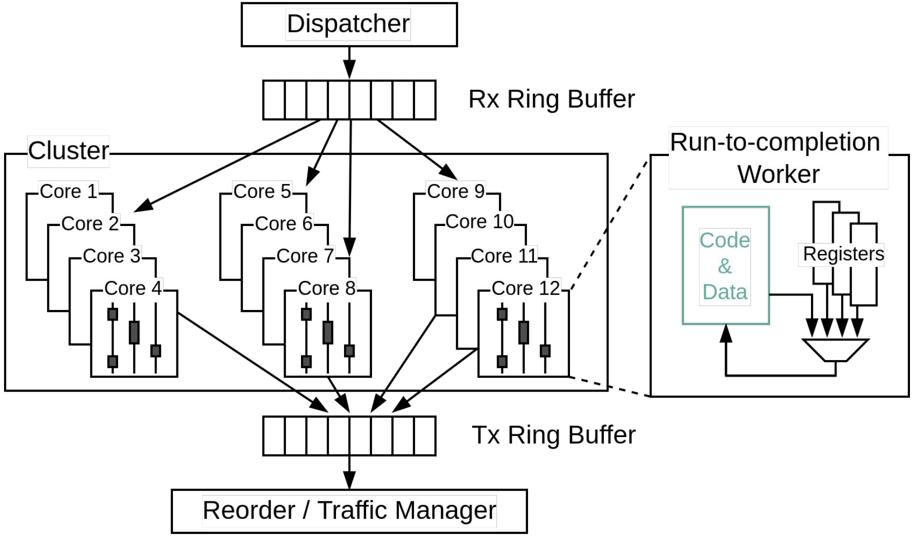


- **Parallel Scheduling**

Embed scheduling function in each core’s processing routine.

Many worker cores coordinate to perform scheduling algorithms.

Parallel Packet Processing



- Parallel Scheduling

Embed scheduling function in each core’s processing routine.

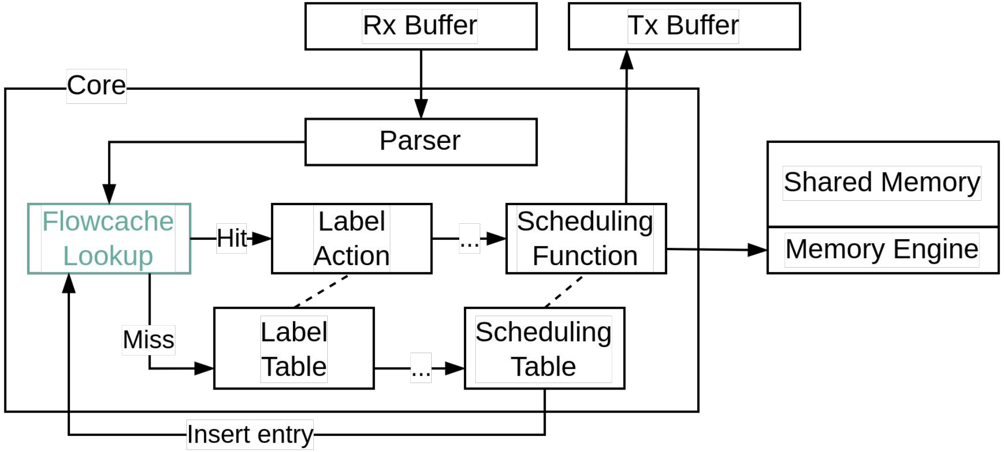
Many worker cores coordinate to perform scheduling algorithms.

- **Flexible Development**

Develop new algorithms in a software manner.

Support P4/Micro-C programming.

Hardware Acceleration

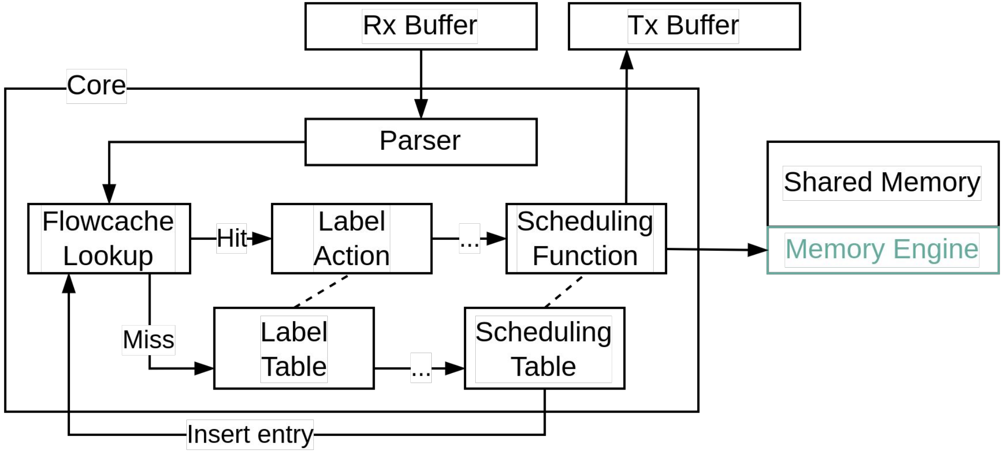


- **Efficient Flowcache**

Specialised cache mechanism accelerate packet classification.

Large on-chip memory caches millions of flows.

Hardware Acceleration



- Efficient Flowcache

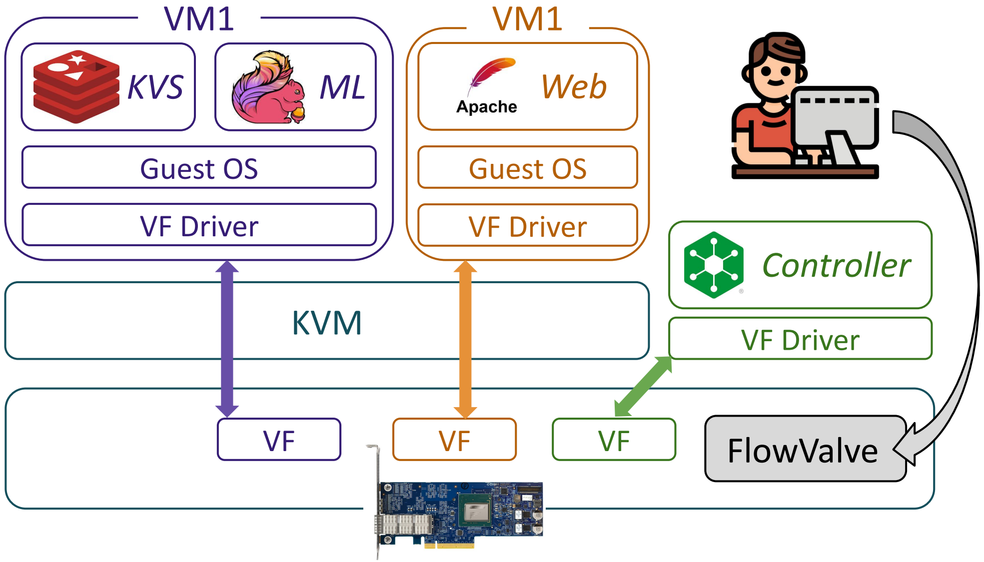
Specialised cache mechanism accelerate packet classification.

Large on-chip memory caches millions of flows.

- **Atomic Instruction**

Memory engines conduct atomic arithmetic operations to alleviate multi-core locking overhead.

Virtualization Support

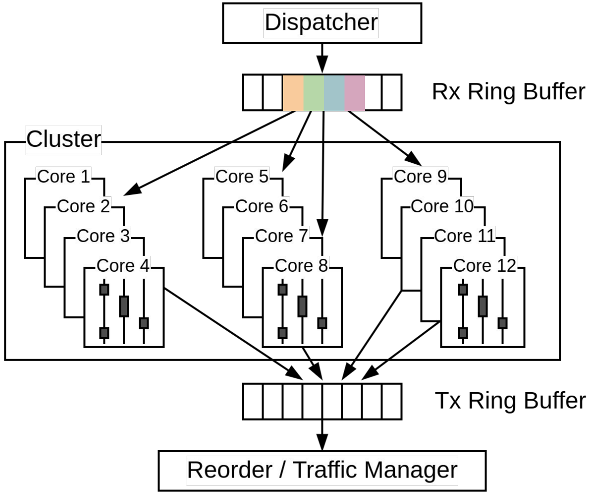


- **Fast Speed**

Deliver high performance to VMs through bypassing the host networking stack.

Meanwhile, conducting network policies on the NIC dataplane.

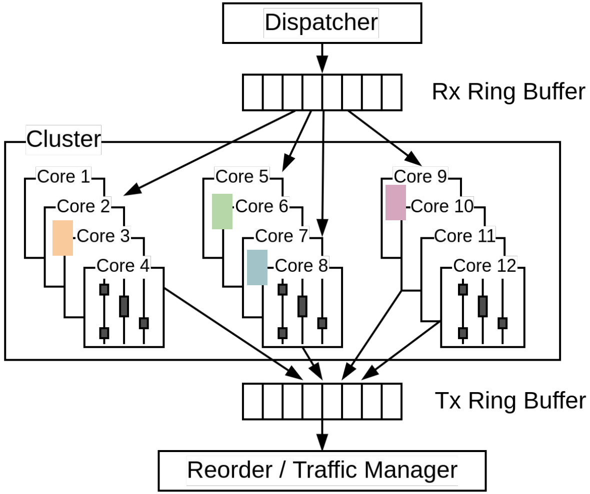
Challenges


Dispatcher
Rx Ring Buffer
Cluster
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
Core 7
Core 8
Core 9
Core 10
Core 11
Core 12
Tx Ring Buffer
Reorder / Traffic Manager

Challenges

- **Multi-core parallelism**

How to reduce inter-core collaboration?



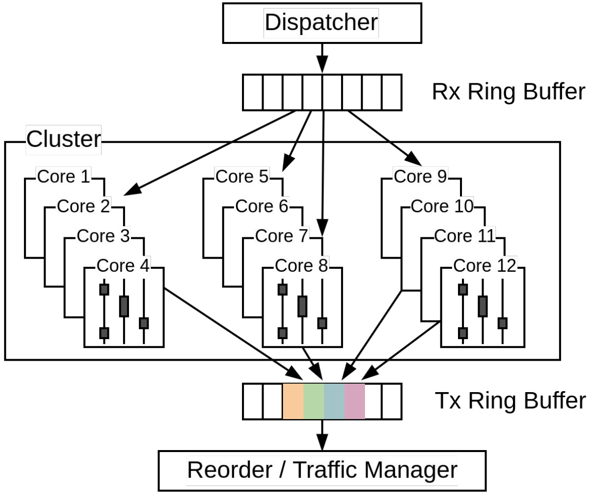
Challenges

- Multi-core parallelism

How to reduce inter-core collaboration?

- **Constrained buffer management**

How to avoid congestion on egress
by handling packets on their way into TX buffers?



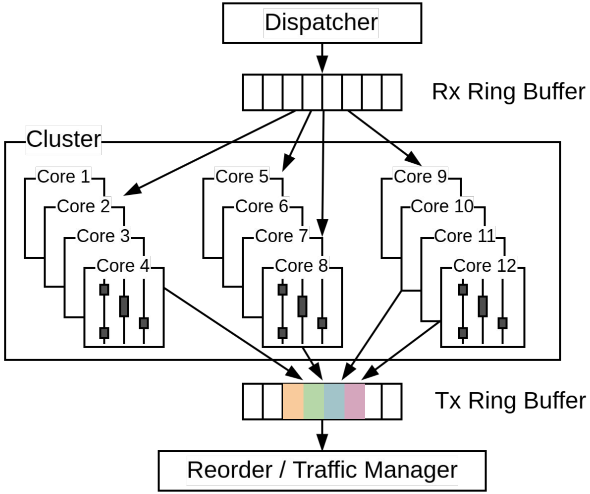
Challenges

- Multi-core parallelism

How to reduce inter-core collaboration?

- Constrained buffer management

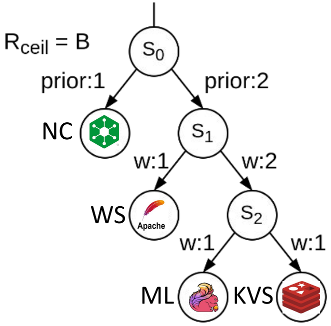
How to avoid congestion on egress
by handling packets on their way into TX buffers?



Insight: Abstract TX buffers as a FIFO queue and perform specialized tail drop to mix the FIFO queue with expected flow proportions.

FlowValve

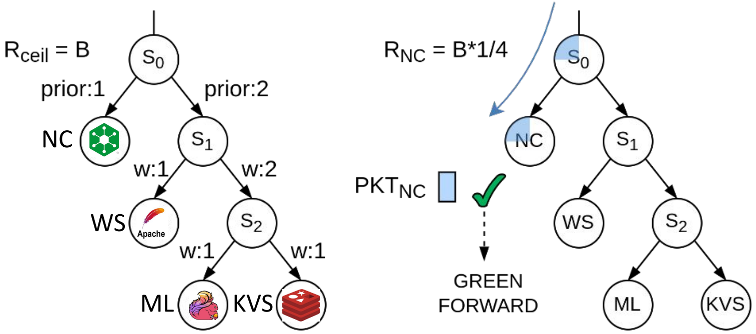
- **Express network policies as a scheduling tree**
  - Traffic classes represent by tree nodes.
  - Flow QoS settings represent by tree paths.



(a)

FlowValve

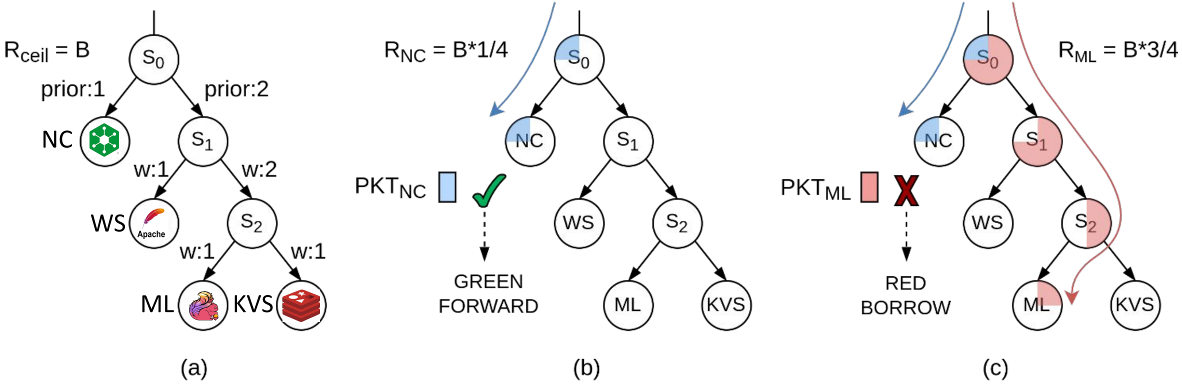
- Express network policies as a scheduling tree
  - Traffic classes represent by tree nodes.
  - Flow QoS settings represent by tree paths.
- **Parallelly update traffic classes on multi-core NPs**
  - Estimate instant flow rate at interior nodes.
  - Enforce rate control at leaf nodes.



(a) (b)

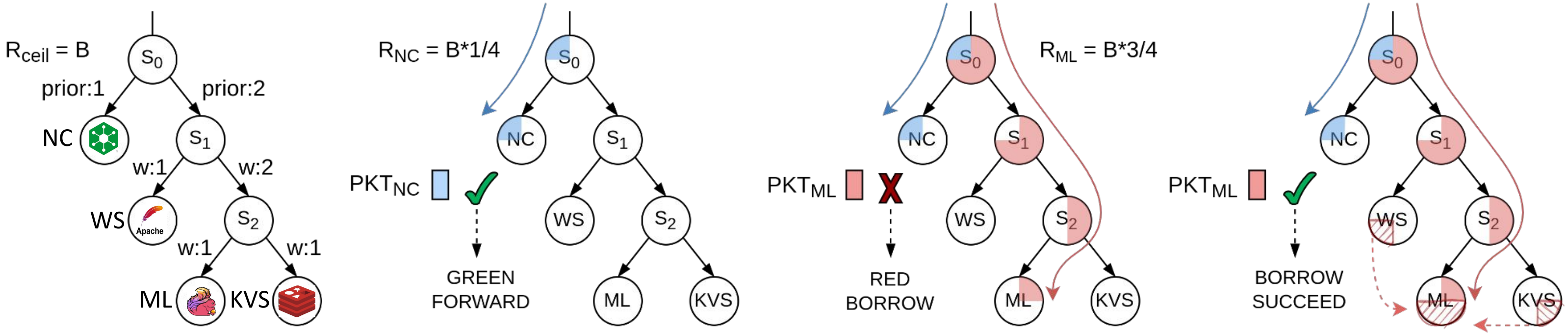
FlowValve

- Express network policies as a scheduling tree
    - Traffic classes represent by tree nodes.
    - Flow QoS settings represent by tree paths.
- **Parallelly update traffic classes on multi-core NPs**
    - Estimate instant flow rate at interior nodes.
    - Enforce rate control at leaf nodes.



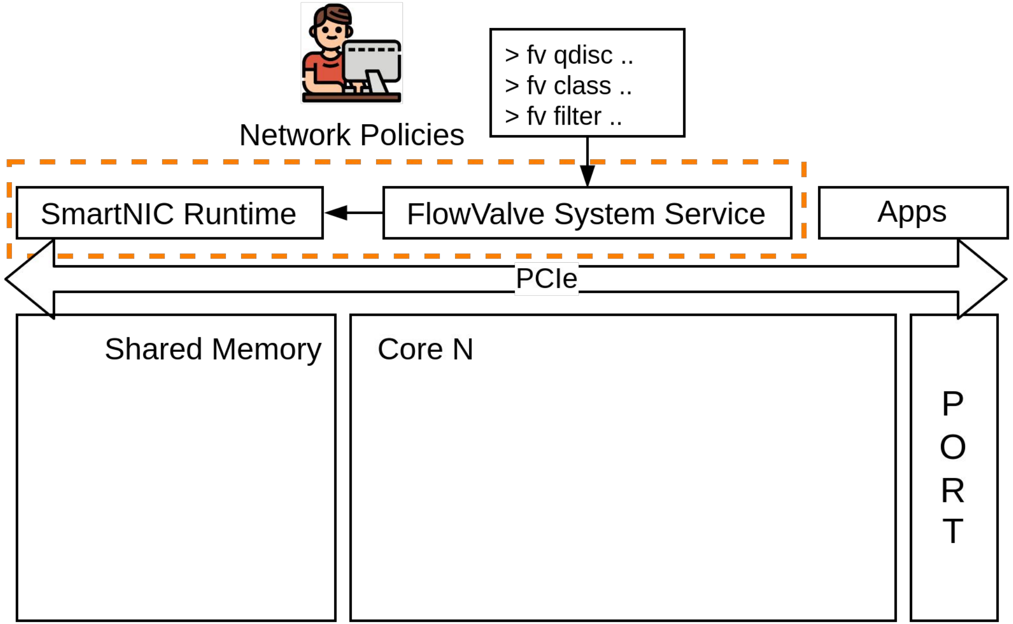
FlowValve

- Express network policies as a scheduling tree
    - Traffic classes represent by tree nodes.
    - Flow QoS settings represent by tree paths.
- **Parallelly update traffic classes on multi-core NPs**
    - Estimate instant flow rate at interior nodes.
    - Enforce rate control at leaf nodes.



(a) (b) (c) (d)

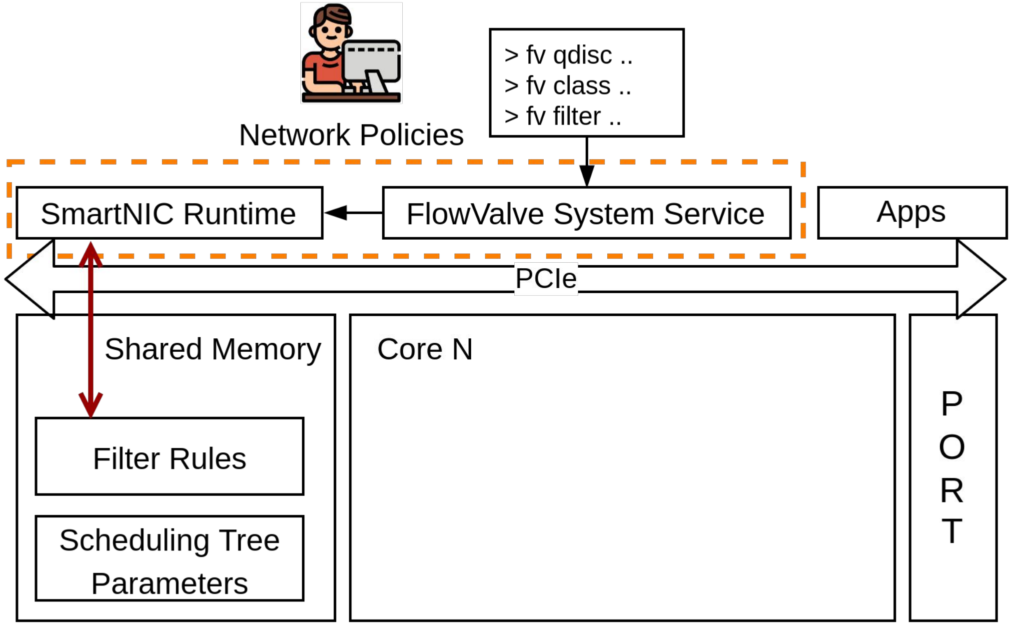
Workflow



- **Frontend**

Take in network policies to construct the scheduling tree.

Workflow

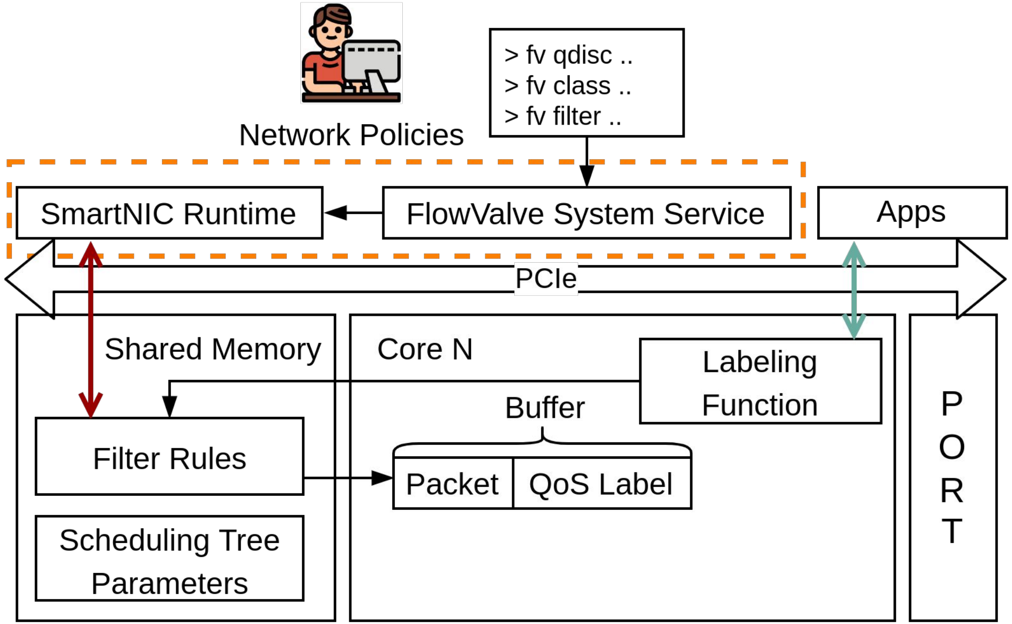


- **Frontend**

Take in network policies to construct the scheduling tree.

Populate tree parameters to NICs.

Workflow



- Frontend

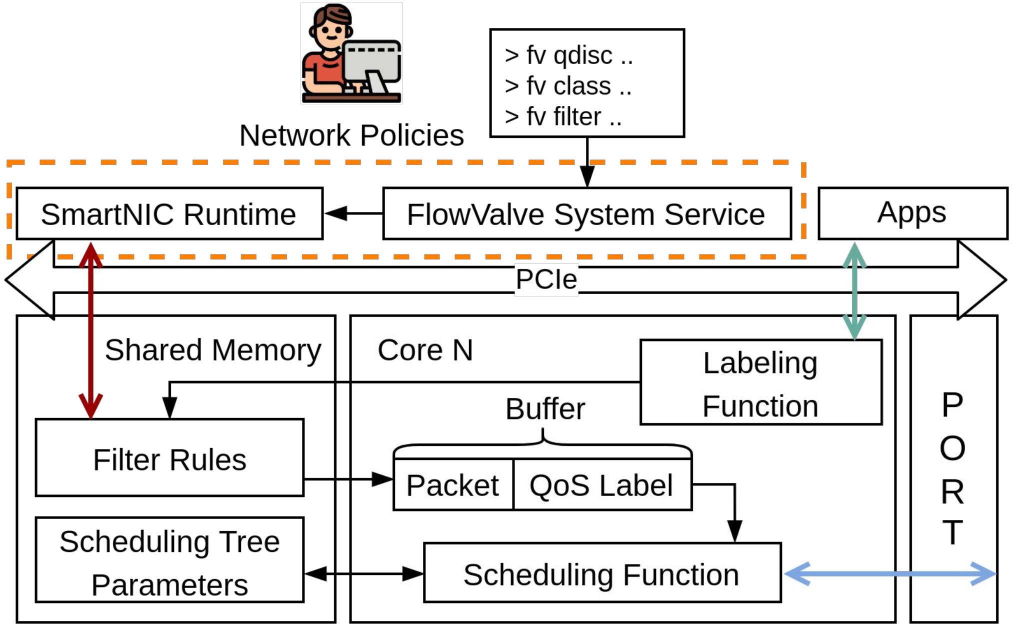
Take in network policies to construct the scheduling tree.

Populate tree parameters to NICs.

- **Backend**

Execute labeling function to get QoS setting labels.

Workflow



- Frontend

Take in network policies to construct the scheduling tree.

Populate tree parameters to NICs.

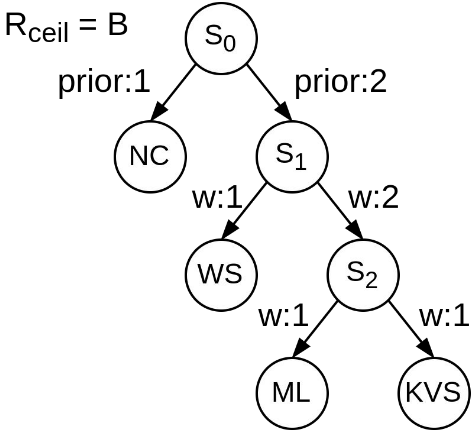
- **Backend**

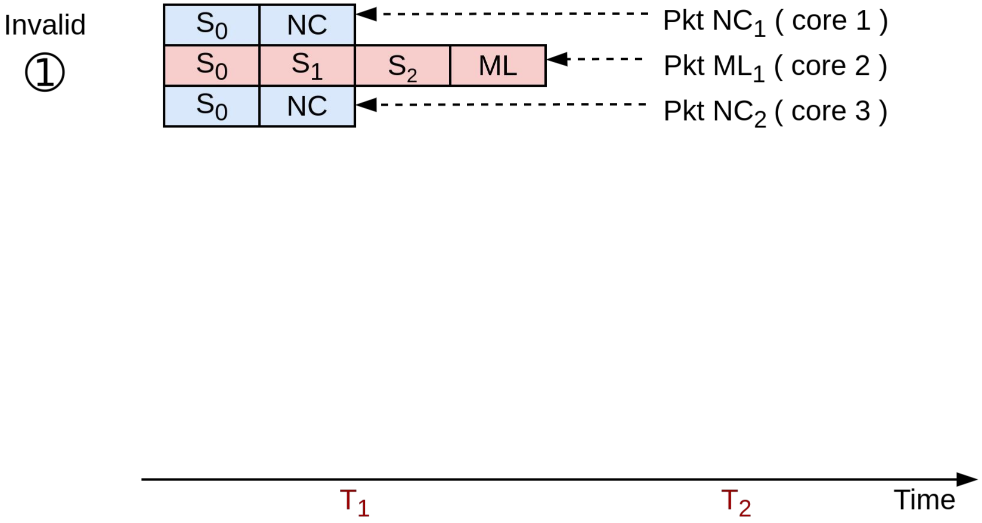
Execute labeling function to get QoS setting labels.

Execute scheduling function to update traffic classes.

Scheduling Function

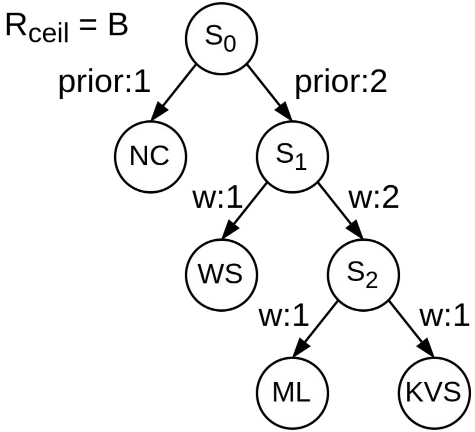
- **Class update without synchronization on NPs is inaccurate ✗**

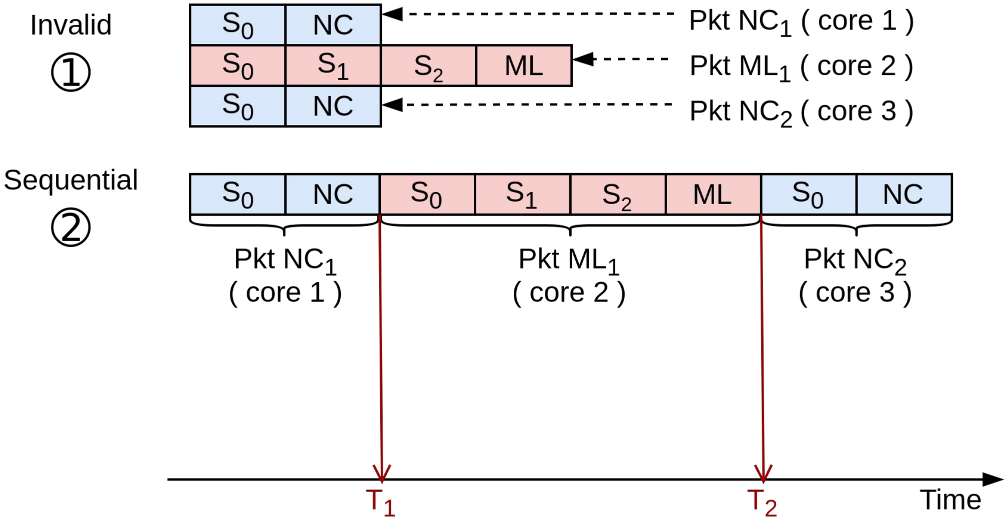




Scheduling Function

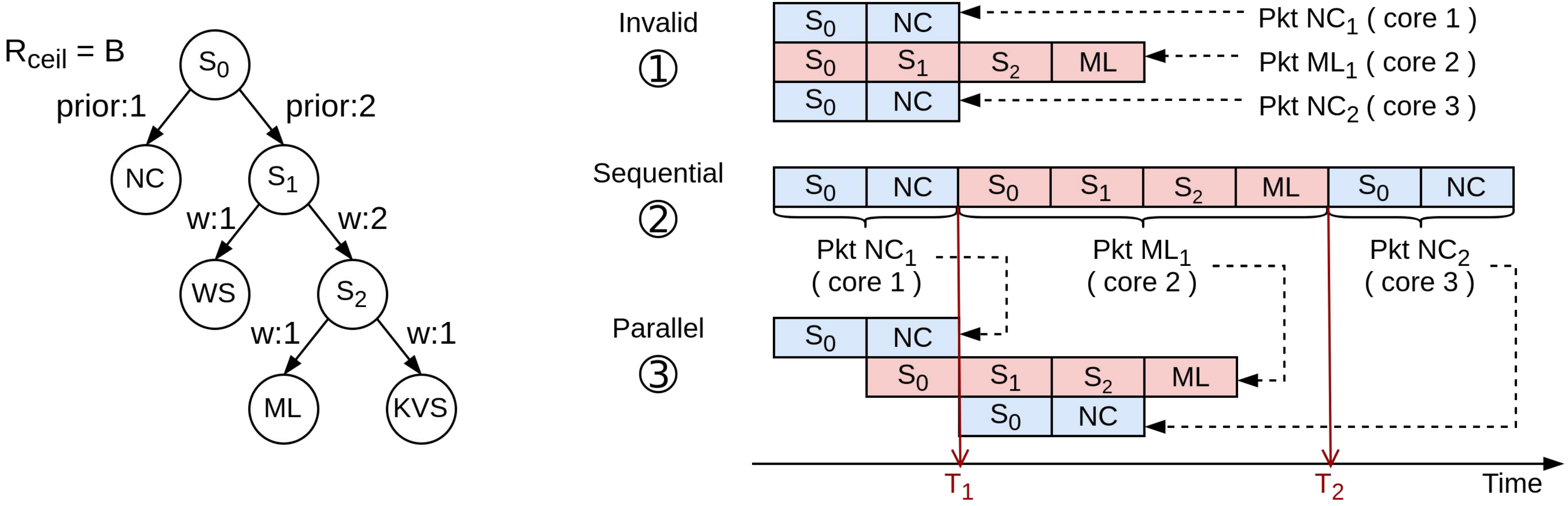
- Class update without synchronization on NPs is inaccurate ✗
- **Sequential update leads to extreme low throughput ✗**





Scheduling Function

- Class update without synchronization on NPs is inaccurate ✗
- Sequential update leads to extreme low throughput ✗
- **Locking at the class level balances accuracy and efficiency ✓**



Hierarchical Rate-limiting

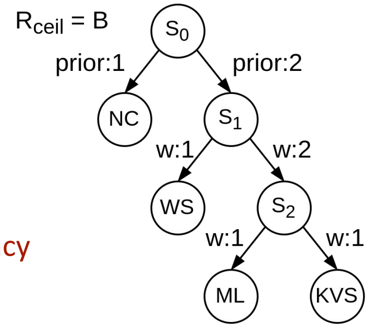
- **Restrict flow rate with token buckets**

token rate of class *C*

$$\theta_C = \frac{b_C}{f} \quad (1)$$

bit rate of class *C*

worker core frequency



Hierarchical Rate-limiting

- Restrict flow rate with token buckets

bit rate of class *C*

token rate of class *C*

$$\theta_C = \frac{b_C}{f} \quad (1)$$

worker core frequency

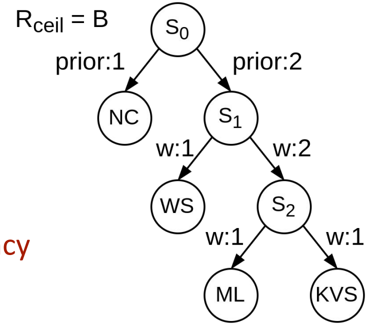
- **Adjust token rate at runtime**
  - Priority

token consumption rate of class *NC*

tokens consumed by forwarded packets

$$\theta_{S_1} = \theta_{S_0} - \Gamma_{NC}, \Gamma_{NC} = \frac{\sum L_P}{\Delta T}, P \in NC \quad (2)$$

update interval



Hierarchical Rate-limiting

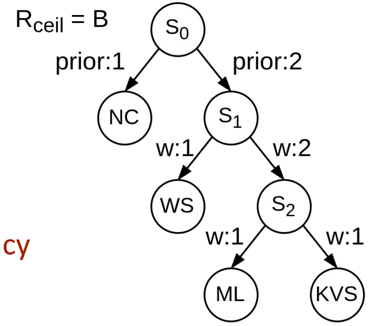
- Restrict flow rate with token buckets

token rate of class *C*

$$\theta_C = \frac{b_C}{f} \quad (1)$$

bit rate of class *C*

worker core frequency



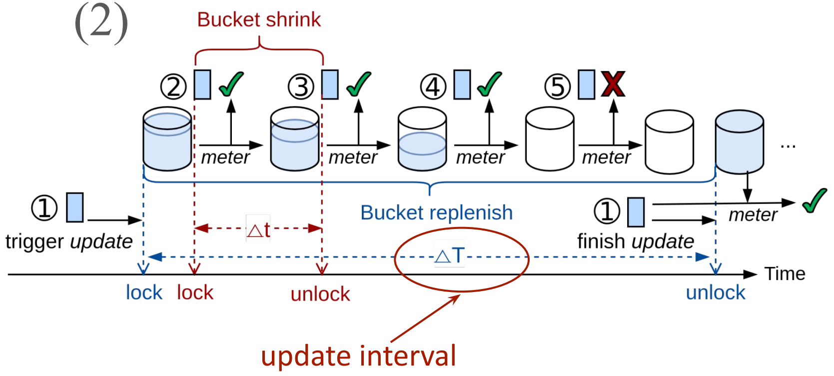
- **Adjust token rate at runtime**
  - Priority

token consumption rate of class *NC*

tokens consumed by forwarded packets

$$\theta_{S_1} = \theta_{S_0} - \Gamma_{NC}, \Gamma_{NC} = \frac{\sum L_P}{\Delta T}, P \in NC \quad (2)$$

update interval



Hierarchical Rate-limiting

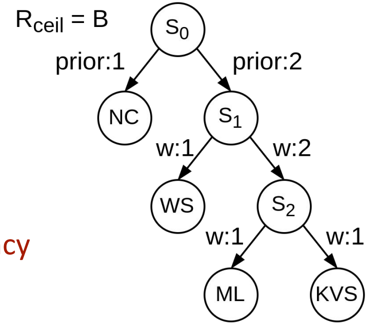
- Restrict flow rate with token buckets

token rate of class *C*

bit rate of class *C*

$$\theta_C = \frac{b_C}{f} \qquad (1)$$

worker core frequency



- **Adjust token rate at runtime**
  - Priority

token consumption rate of class *NC*

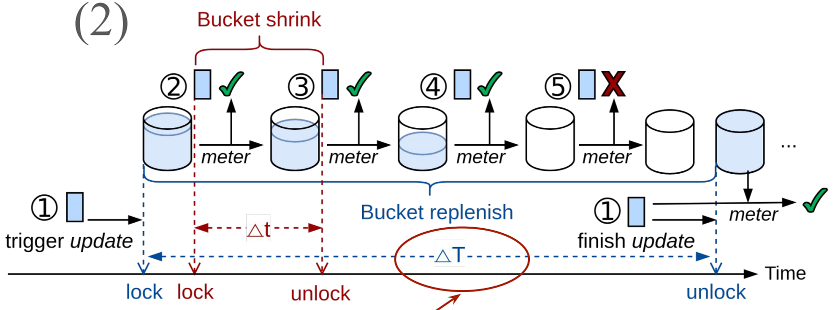
tokens consumed by forwarded packets

$$\theta_{S_1} = \theta_{S_0} - \Gamma_{NC}, \Gamma_{NC} = \frac{\sum L_P}{\Delta T}, P \in NC \qquad (2)$$

update interval

  - Weight

$$\theta_{WS} = \theta_{S_1} \times \frac{1}{3}, \theta_{S_2} = \theta_{S_1} \times \frac{2}{3} \qquad (3)$$



update interval

Bandwidth Sharing

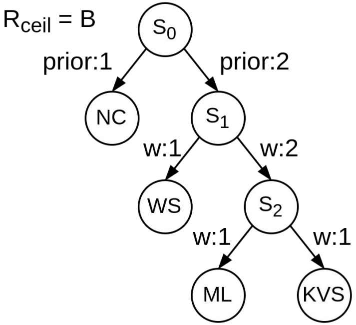
- **Sharing of unconsumed tokens**

token rate of class *C*

lendable token rate of class *C*

$$\theta_{lendable} = \theta_C - \Gamma_C$$

token consumption rate of class *C*



Bandwidth Sharing

- Sharing of unconsumed tokens

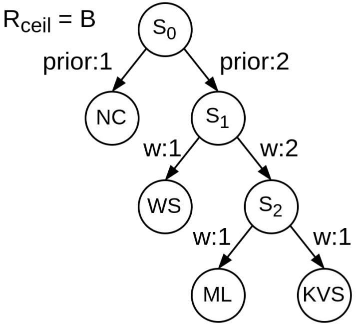
lendable token rate of class *C*

token rate of class *C*

$$\theta_{lendable} = \theta_C - \Gamma_C$$

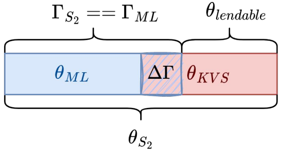
token consumption rate of class *C*

- **Preferent sharing among interior classes**



Example: *KVS* is idle. *WS* and *ML* are hungry.

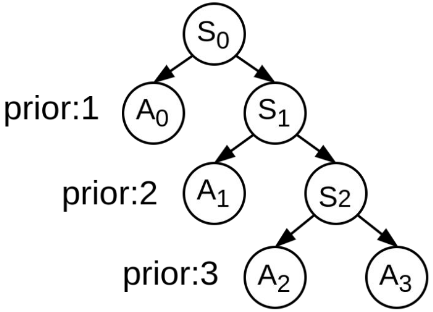
WS borrows from *S*2. *ML* borrows from *KVS*.

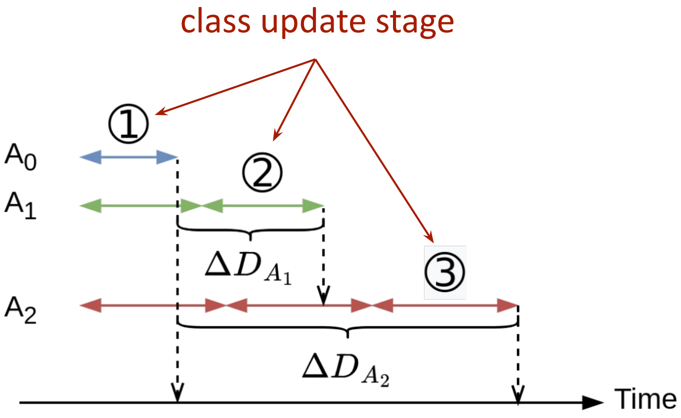


$$\theta_{lendable} = \theta_{S_2} - \Gamma_{S_2}$$

Error Analysis

- Single class rate-limiting is accurate
- **Main error: propagation delay of token rate adjustment**





Experiment Setup

- Implementation
  - Frontend: Python
  - Backend: Netronome Agilio CX 40GbE SmartNIC
- Testbed
  - Hardware
    - Nertonome SmartNIC: send + schedule
    - Intel X710 40GbE NIC: receive
  - FlowValve: DPDK driver + mTCP stack
  - Software scheduler
    - Linux HTB: iperf3 traffic generator
    - DPDK QoS Scheduler

Evaluation

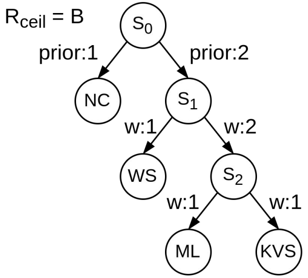
- **QoS Policy Enforcement**

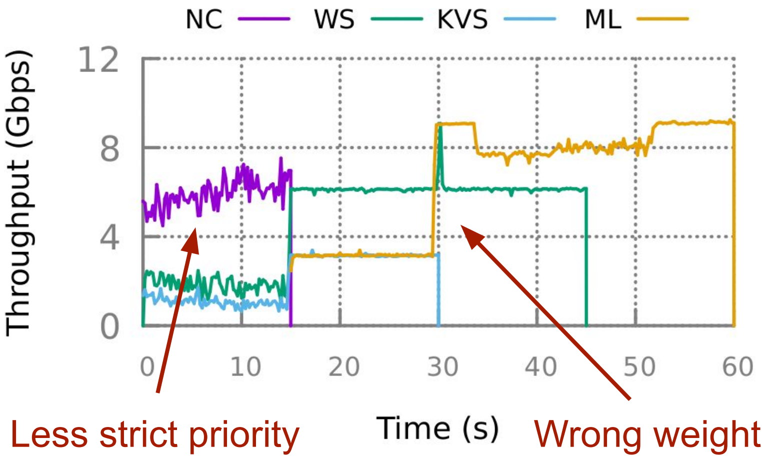
Q1: Can FlowValve enforce network policies?

Q2: Can FlowValve drive line rate?

QoS Policy Enforcement

FlowValve offers better rate conformance than HTB on a 10Gbps link.

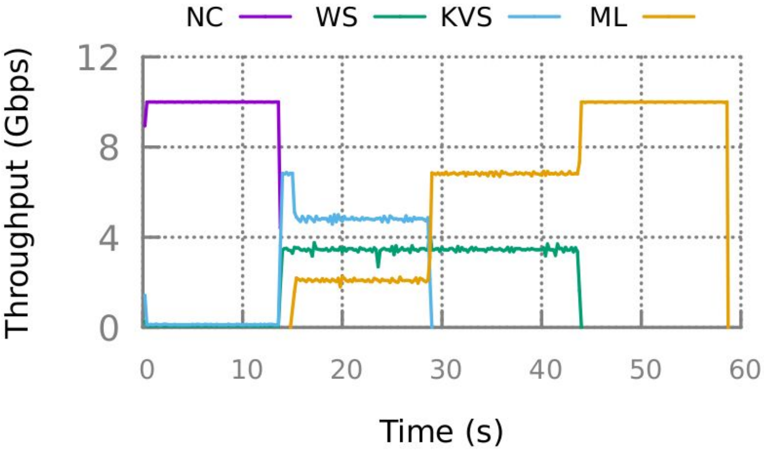




Less strict priority

Wrong weights

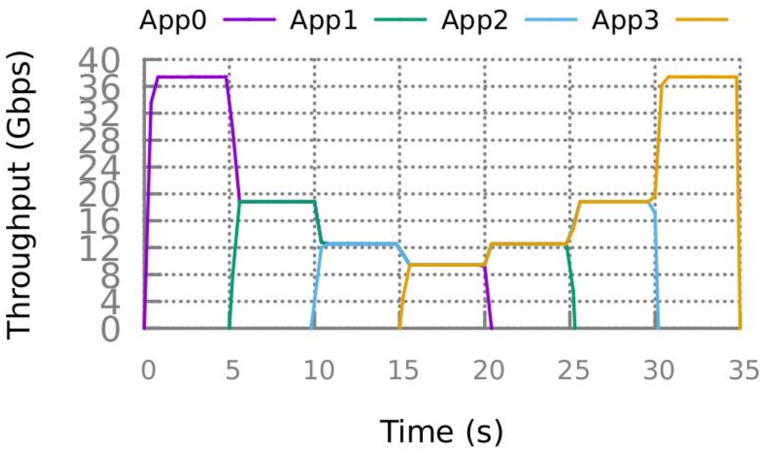
HTB



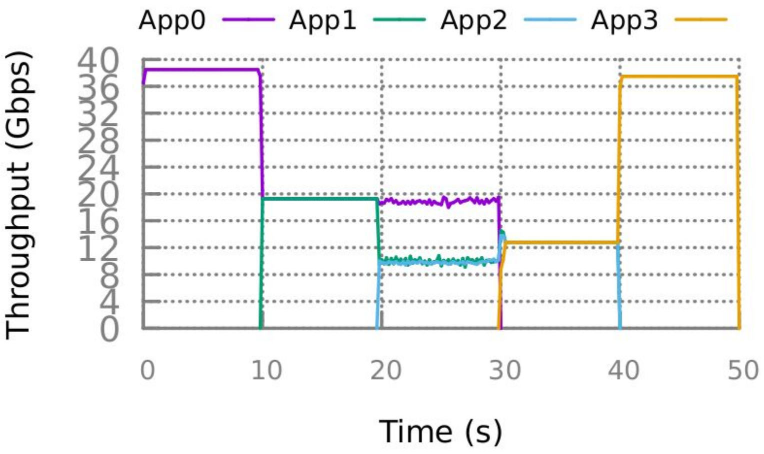
FlowValve

QoS Policy Enforcement

FlowValve drives to line rate while accurately scheduling traffic.



FQ



WQ

Evaluation

- QoS Policy Enforcement

Q1: Can FlowValve enforce network policies?

Q2: Can FlowValve drive line rate?

- **Offloading Effectiveness**

Q3: How many CPU cores can FlowValve save?

Q4: How does FlowValve impact transmission delay?

Offloading Effectiveness

FlowValve contributes to save at least 2 CPU cores when driving line rate.

Packet Size (Byte)	FlowValve	DPDK QoS Scheduler	
	Maximum Throughput (Mpps)	Maximum Throughput (Mpps)	Used Cores
1518	3.23	2.25	1
		3.24	2
1024	4.75	4.49	2
64	19.69	9.06	4

Offloading Effectiveness

FlowValve significantly lowers delay variation.

Bandwidth (Gbps)	Scheduler	One-way Delay (us)	
		Mean	Standard Deviation
10	HTB	36.74	348.25
	FlowValve	30.05	0.30
	DPDK QoS	50.51	41.06
40	FlowValve	162.93 (161.01)	0.30 (0.11)
	DPDK QoS	70.38	83.29

Conclusion

- FlowValve is the first parallel packet scheduler for NP-based SmartNICs that offloads critical functions of Linux traffic control.
- FlowValve offers high throughput and substantially reduces CPU burdens.

Thanks!