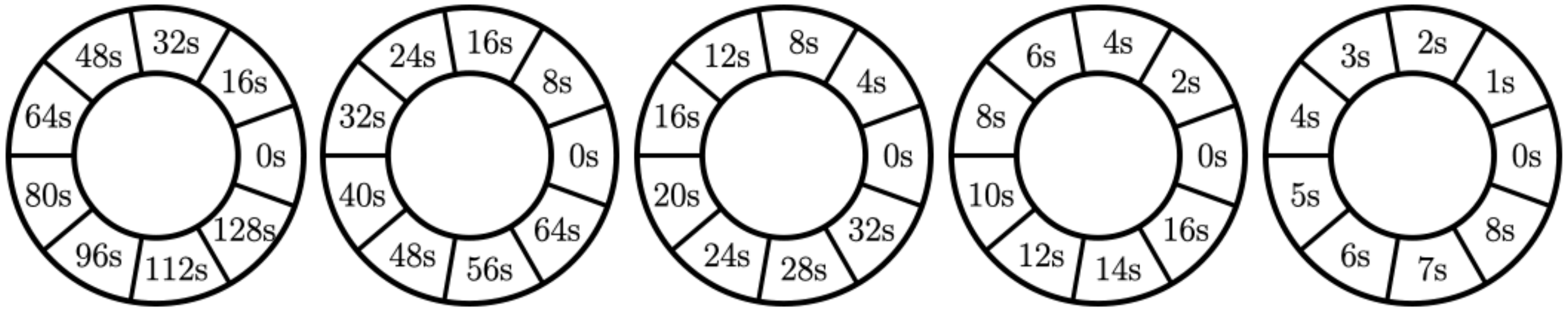
IEEE INFOCOM



Burst can be Harmless: Achieving Line-rate Software Traffic Shaping by *Inter-flow Batching*

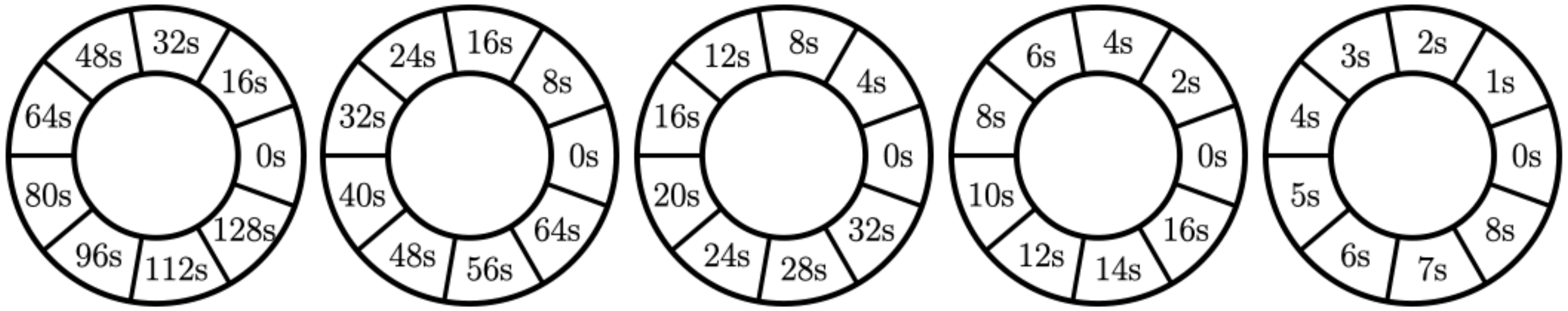
Danfeng Shan, Shihao Hu, Yuqi Liu, Wanchun Jiang, Hao Li, Peng Zhang, Yazhe Tang, Huanzhao Wang, and Fengyuan Ren

西安交通大學 XI'AN JIAOTONG UNIVERSITY

中南大學 CENTRAL SOUTH UNIVERSITY

清華大學 Tsinghua University

IEEE INFOCOM



Burst can be Harmless: Achieving Line-rate Software Traffic Shaping by *Inter-flow Batching*

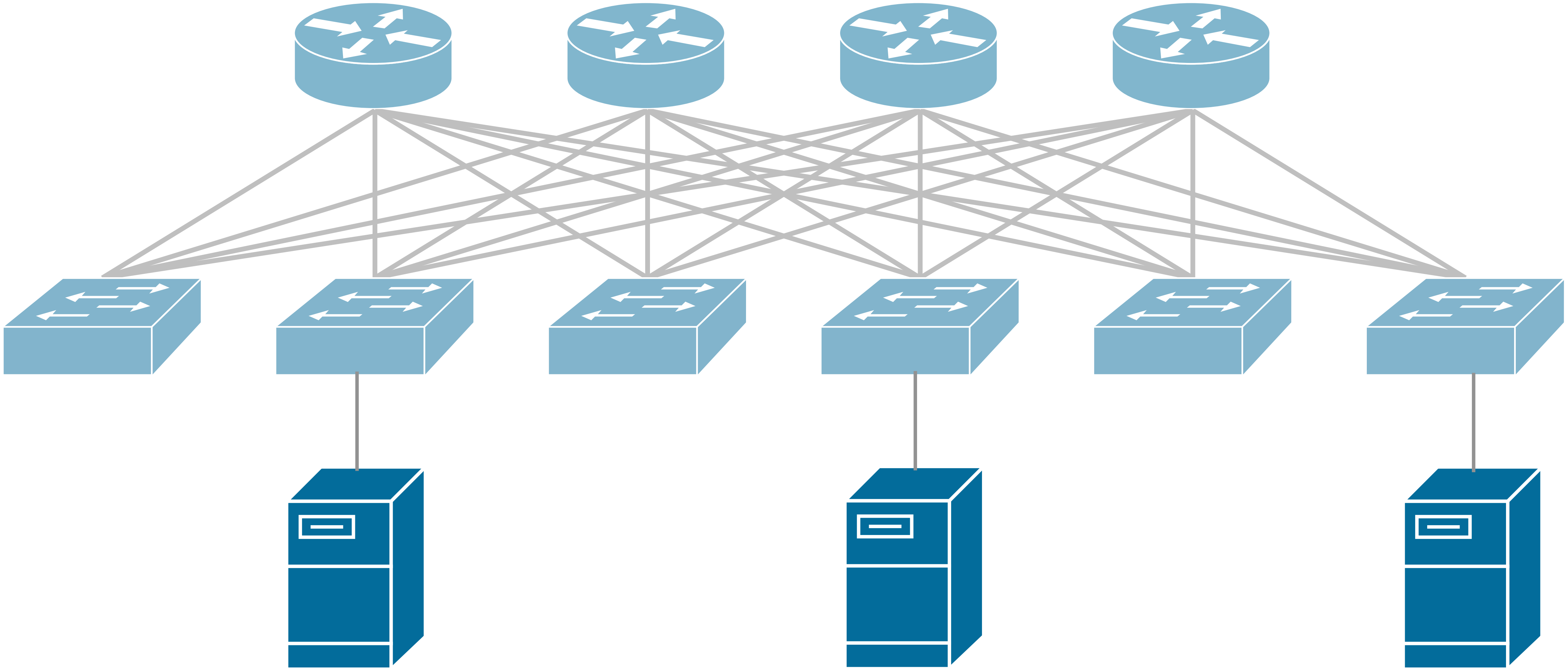
Danfeng Shan, Shihao Hu, Yuqi Liu, Wanchun Jiang, Hao Li, Peng Zhang, Yazhe Tang, Huanzhao Wang, and Fengyuan Ren

西安交通大學 XI'AN JIAOTONG UNIVERSITY

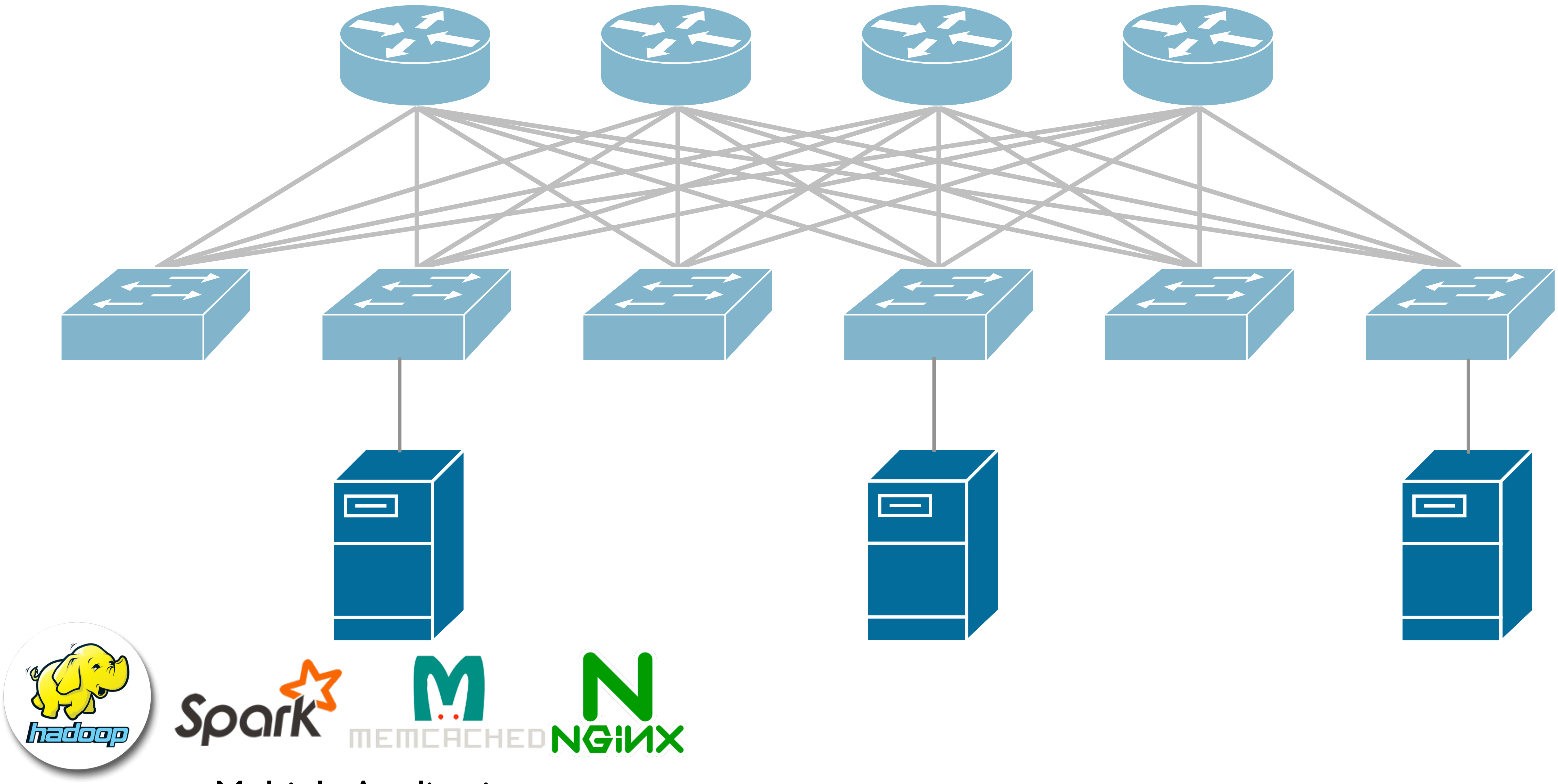
中南大學 CENTRAL SOUTH UNIVERSITY

清華大學 Tsinghua University

Traffic Shaping / Rate Limiting at End Hosts

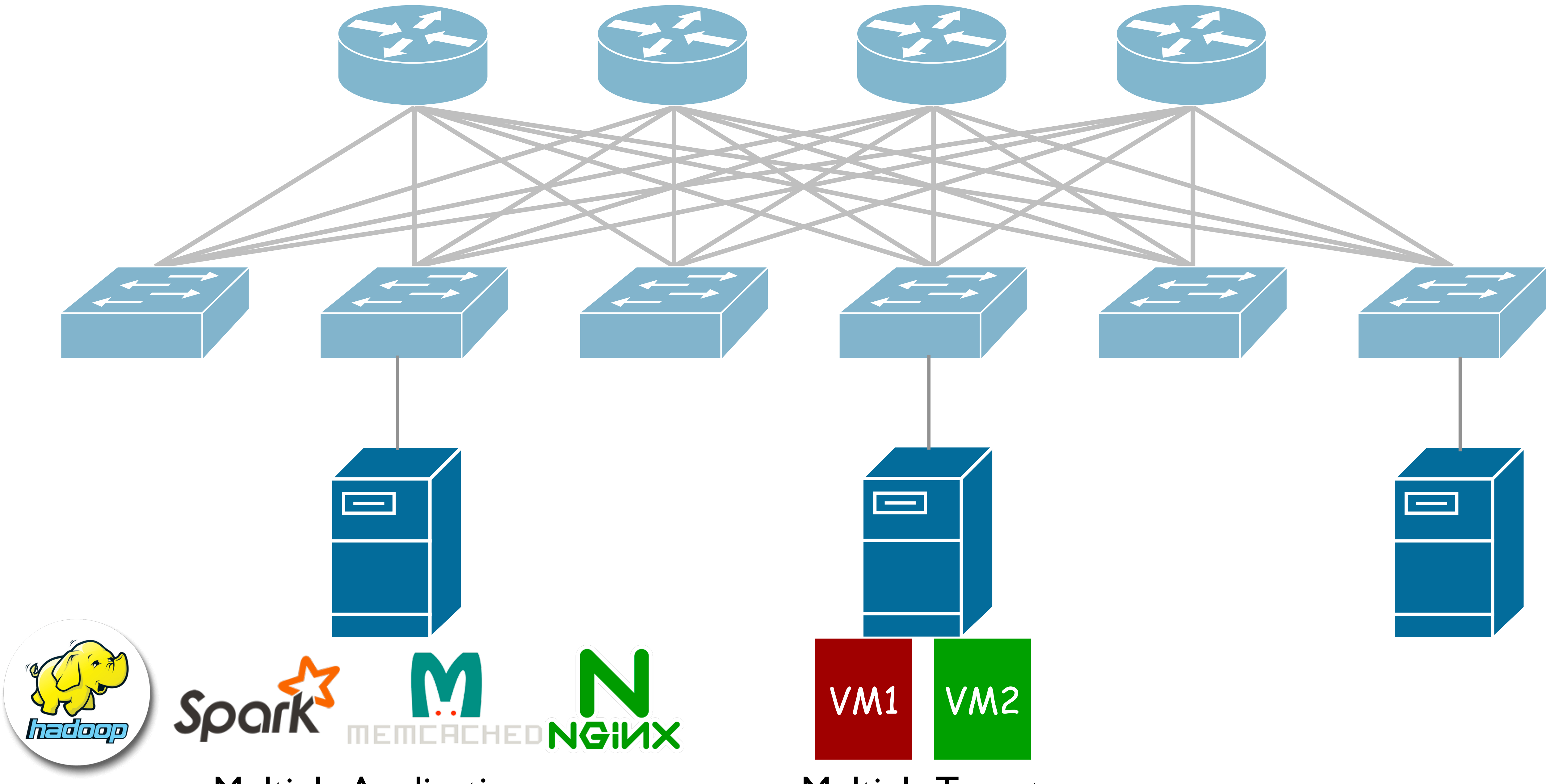


Traffic Shaping / Rate Limiting at End Hosts



Multiple Applications

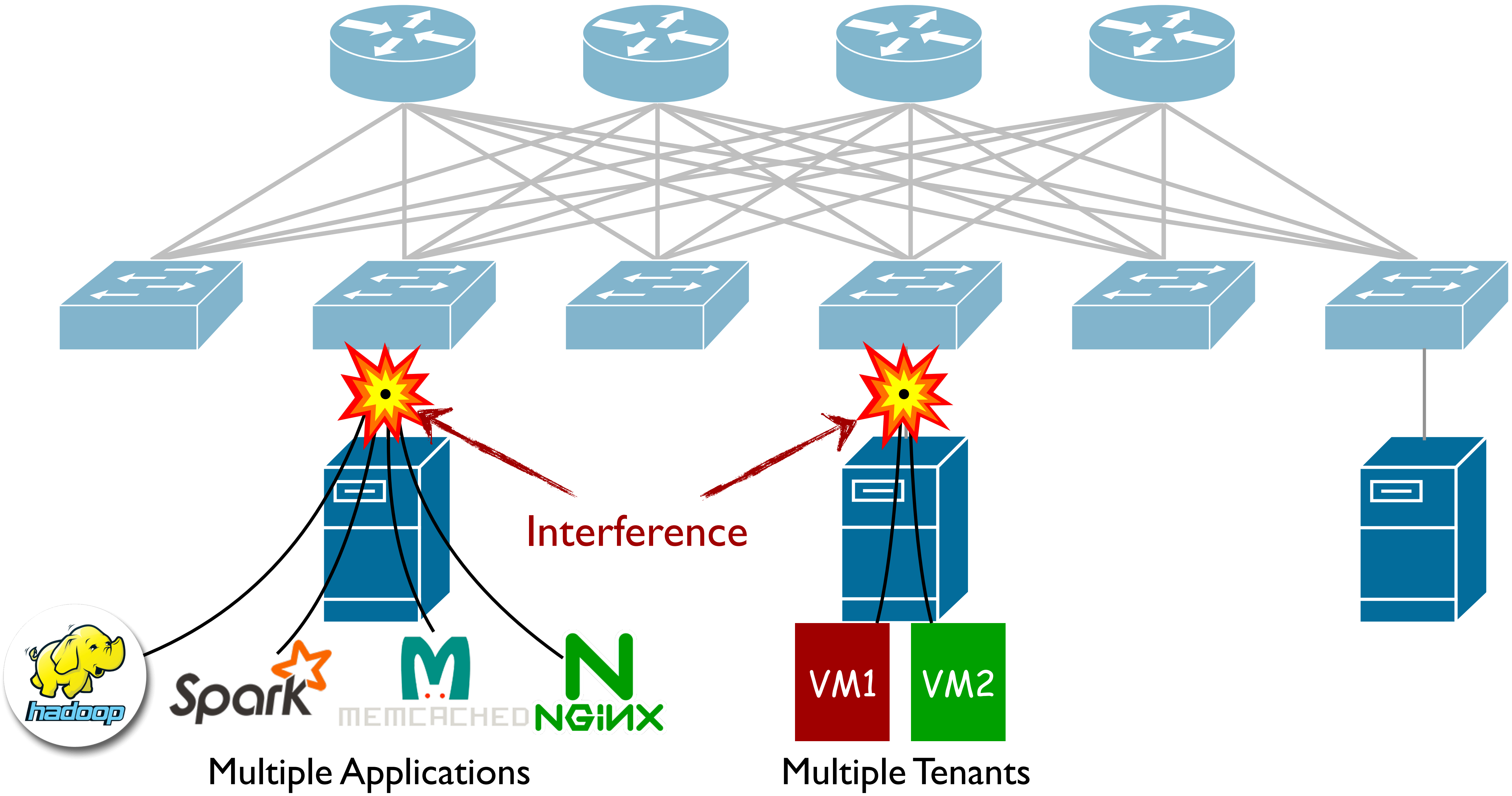
Traffic Shaping / Rate Limiting at End Hosts



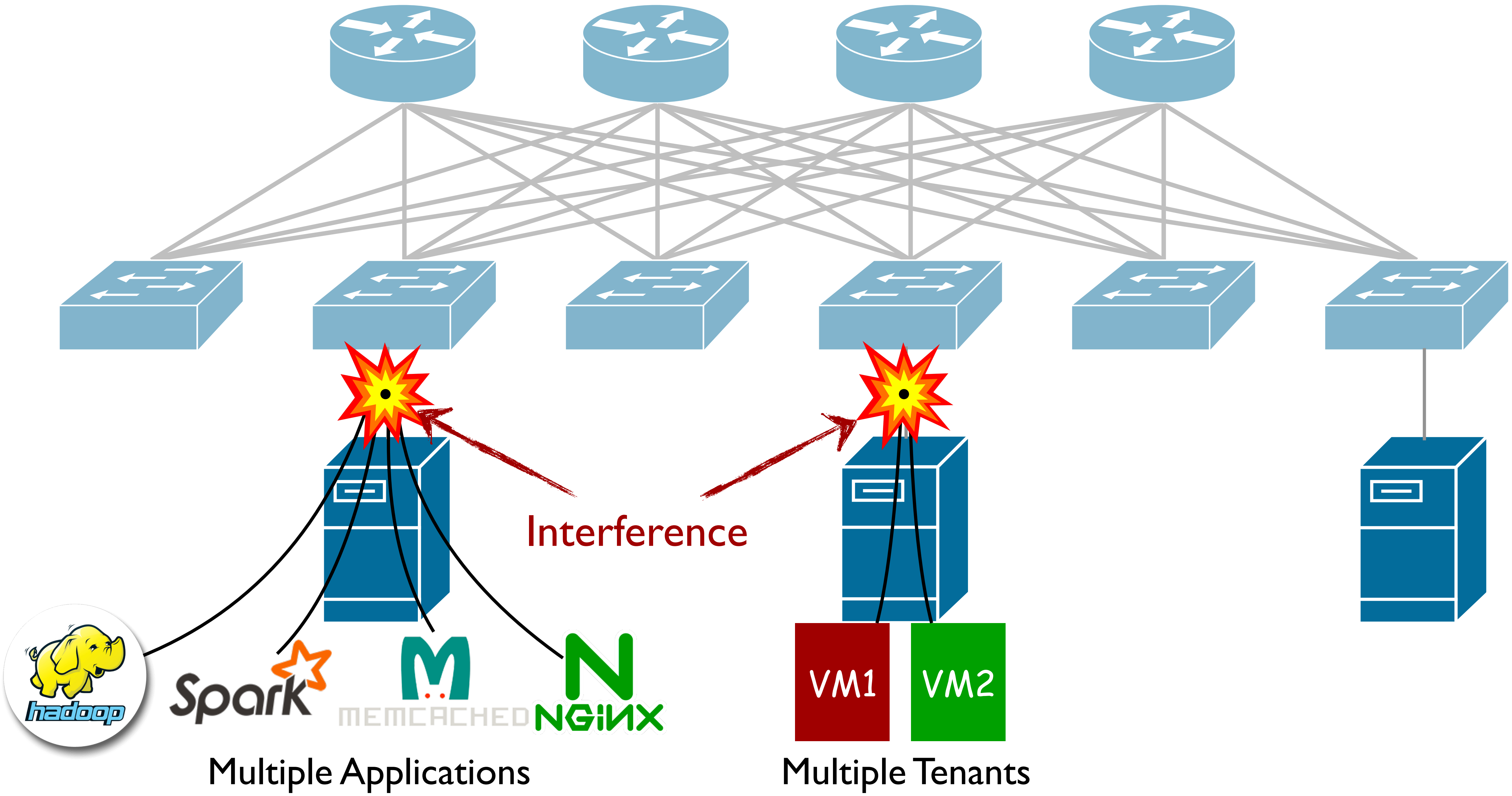
Multiple Applications

Multiple Tenants

Traffic Shaping / Rate Limiting at End Hosts



Traffic Shaping / Rate Limiting at End Hosts



Interference

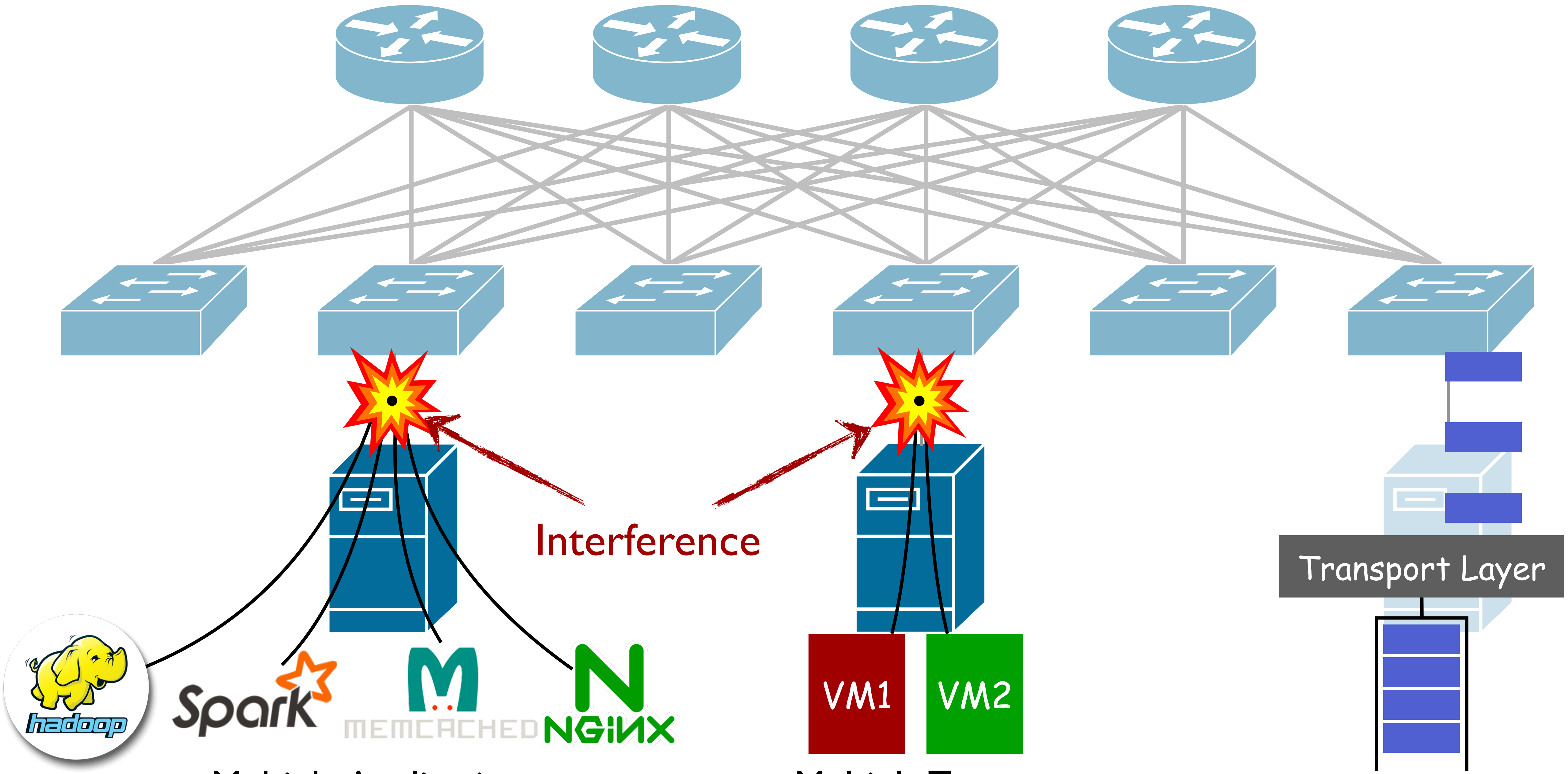
Multiple Applications

Multiple Tenants

❶ Performance Isolation

- Throttle traffic rate

Traffic Shaping / Rate Limiting at End Hosts



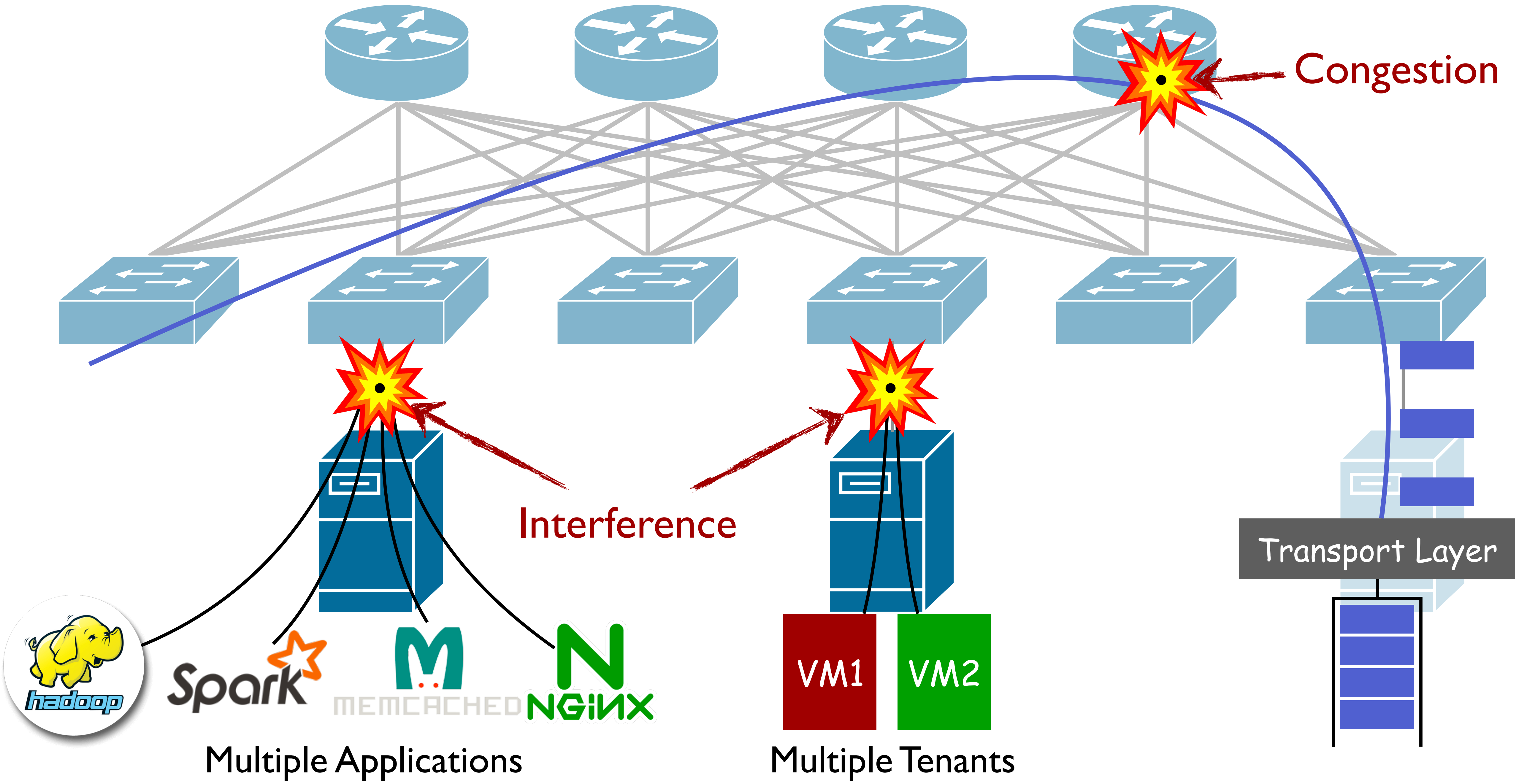
Multiple Applications

Multiple Tenants

❶ Performance Isolation

- Throttle traffic rate

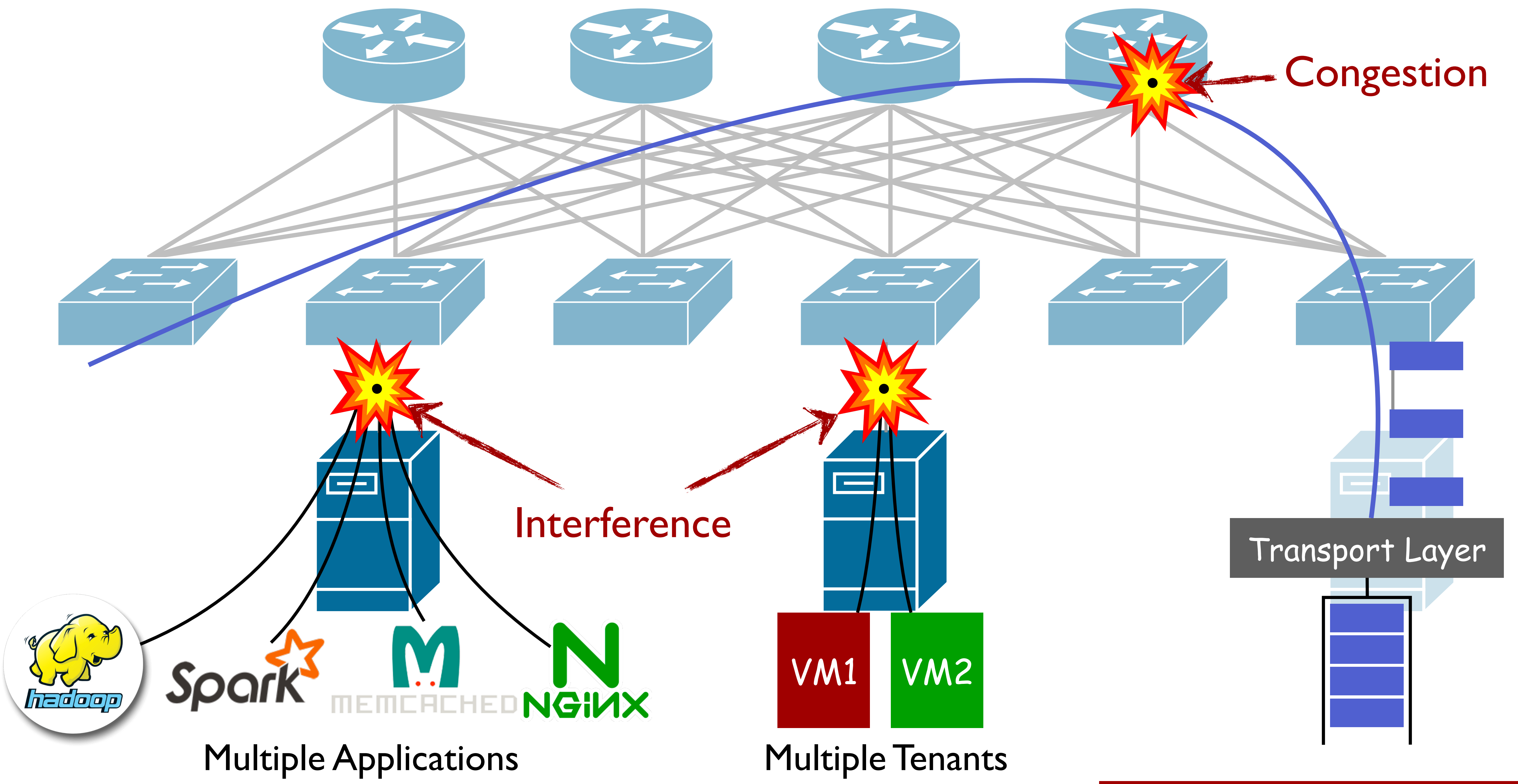
Traffic Shaping / Rate Limiting at End Hosts



❶ Performance Isolation

- Throttle traffic rate

Traffic Shaping / Rate Limiting at End Hosts



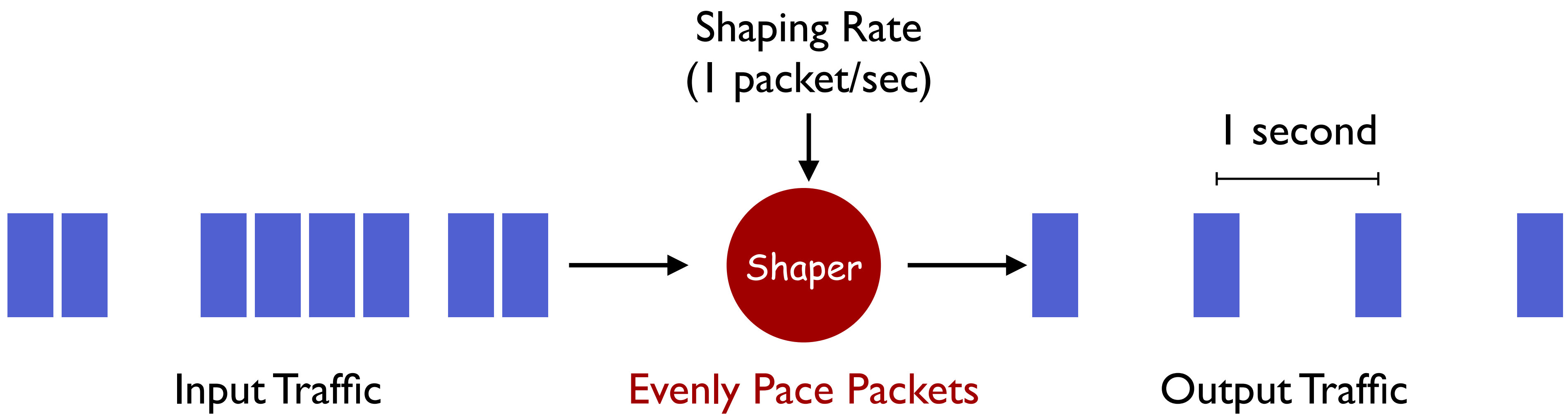
❶ Performance Isolation

- Throttle traffic rate

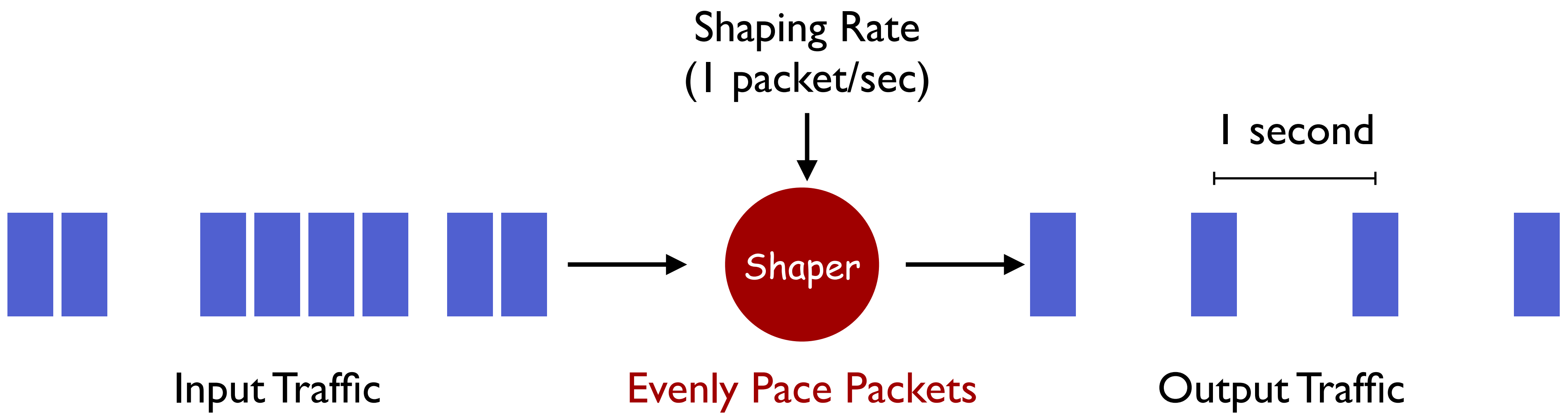
❷ Congestion Control

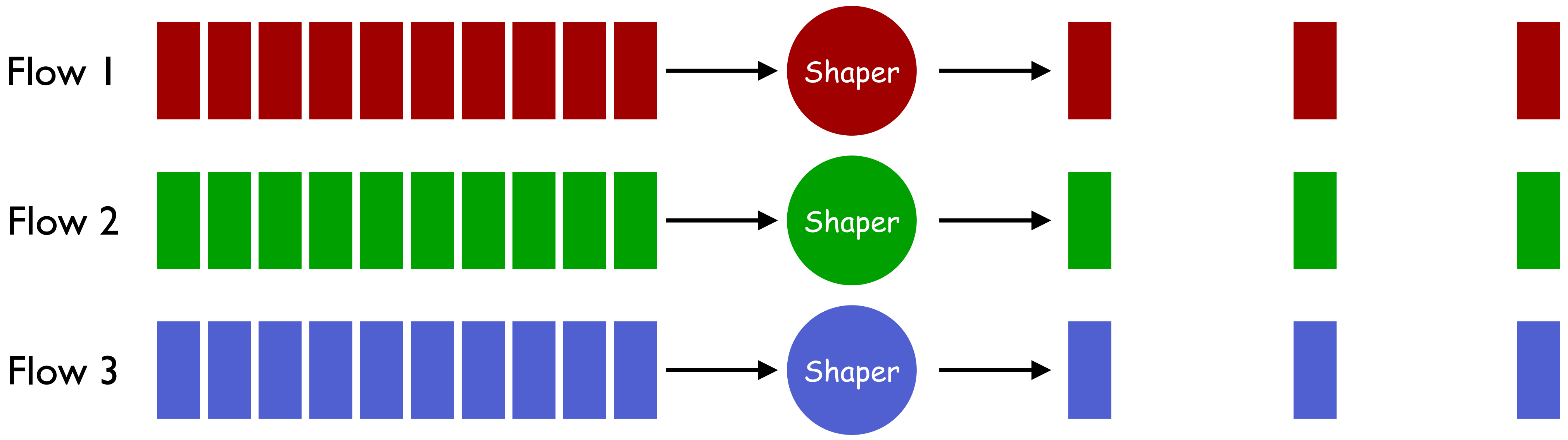
- Adjust sending rate
- Eliminate traffic bursts

Software Traffic Shaping



Software Traffic Shaping

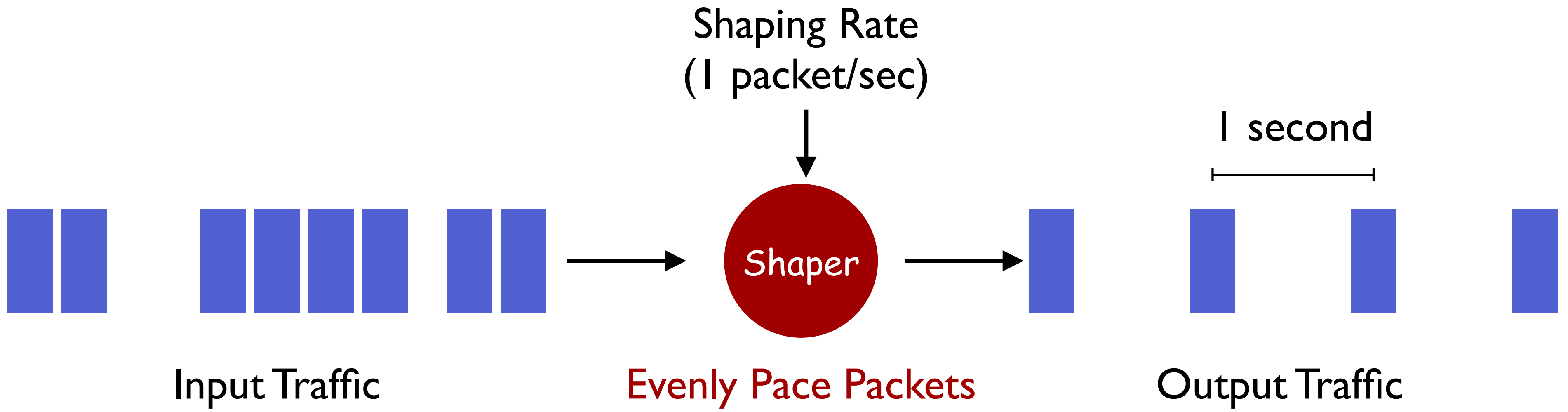


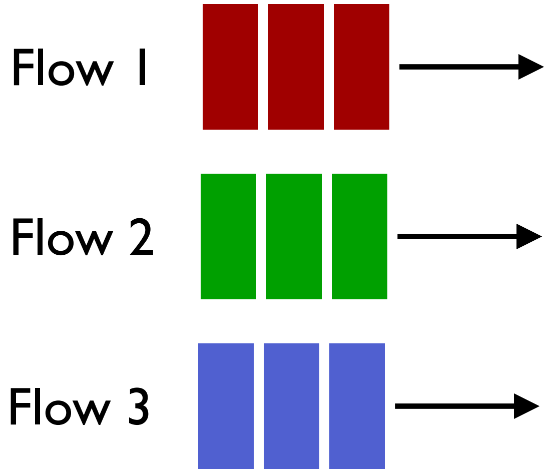


Traditional Traffic Shaper (e.g., tbf, htb)

- Each flow has a separate shaper
- High overhead with massive flows

Software Traffic Shaping

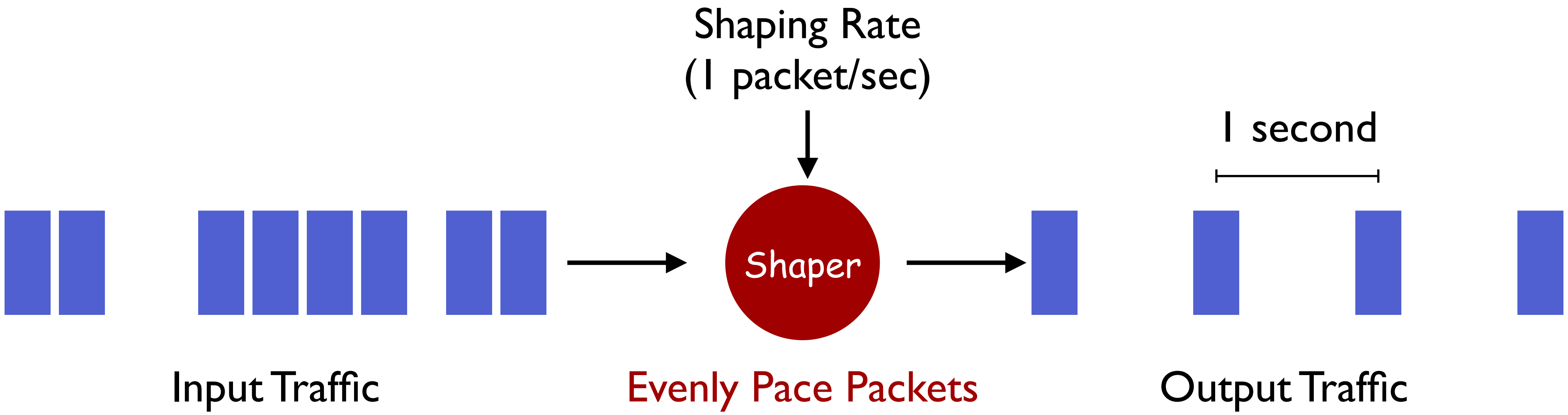


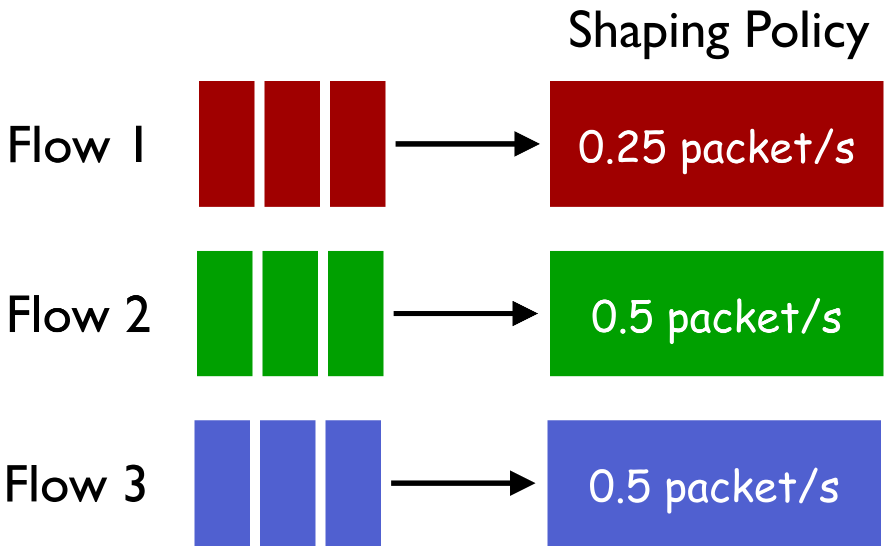


State-of-the-Art Traffic Shaper (Carousel[SIGCOMM'17], Eiffel[NSDI'19])

- Decouple the shaping policy and shaping enforcement
- Shape all flows with a single queue

Software Traffic Shaping

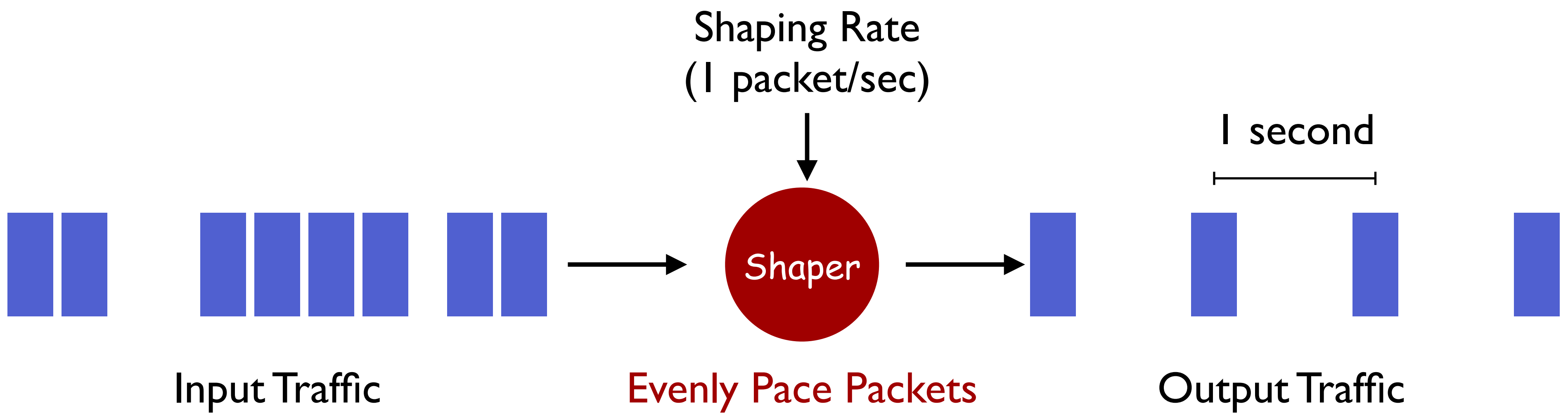


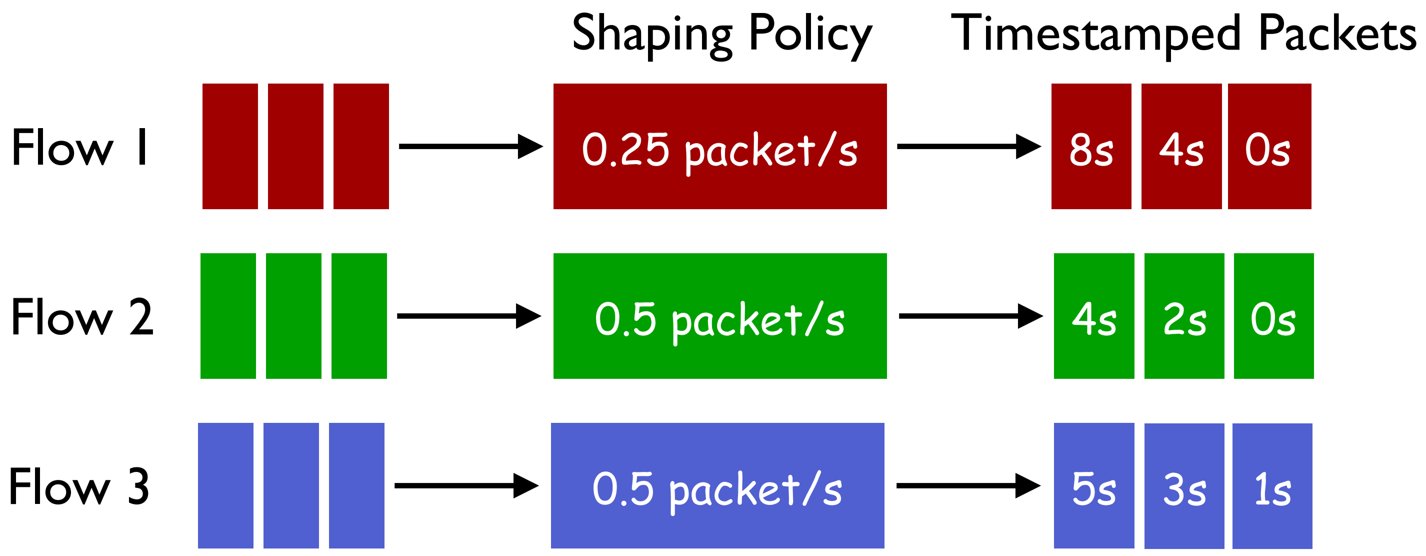


State-of-the-Art Traffic Shaper (Carousel[SIGCOMM'17], Eiffel[NSDI'19])

- Decouple the shaping policy and shaping enforcement
- Shape all flows with a single queue

Software Traffic Shaping



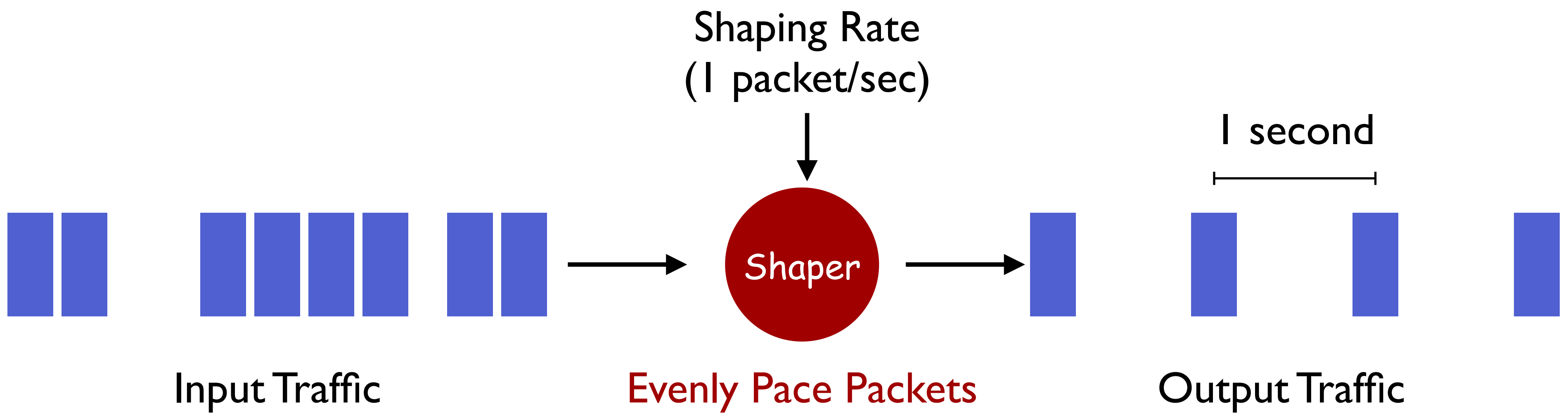


❶
Timestaming packets

State-of-the-Art Traffic Shaper (Carousel[SIGCOMM'17], Eiffel[NSDI'19])

- Decouple the shaping policy and shaping enforcement
- Shape all flows with a single queue

Software Traffic Shaping



Shaping Rate
(1 packet/sec)

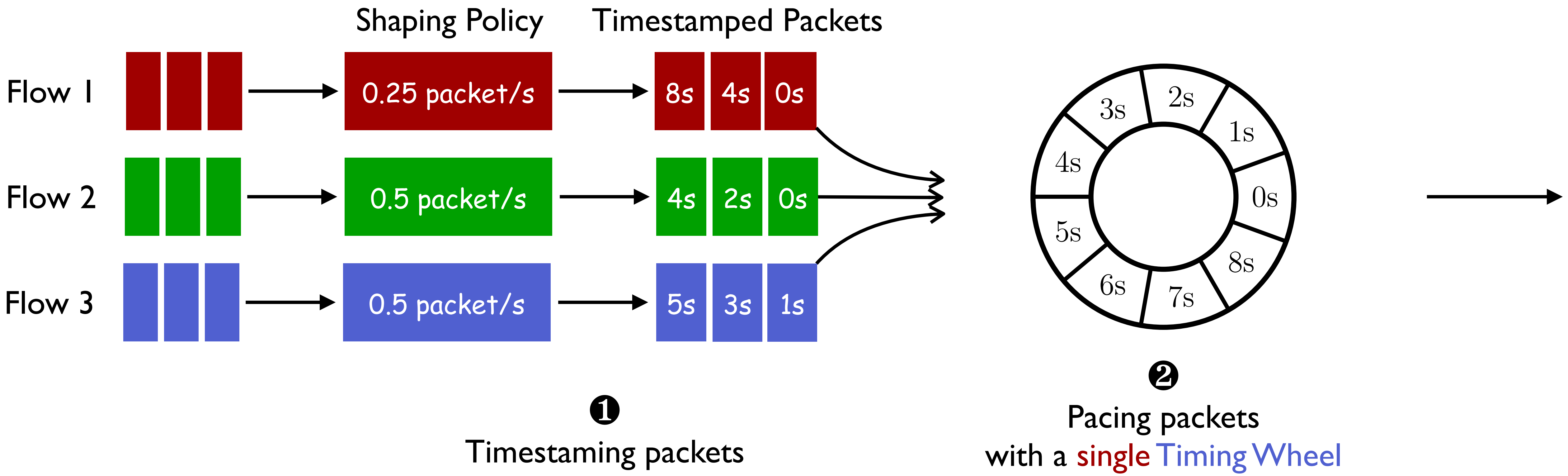
1 second

Shaper

Input Traffic

Evenly Pace Packets

Output Traffic



Shaping Policy

Timestamped Packets

Flow 1

0.25 packet/s

8s 4s 0s

Flow 2

0.5 packet/s

4s 2s 0s

Flow 3

0.5 packet/s

5s 3s 1s

3s 2s 1s 4s 0s 5s 8s 6s 7s

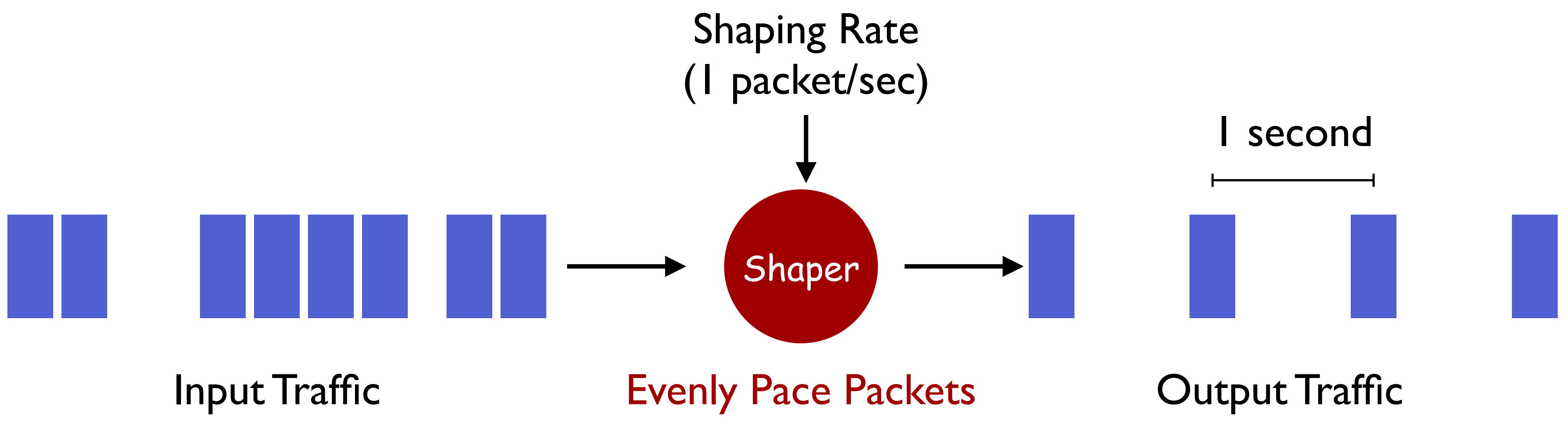
❶
Timestaming packets

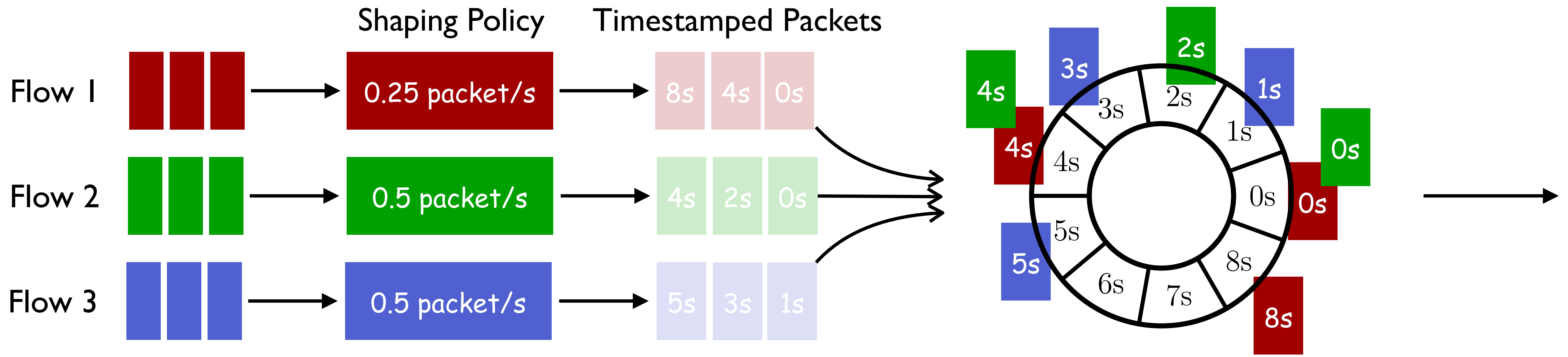
❷
Pacing packets
with a single Timing Wheel

State-of-the-Art Traffic Shaper (Carousel[SIGCOMM'17], Eiffel[NSDI'19])

- Decouple the shaping policy and shaping enforcement
- Shape all flows with a single queue

Software Traffic Shaping





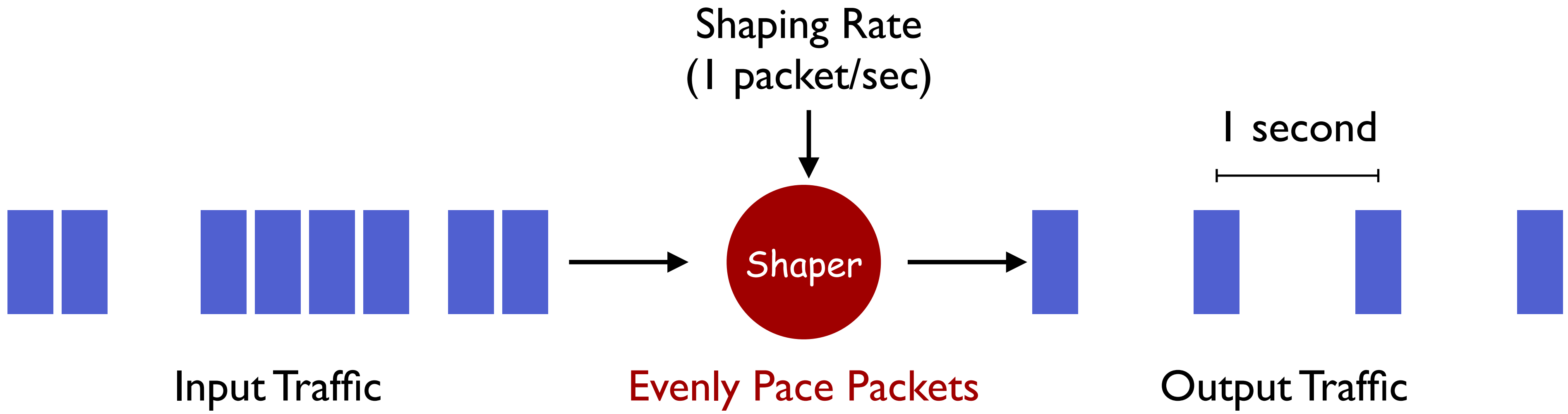
❶ Timestaming packets

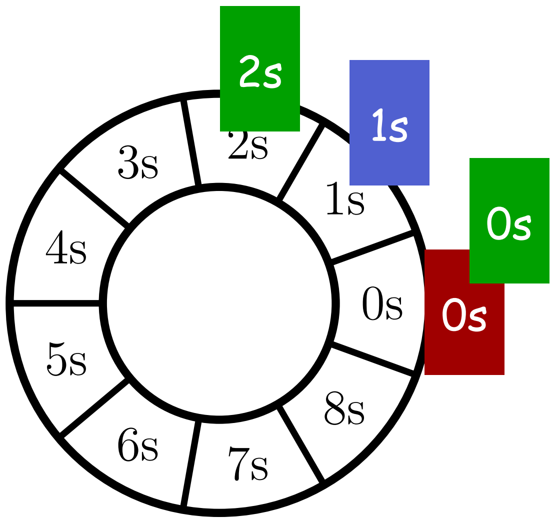
❷ Pacing packets with a single Timing Wheel

State-of-the-Art Traffic Shaper (Carousel[SIGCOMM'17], Eiffel[NSDI'19])

- Decouple the shaping policy and shaping enforcement
- Shape all flows with a single queue

Software Traffic Shaping

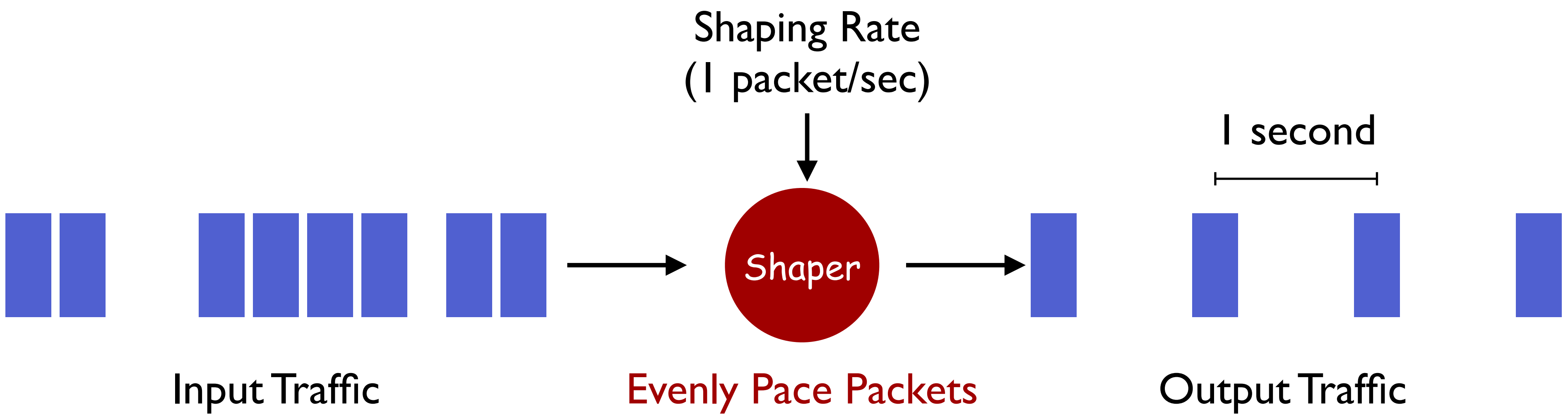




Current time = 0s

Pacing packets with Timing Wheel

Software Traffic Shaping



Shaping Rate
(1 packet/sec)

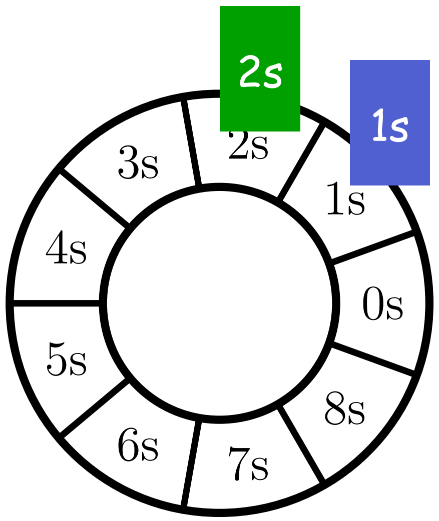
1 second

Shaper

Input Traffic

Evenly Pace Packets

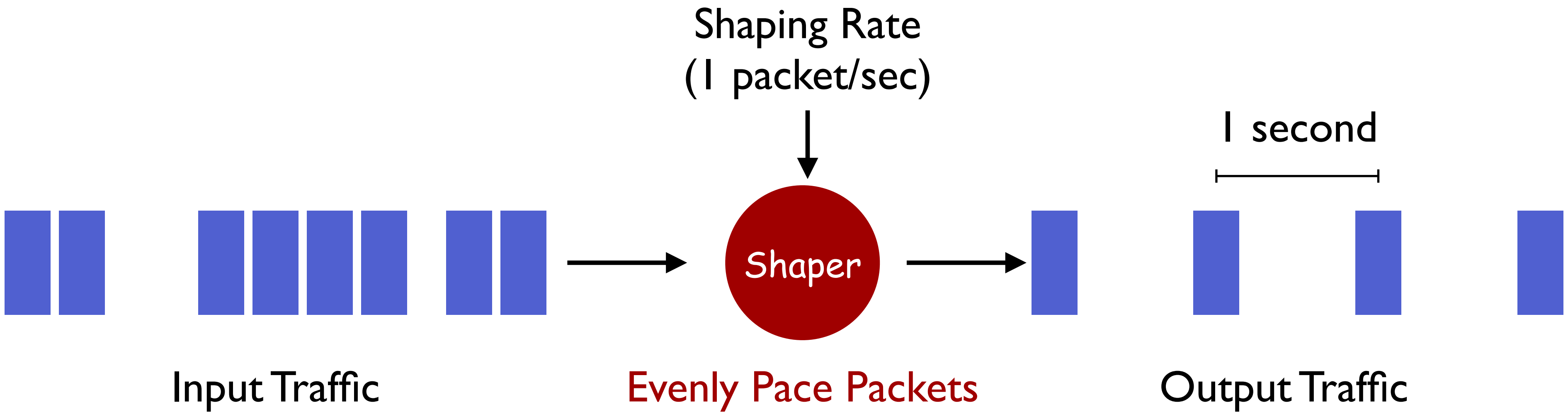
Output Traffic

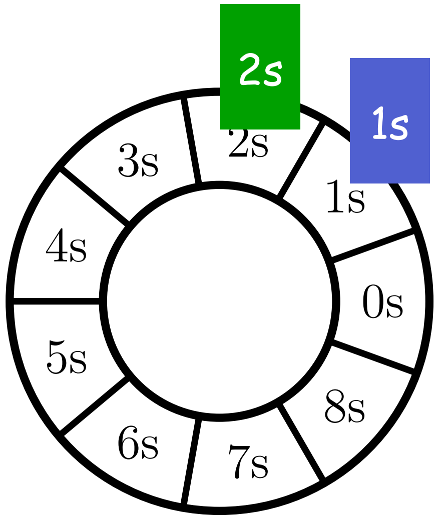


Current time = 0s

Pacing packets with Timing Wheel

Software Traffic Shaping

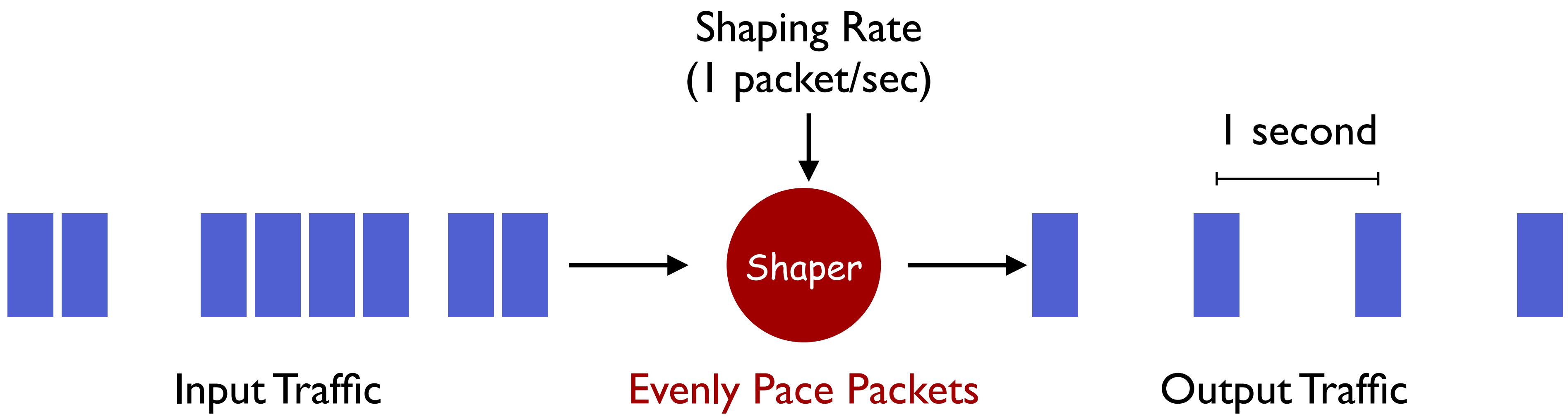




Current time = 1s

Pacing packets with Timing Wheel

Software Traffic Shaping



Shaping Rate
(1 packet/sec)

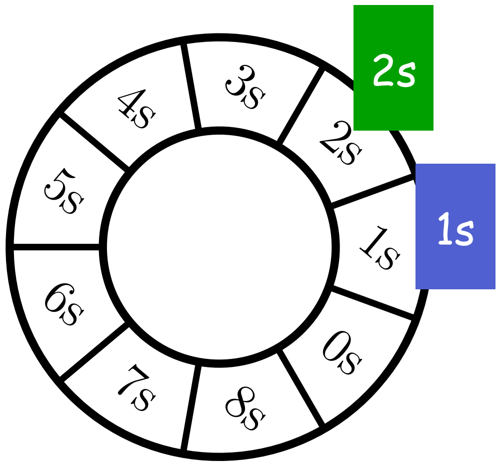
1 second

Shaper

Input Traffic

Evenly Pace Packets

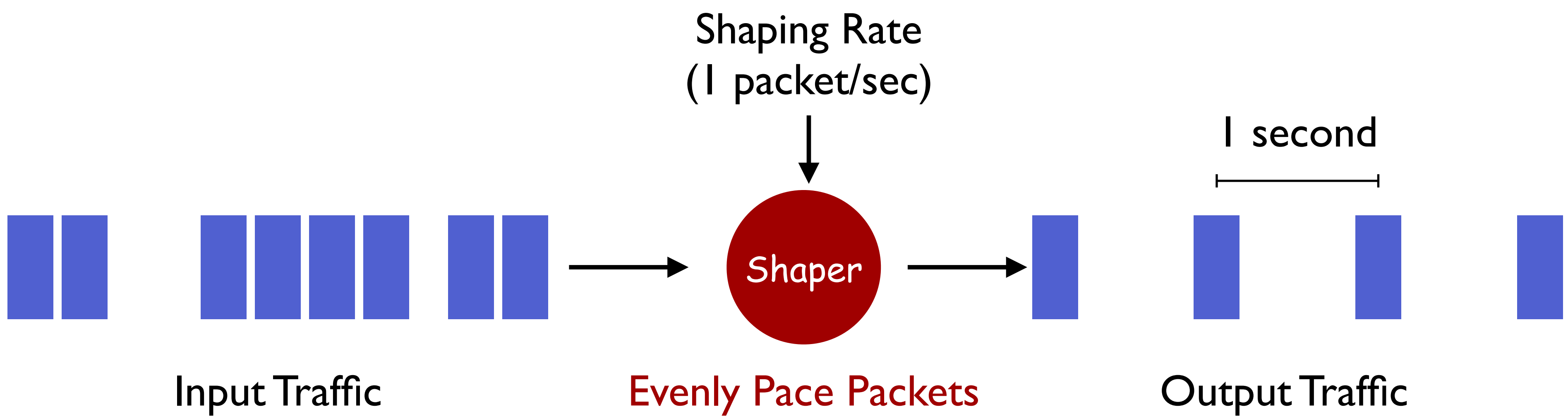
Output Traffic



Current time = 1s

Pacing packets with Timing Wheel

Software Traffic Shaping



Shaping Rate
(1 packet/sec)

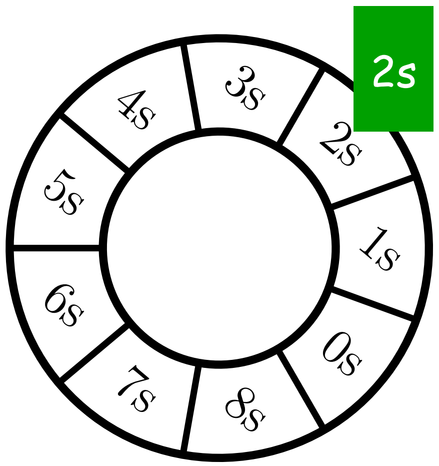
1 second

Shaper

Input Traffic

Evenly Pace Packets

Output Traffic



Current time = 1s

Pacing packets with Timing Wheel

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - ✓ Minimal queue maintenance overhead (i.e., One queue)
  - ✓ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory
    - Incur high overhead
    - Unable to achieve accurate shaping in 100Gbps network

Overhead of Software Traffic Shaping

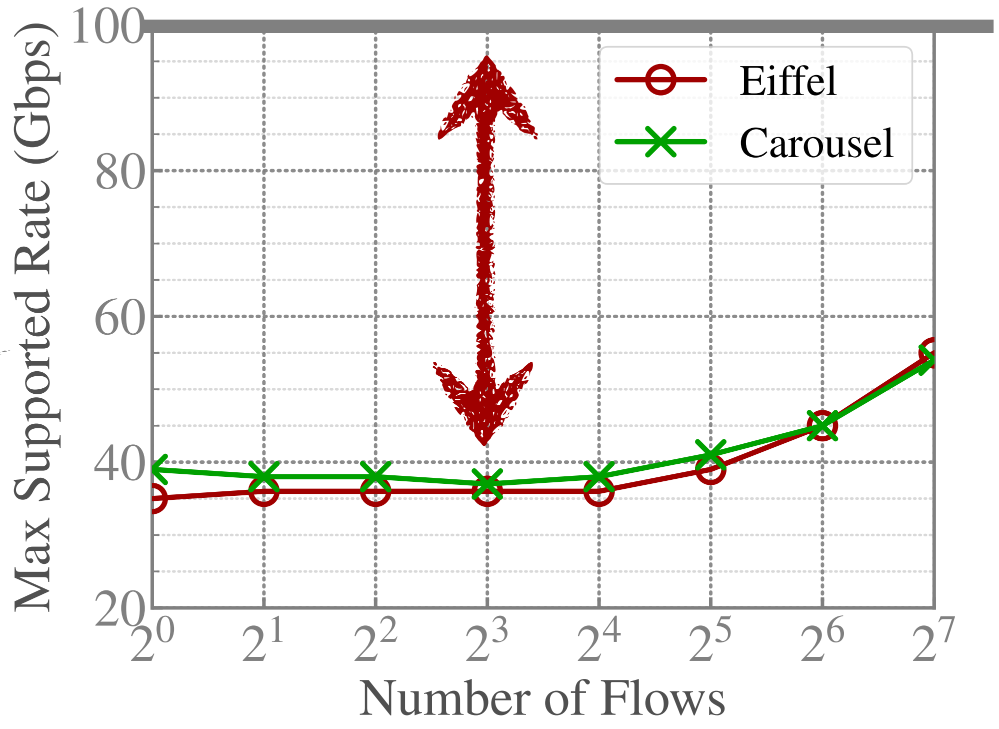
- State-of-the-art Shaper: Timestamping + Timing Wheel

  ✓ Minimal queue maintenance overhead (i.e., One queue)

  ✓ Minimal enqueue/dequeue overhead (i.e., O(1))

  ✗ Still unsatisfactory

  - Incur high overhead
  - Unable to achieve accurate shaping in 100Gbps network

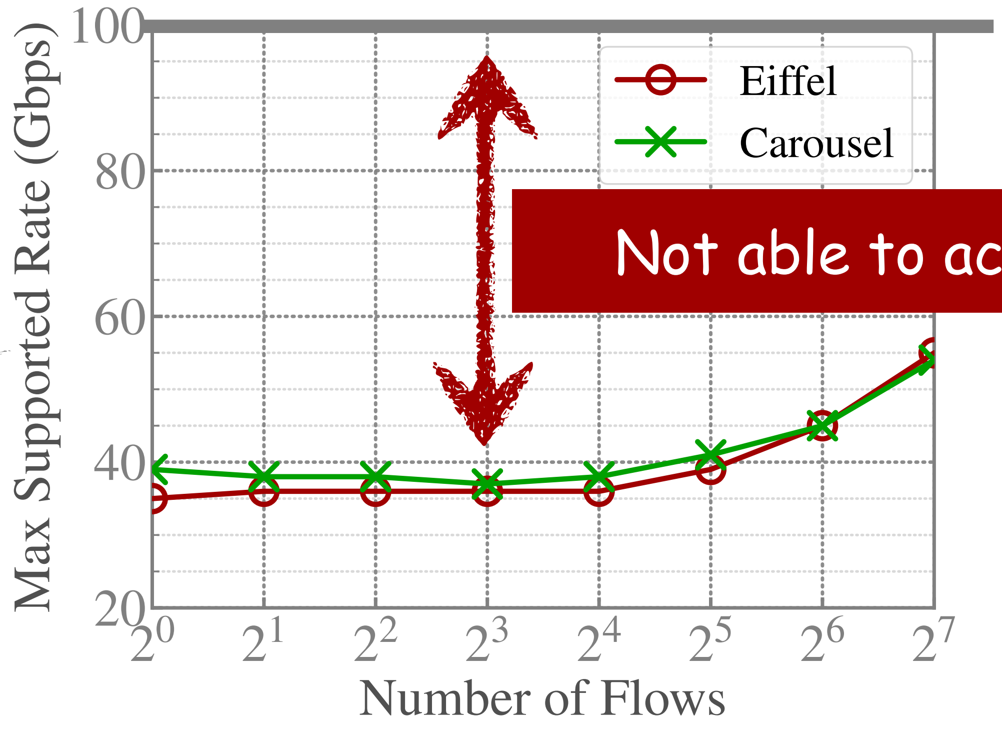


Network Capacity

maximum shaping rate with which a shaping scheme is able to achieve high shaping accuracy (i.e., shaping error < 1%).

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - ✓ Minimal queue maintenance overhead (i.e., One queue)
  - ✓ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory
    - Incur high overhead
    - Unable to achieve accurate shaping in 100Gbps network



Network Capacity

Not able to achieve high shaping speed

maximum shaping rate with which a shaping scheme is able to achieve high shaping accuracy (i.e., shaping error < 1%).

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - ✓ Minimal queue maintenance overhead (i.e., One queue)
  - ✓ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - ✓ Minimal queue maintenance overhead (i.e., One queue)
  - ✓ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - √ Minimal queue maintenance overhead (i.e., One queue)
  - √ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory

☹ Software traffic shaping has reached its limit

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - √ Minimal queue maintenance overhead (i.e., One queue)
  - √ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory

☹ Software traffic shaping has reached its limit

Is it?

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - ✓ Minimal queue maintenance overhead (i.e., One queue)
  - ✓ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory

☹ Software traffic shaping has reached its limit

Is it?

internal overhead

Overhead of Software Traffic Shaping

- State-of-the-art Shaper: Timestamping + Timing Wheel
  - √ Minimal queue maintenance overhead (i.e., One queue)
  - √ Minimal enqueue/dequeue overhead (i.e., O(1))
  - ✗ Still unsatisfactory

☹ Software traffic shaping has reached its limit

Is it?

internal overhead

Our observation:

It is the external overhead that hinders shaping from achieving higher speed

Overhead of Software Traffic Shaping

- What is external overhead?
    - Massive Software Interrupts
        - Wait for some time to send another packet
    - Per-packet PCIe operations

Overhead of Software Traffic Shaping

- What is external overhead?
  - Massive software Interrupts
    - Wait for some time to send another packet
  - Per-packet PCIe operations
    - 40Gbps rate for 1500B packets → PCIe write every 300ns
    - A separate PCIe write can take up to 900ns[1]

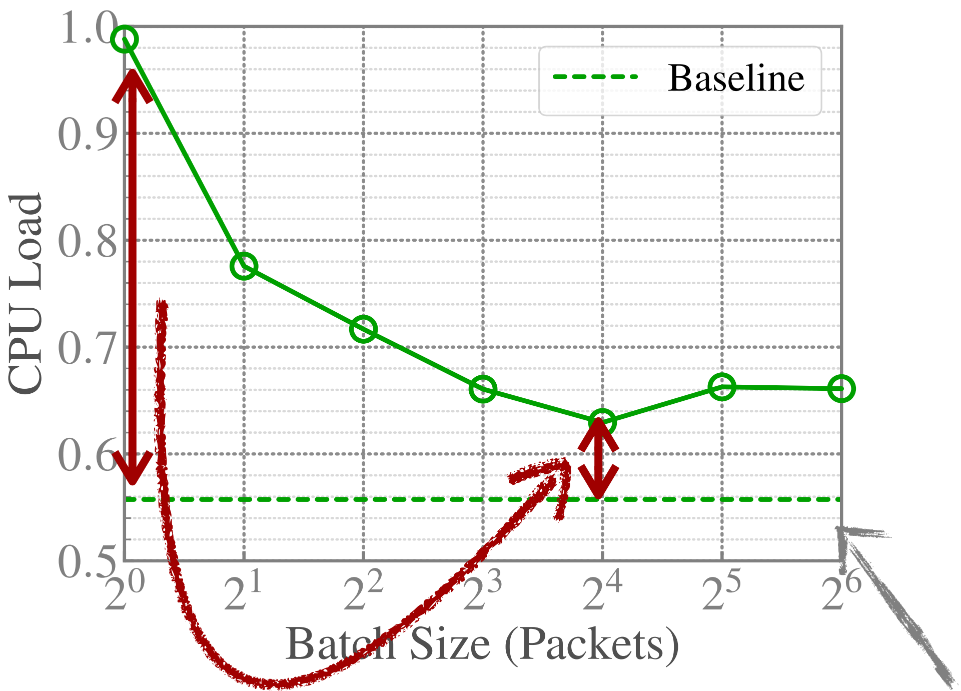
[1] B. Stephens, A. Akella, and M. Swift, "Loom: Flexible and Efficient NIC Packet Scheduling," in USENIX NSDI, 2019.

Overhead of Software Traffic Shaping

- What is external overhead?
  - Massive software Interrupts
  - Per-packet PCIe operations
- How to reduce external overhead?
  - Batching to amortize per-packet overhead

Overhead of Software Traffic Shaping

- What is external overhead?
  - Massive software Interrupts
  - Per-packet PCIe operations
- How to reduce external overhead?
  - Batching to amortize per-packet overhead



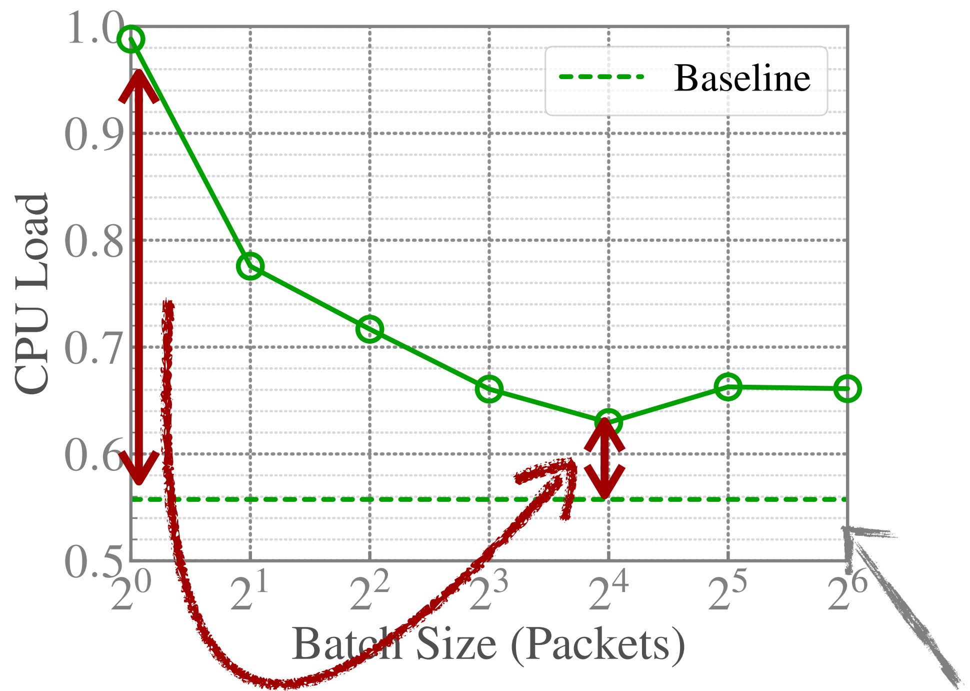
erhead of network stack

Reduce the CPU overhead by 5.9×

Kernel

Overhead of Software Traffic Shaping

- What is external overhead?
  - Massive software Interrupts
  - Per-packet PCIe operations
- How to reduce external overhead?
  - Batching to amortize per-packet overhead

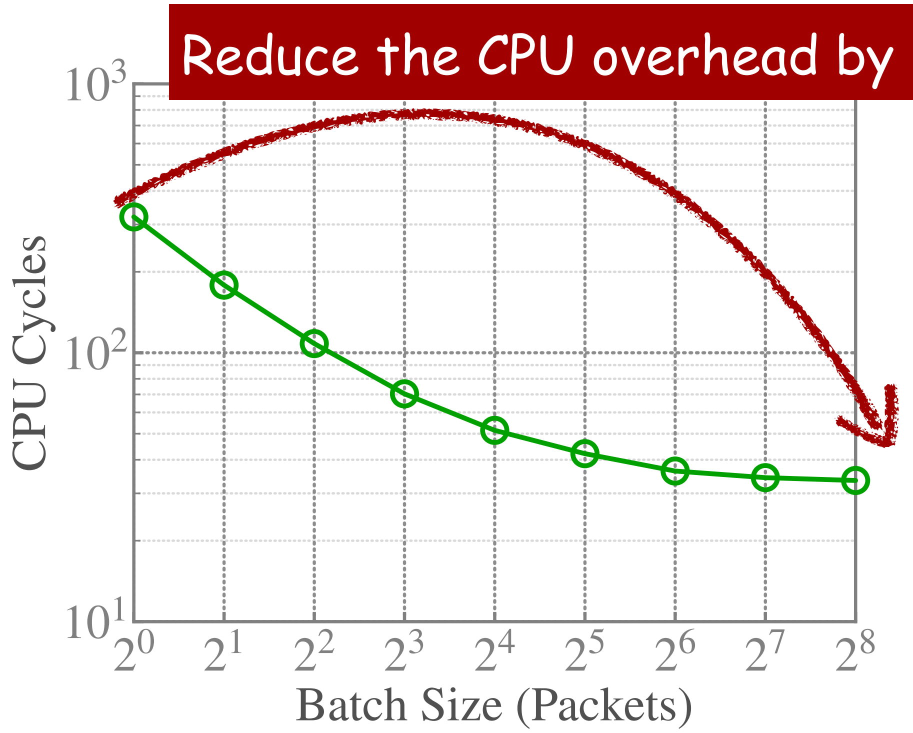


Reduce the CPU overhead by 5.9×

erhead of network stack

Kernel

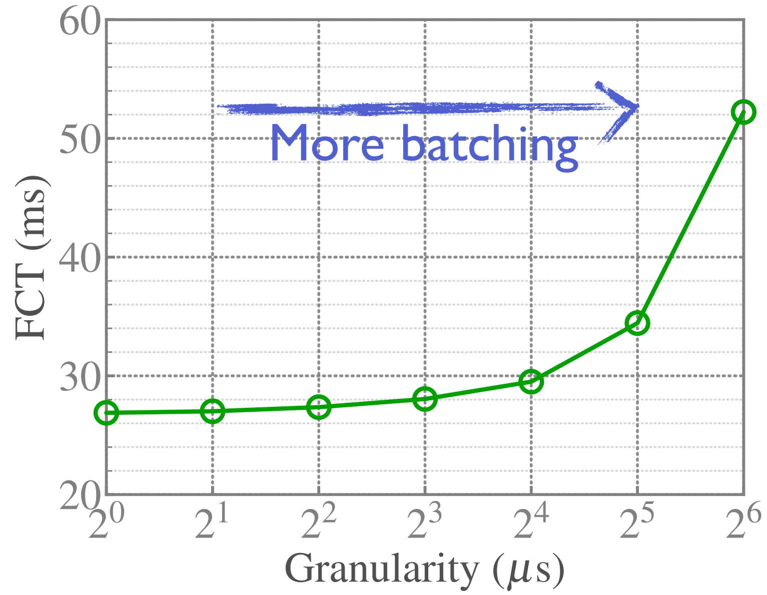
Reduce the CPU overhead by 9.5×



DPDK (w/o interrupt)

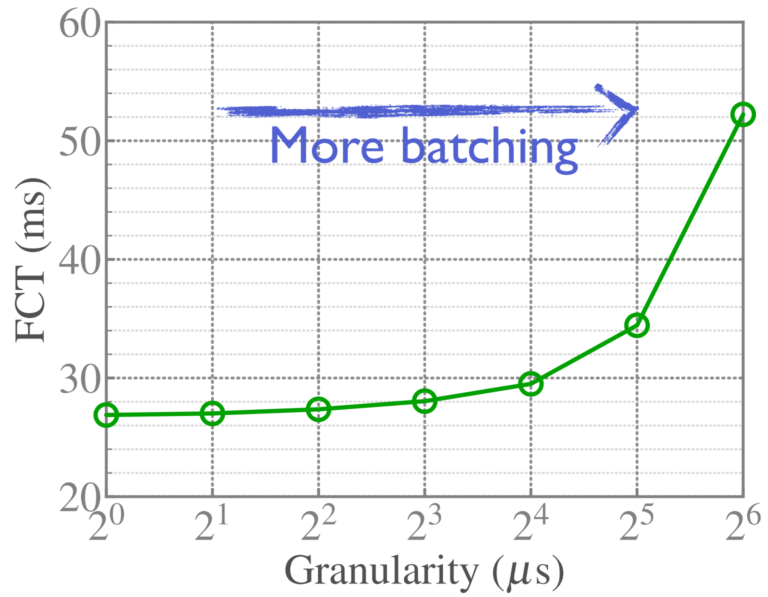
Overhead of Software Traffic Shaping

- What is external overhead?
- How to reduce external overhead?
  - Batching to amortize per-packet overhead
- Why not using batching?
  - Batching results in traffic bursts
  - Traffic bursts can degrade transmission performance



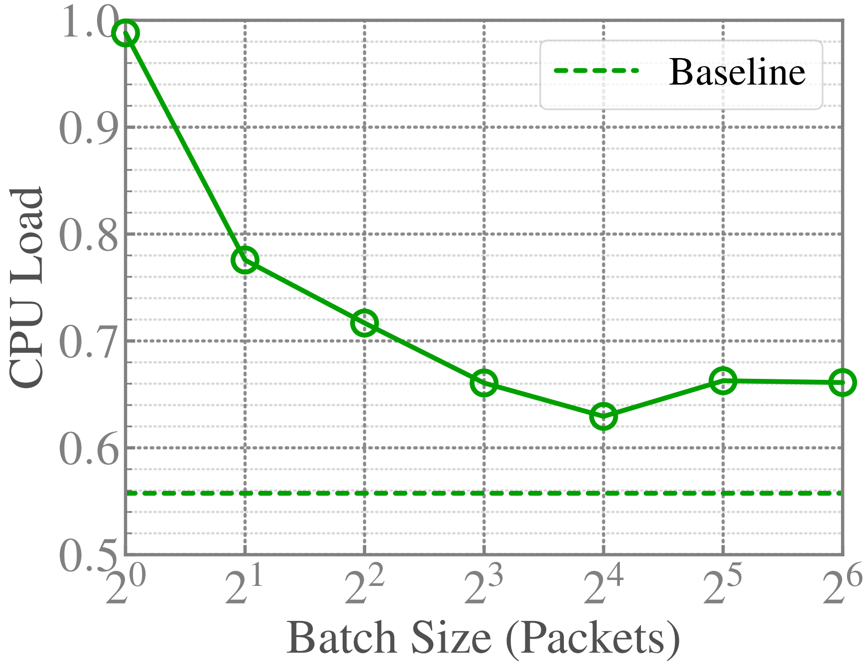
Overhead of Software Traffic Shaping

- What is external overhead?
- How to reduce external overhead?
  - Batching to amortize per-packet overhead
- Why not using batching?
  - Batching results in traffic bursts
  - Traffic bursts can degrade transmission performance



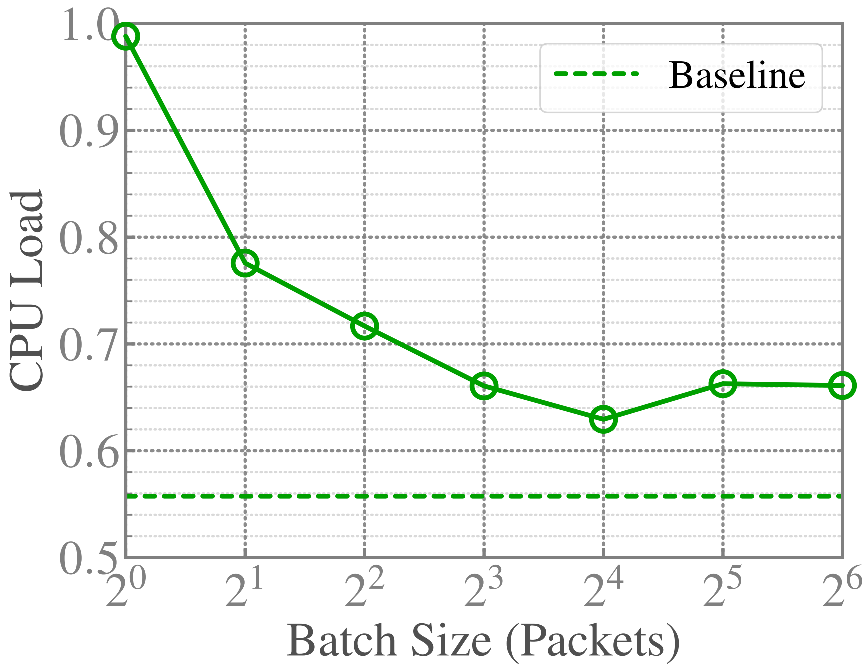
Batching can extend the FCT by ~2×

Dilemma of batching

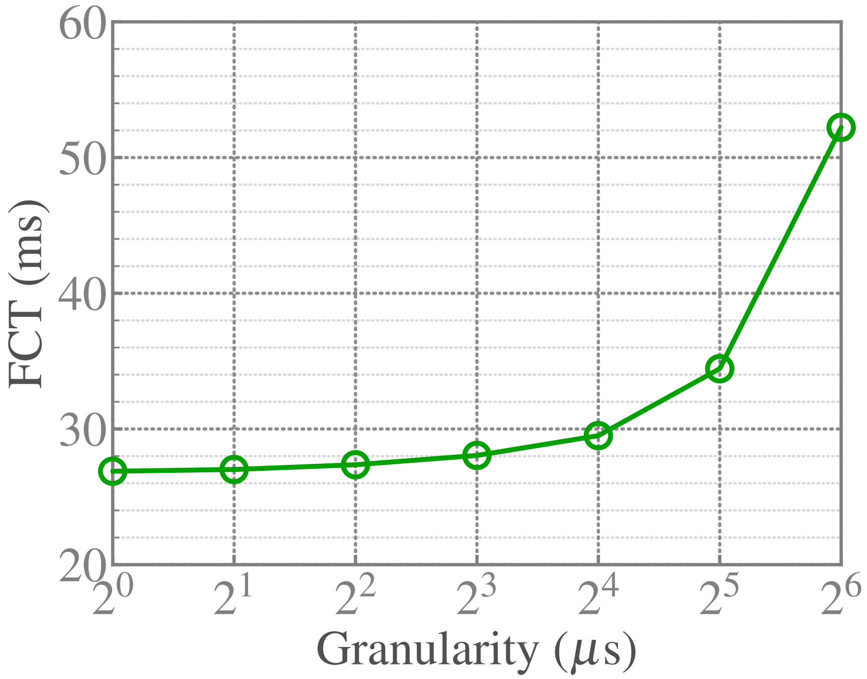


Reduce the CPU overhead by 5.9×

Dilemma of batching

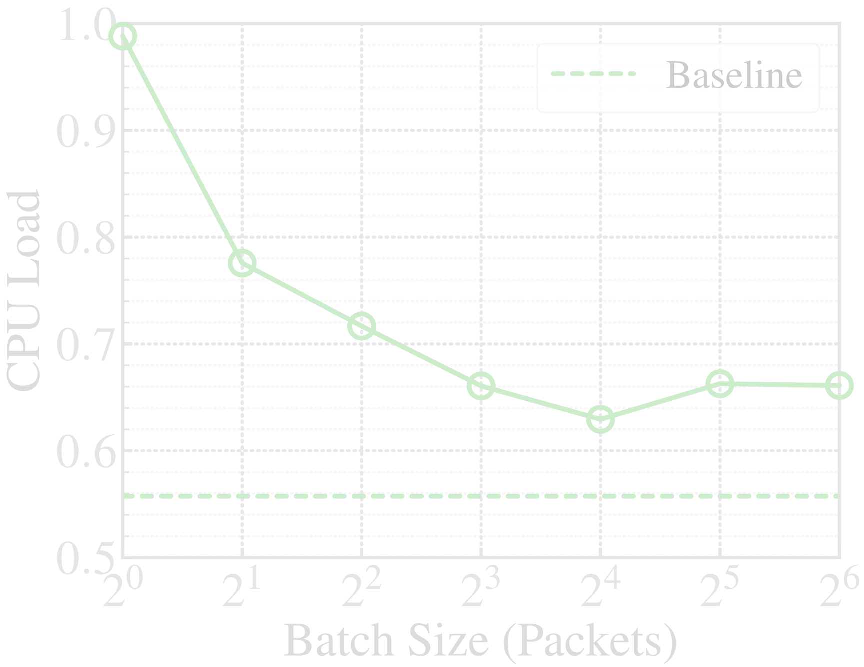


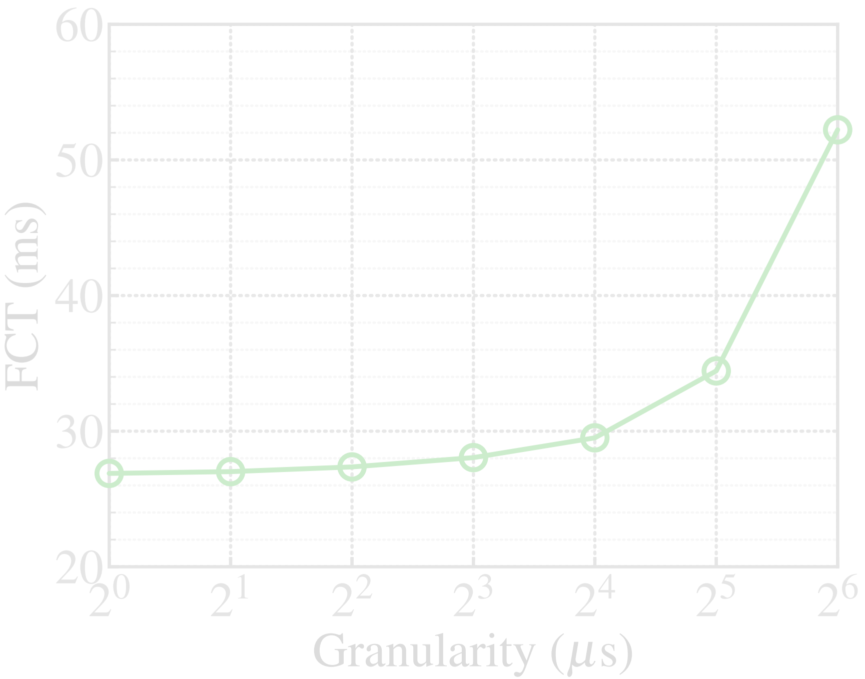
Reduce the CPU overhead by 5.9×



Extend the FCT by ~2×

Dilemma of batching





Can we achieve the best of both worlds?

Dilemma of batching

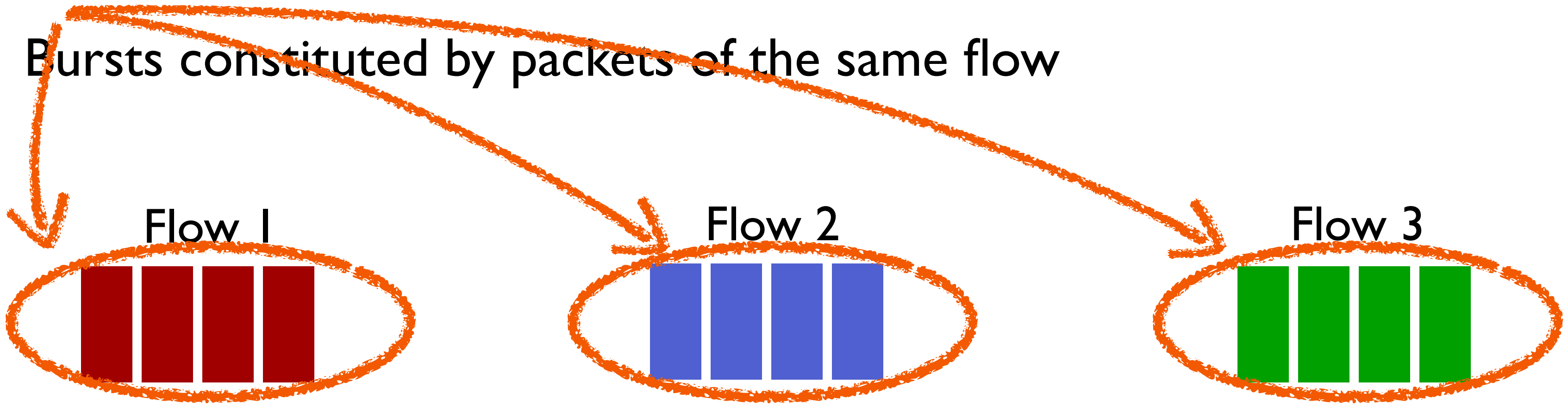
Our Insight

- Intra-flow burst is to blame
  - Bursts constituted by packets of the same flow

- Inter-flow burst can be *demultiplexed* before congestion point

Our Insight

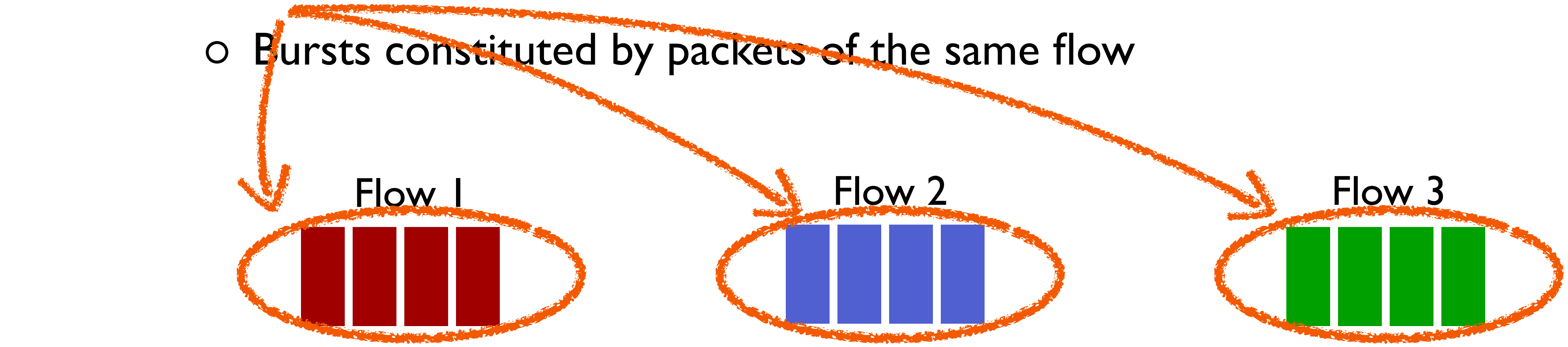
- Intra-flow burst is to blame
  - Bursts constituted by packets of the same flow



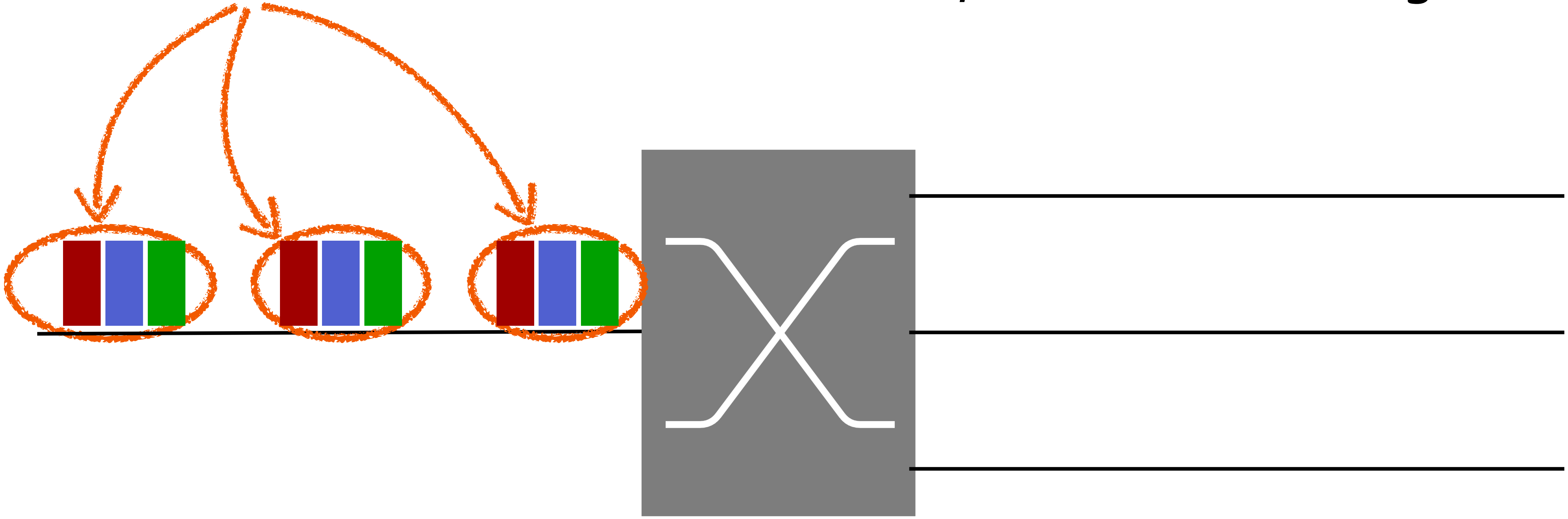
- Inter-flow burst can be *demultiplexed* before congestion point

Our Insight

- Intra-flow burst is to blame
  - Bursts constituted by packets of the same flow

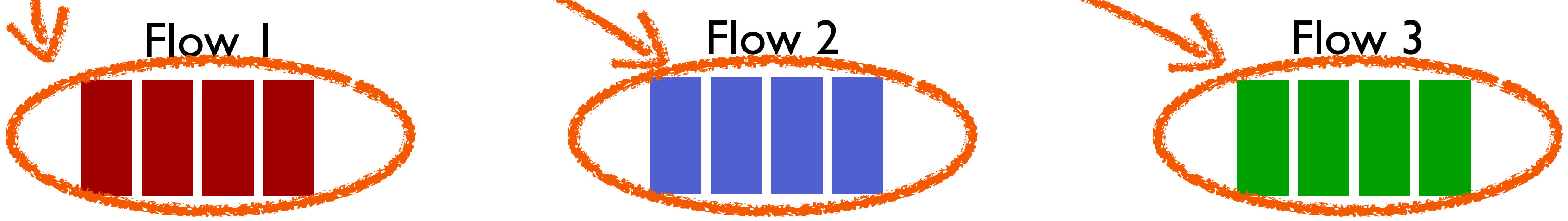


- Inter-flow burst can be *demultiplexed* before congestion point

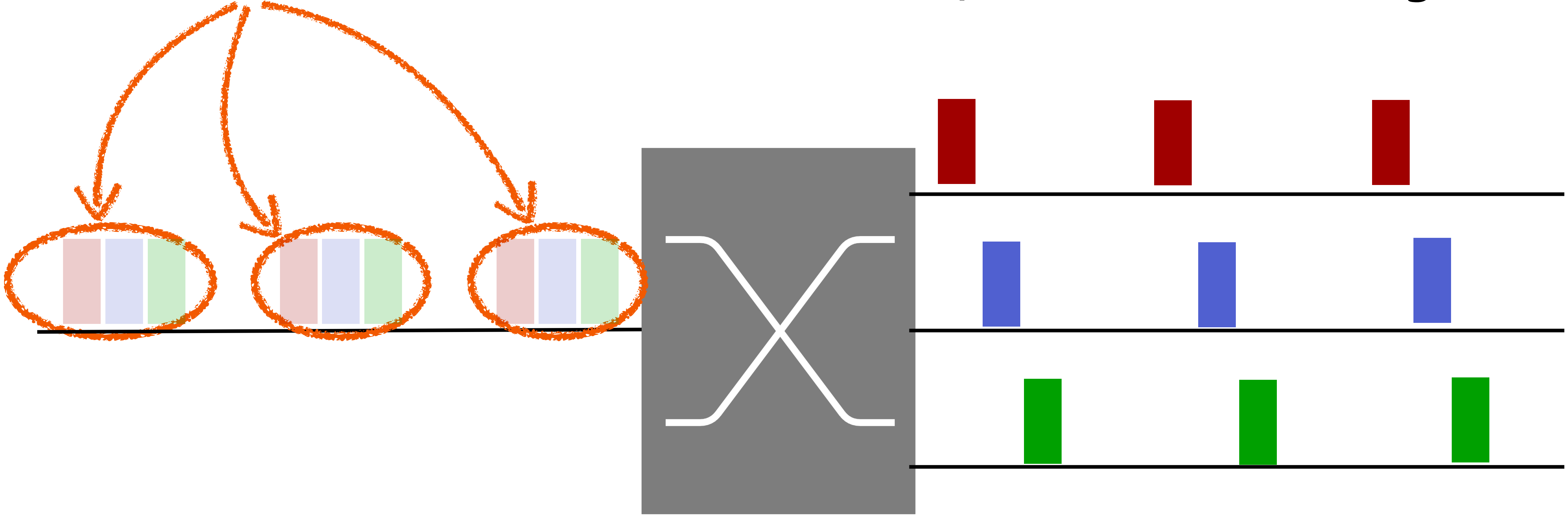


Our Insight

- Intra-flow burst is to blame
  - Bursts constituted by packets of the same flow



- Inter-flow burst can be *demultiplexed* before congestion point



Our Insight

- Inter-flow burst can be *demultiplexed* before congestion point
  - Different flows from a host tend to have different routes
    - Most traffic is inner-rack
      - 75.7% of Hadoop traffic is destined to servers in the the same rack[SIGCOMM'17 Facebook]
      - 80% of cloud data center traffic stays within a rack[IMC'10 Microsoft]
    - Inter-rack traffic: ECMP
  - Most congestion occurs at the last hops[SIGCOMM'15 Google, IMC'17 Facebook]

Summary of Observations

- Batching is essential to achieve fast software traffic shaping on high-speed networks

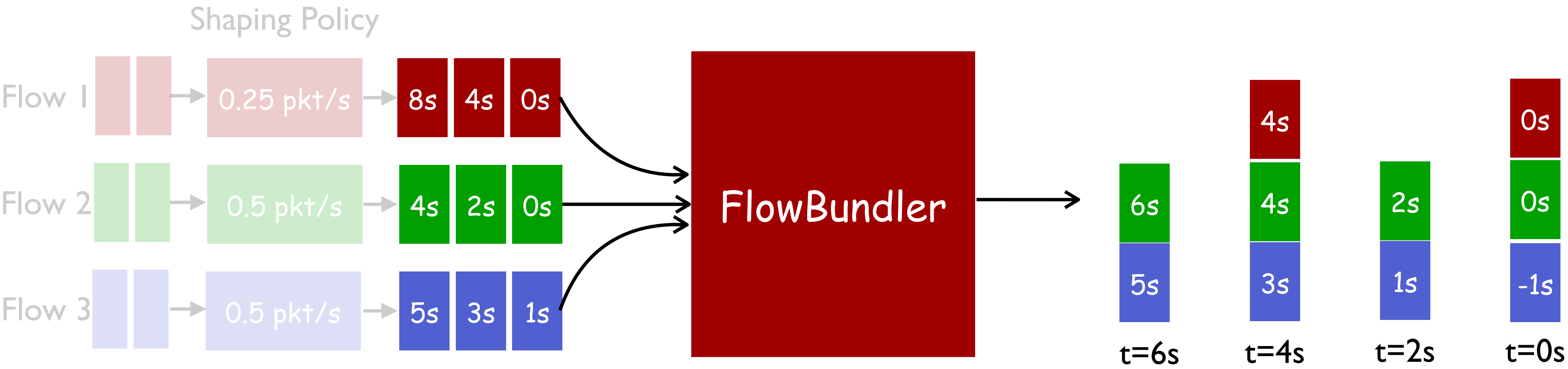
+

- Traffic shaping *only* needs to eliminate intra-flow bursts

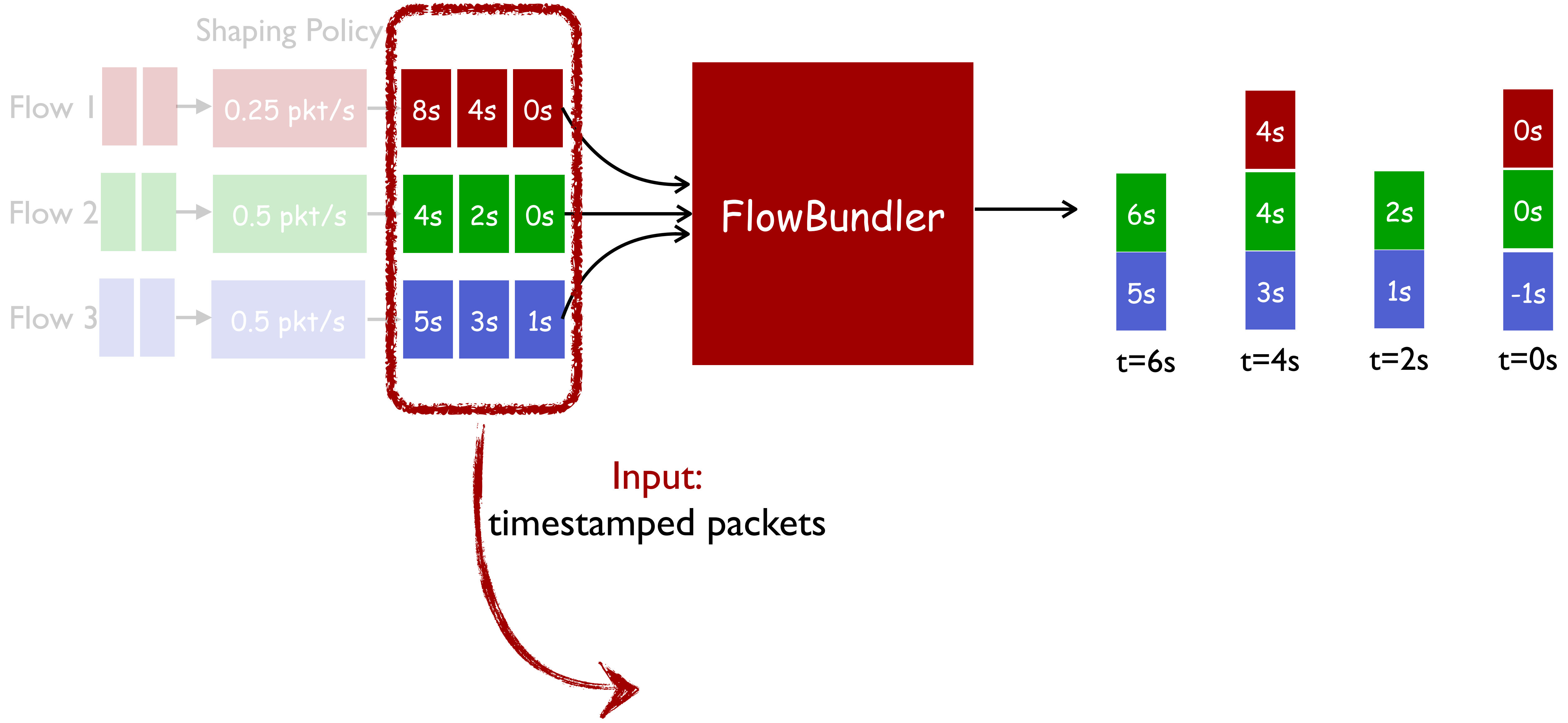
↓

- Traffic shaping can utilize inter-flow batching to reduce CPU overhead

Our Approach: FlowBundler

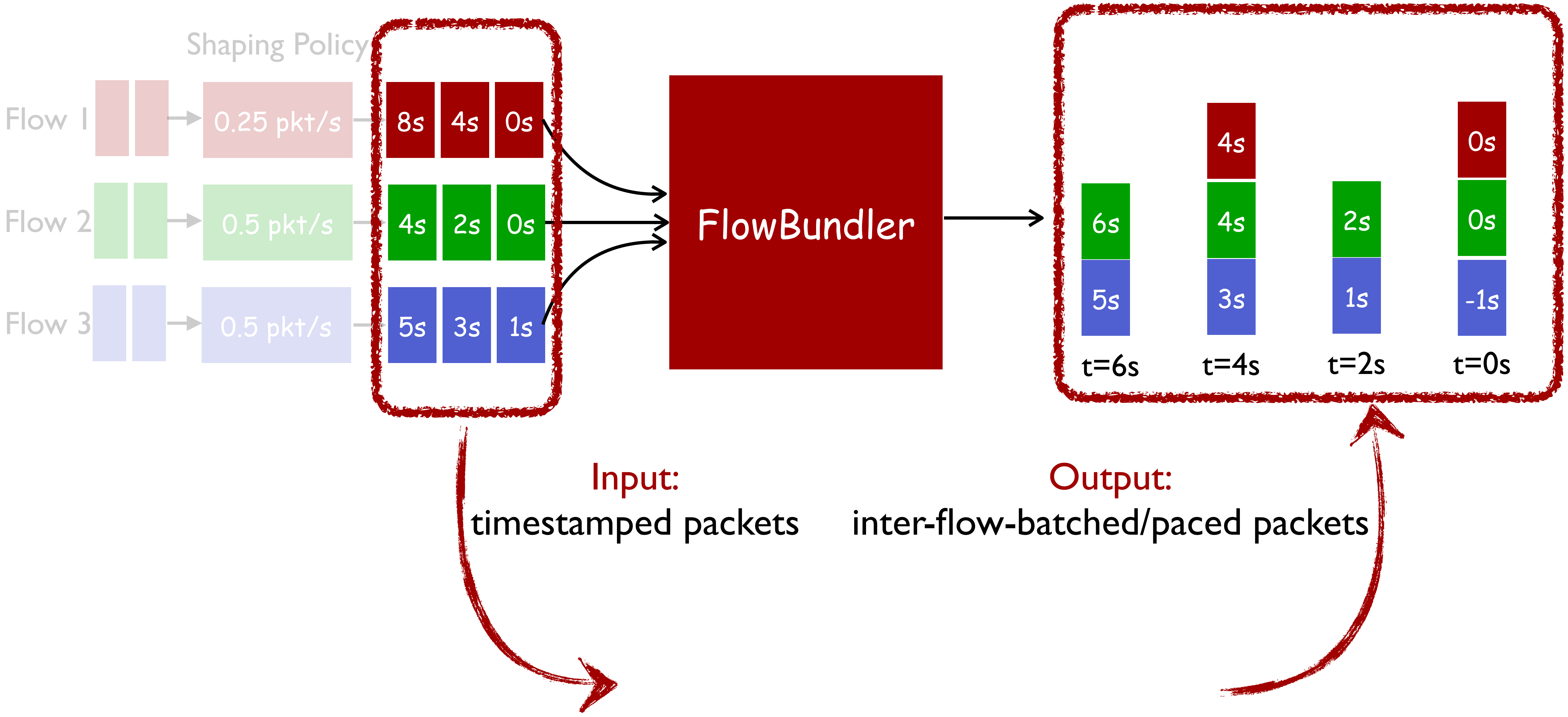


Our Approach: FlowBundler



Input:
timestamped packets

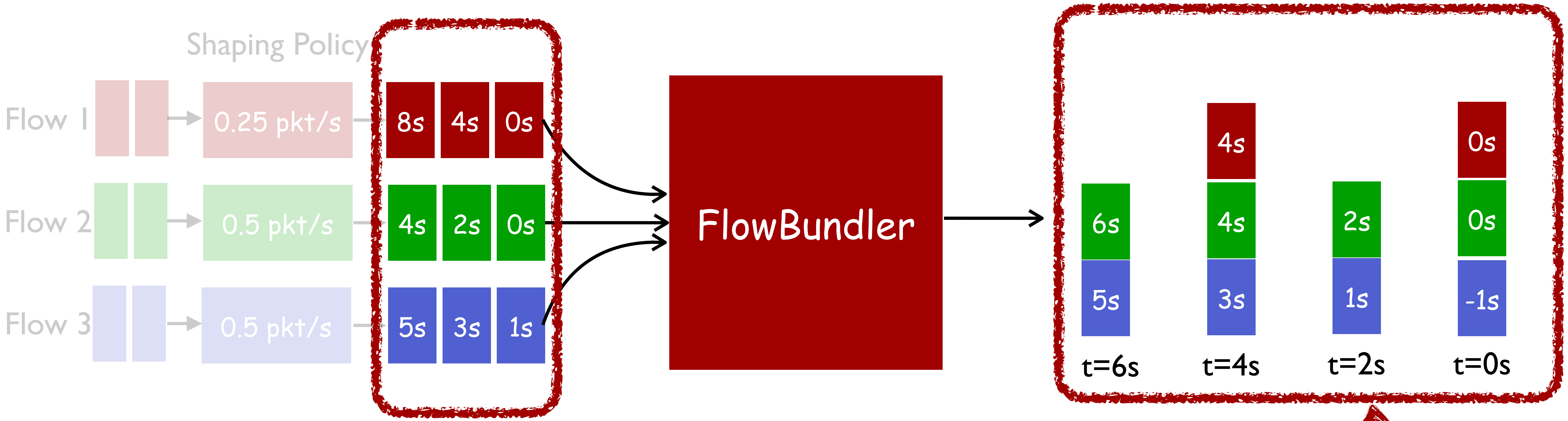
Our Approach: FlowBundler



Input:
timestamped packets

Output:
inter-flow-batched/paced packets

Our Approach: FlowBundler



Input:
timestamped packets

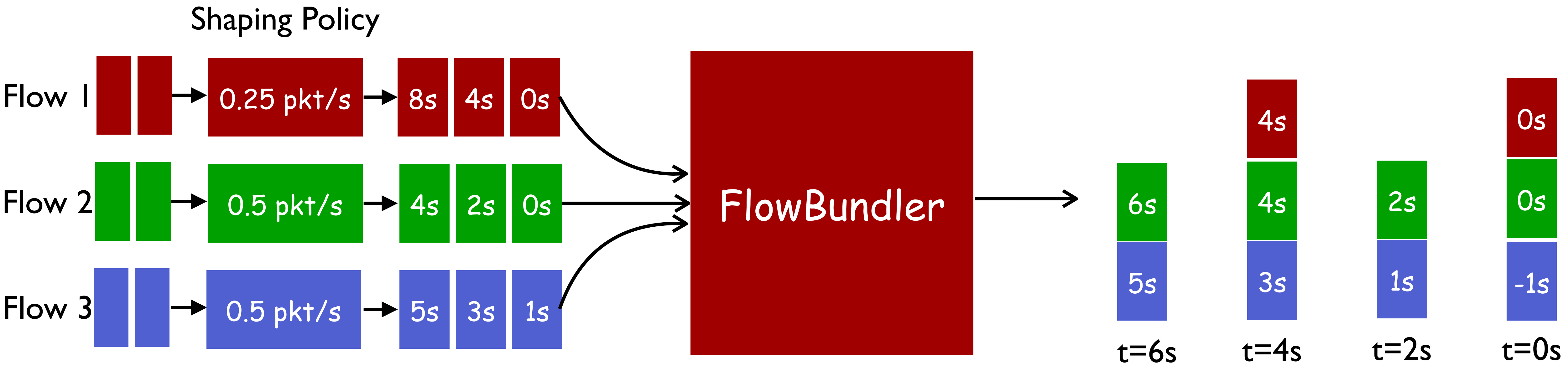
Output:
inter-flow-batched/paced packets

Goal of FlowBundler:

- Batching inter-flow packets
- Eliminating inner-flow bursts
- O(1) enqueue and dequeue time

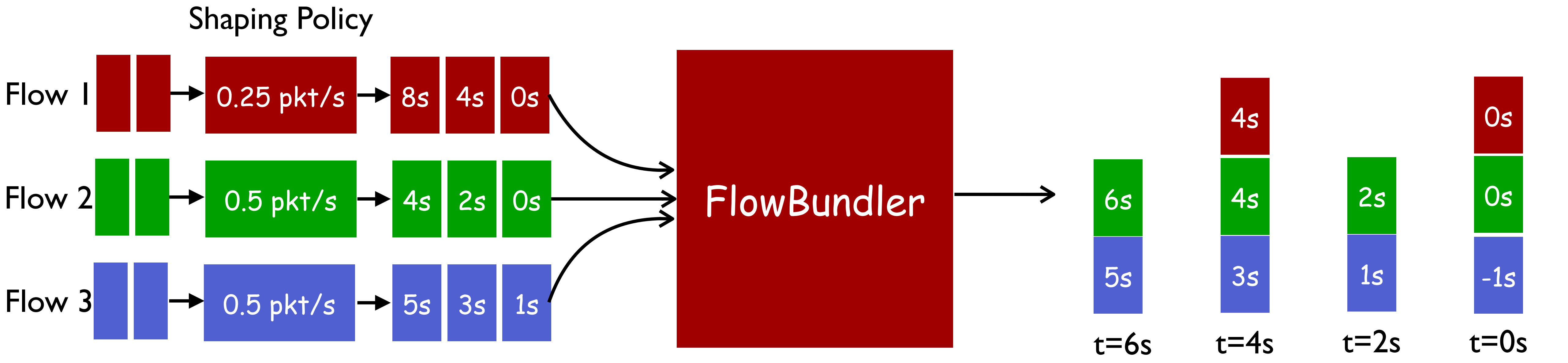
Challenge

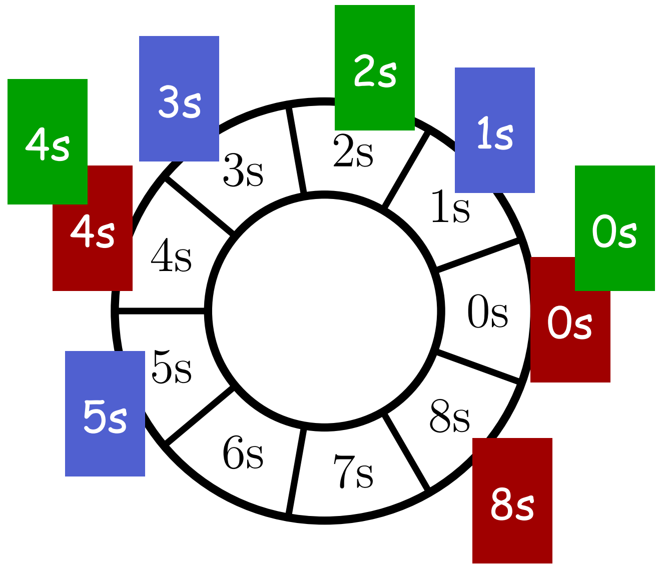
Question: How to efficiently place and extract inter-flow-batched packets



Challenge

Question: How to efficiently place and extract inter-flow-batched packets



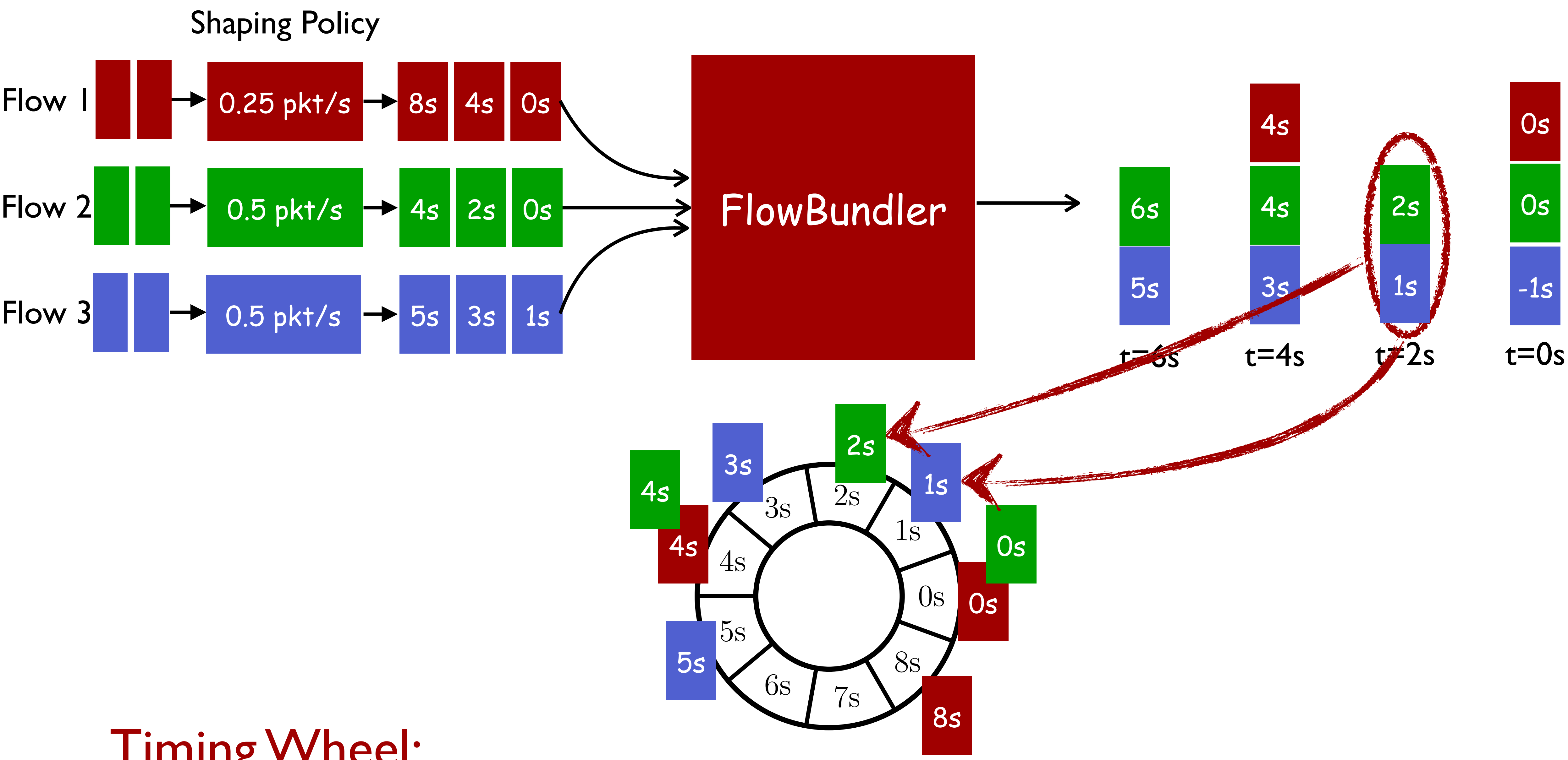


Timing Wheel:

✗ CPU inefficient: Multiple dequeue operations for a single batch

Challenge

Question: How to efficiently place and extract inter-flow-batched packets

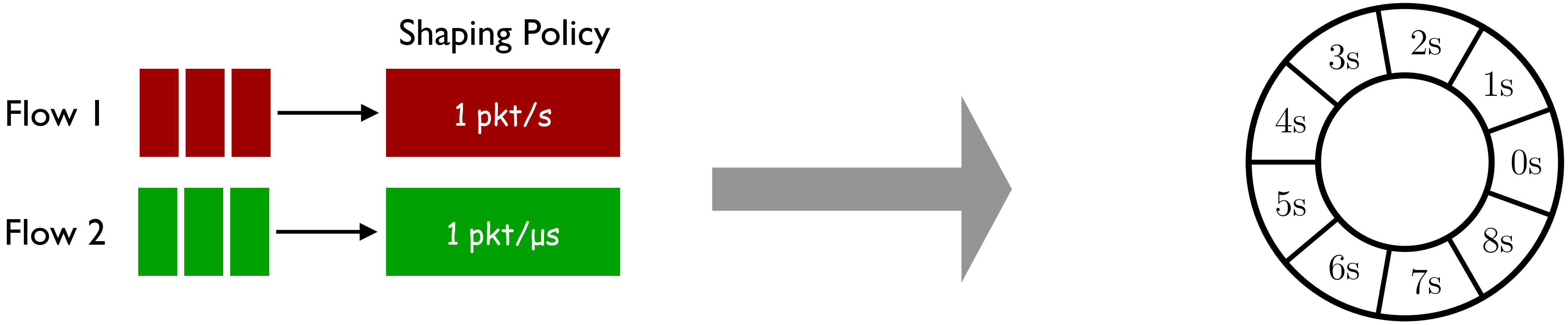


Timing Wheel:

✗ CPU inefficient: Multiple dequeue operations for a single batch

Challenge

Question: How to efficiently place and dequeue inter-flow-batched packets



Granularity: 1s
# Slots: 9

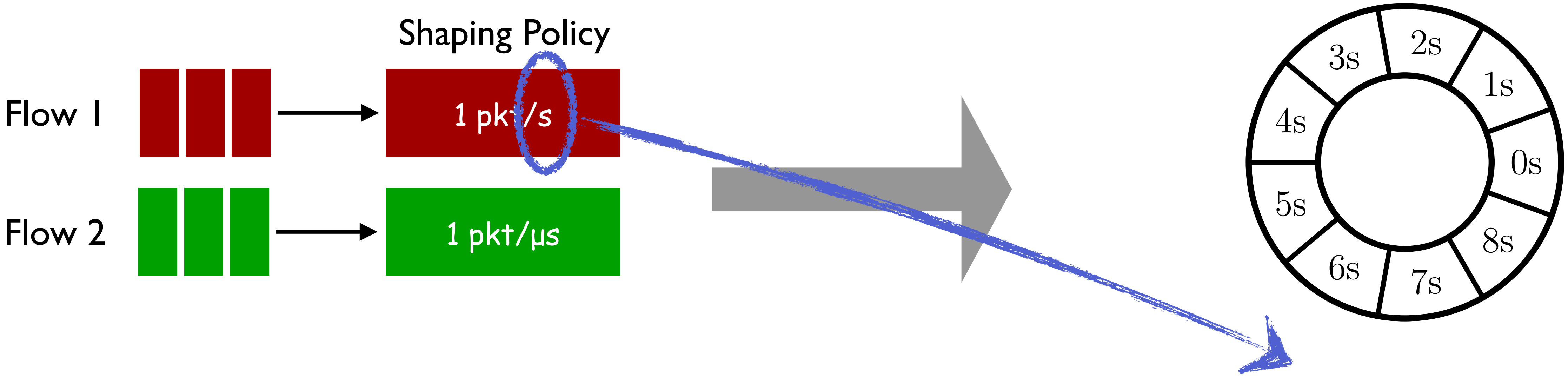
Timing Wheel:

✗ CPU inefficient: Multiple dequeue operations for a single batch

✗ Memory inefficient: Huge memory requirement

Challenge

Question: How to efficiently place and dequeue inter-flow-batched packets



Granularity: 1s
# Slots: 9

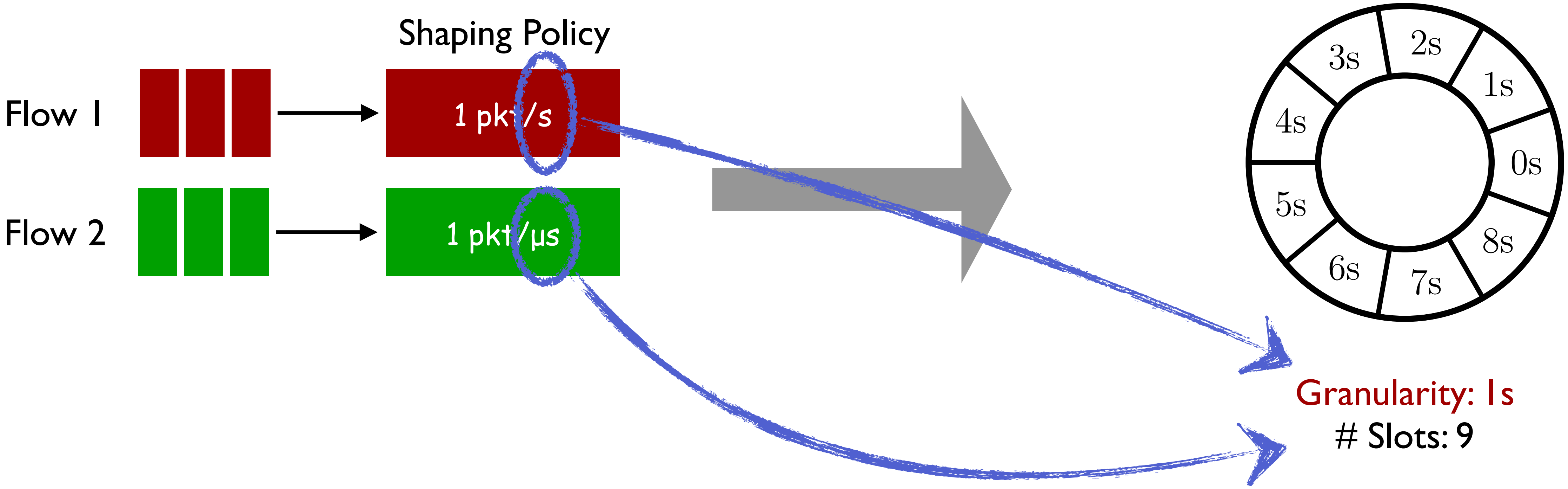
Timing Wheel:

✗ CPU inefficient: Multiple dequeue operations for a single batch

✗ Memory inefficient: Huge memory requirement

Challenge

Question: How to efficiently place and dequeue inter-flow-batched packets



Granularity: 1s
# Slots: 9

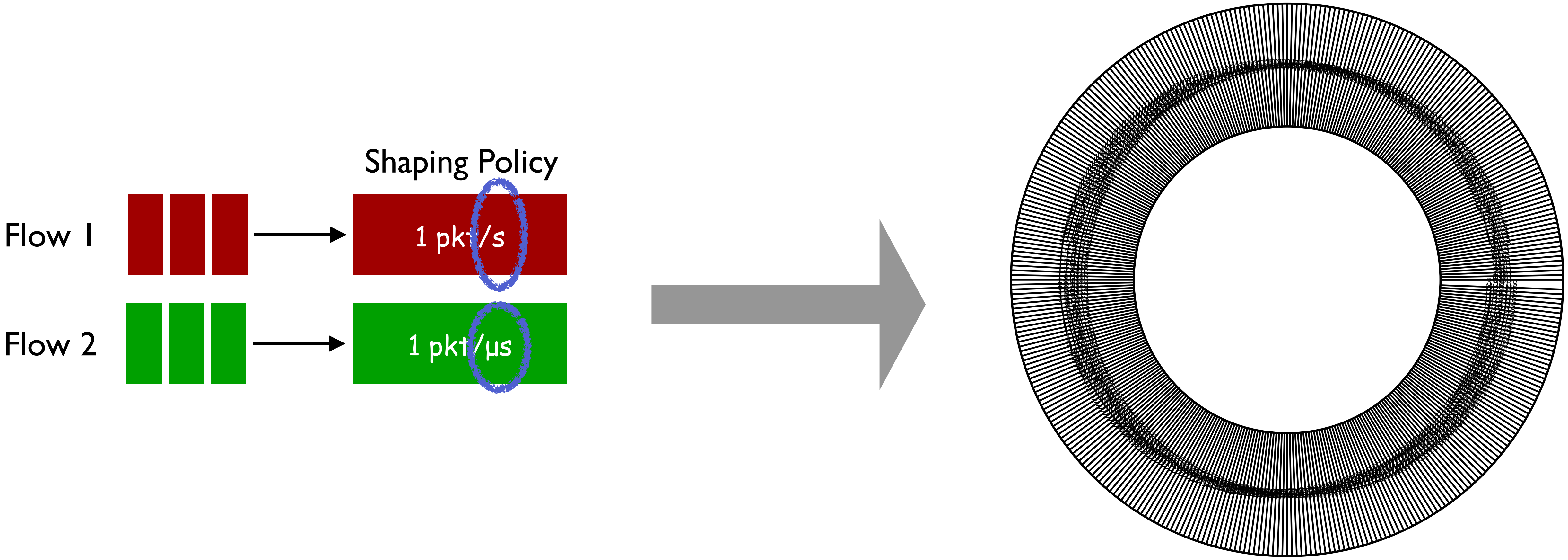
Timing Wheel:

✗ CPU inefficient: Multiple dequeue operations for a single batch

✗ Memory inefficient: Huge memory requirement

Challenge

Question: How to efficiently place and dequeue inter-flow-batched packets



Granularity: 1 μs
# Slots: 9×10^6

Timing Wheel:

✗ CPU inefficient: Multiple dequeue operations for a single batch

✗ Memory inefficient: Huge memory requirement

Challenge

Question: How to efficiently place and extract inter-flow-batched packets

🤔 How to achieve fine granularity and wide time-range simultaneously?

Challenge

Question: How to efficiently place and extract inter-flow-batched packets

🤔 How to achieve fine granularity and wide time-range simultaneously?

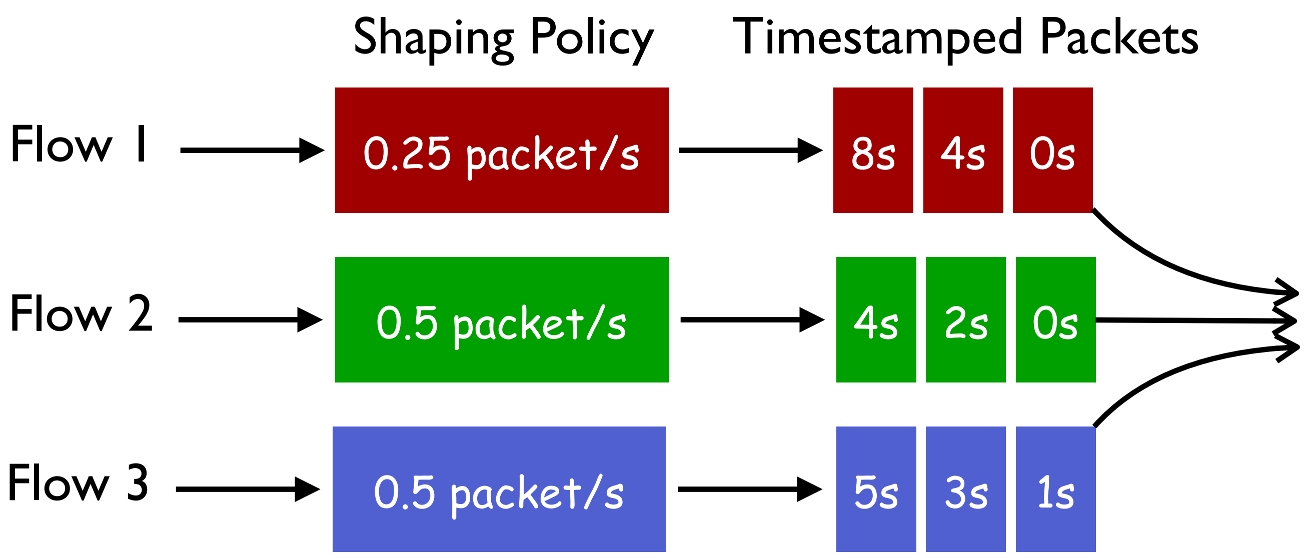
↓

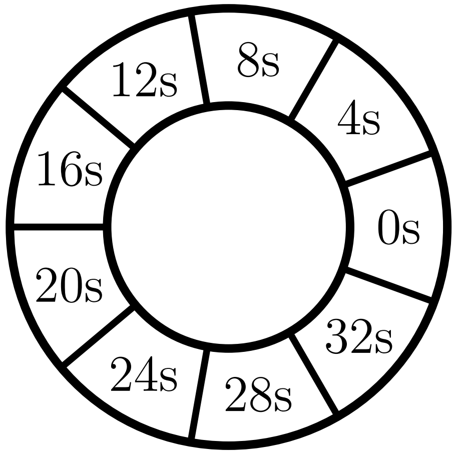
💡 Water meter

Challenge

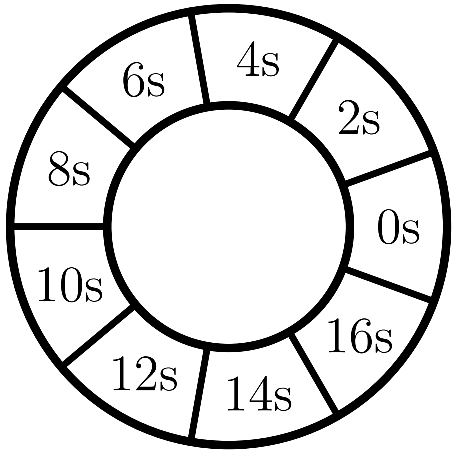
Question: How to efficiently place and extract inter-flow-batched packets

Answer: Muti-level Timing Wheel

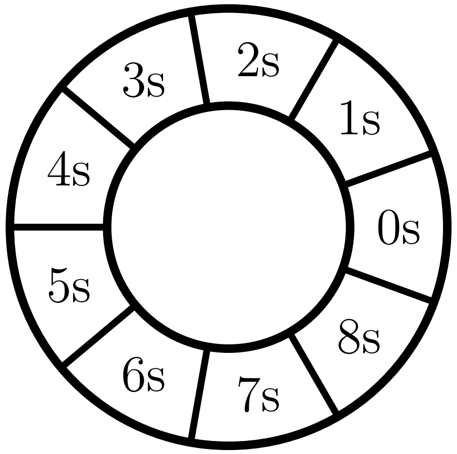




Granularity: 4s
Rate: 0.25 packet/s



Granularity: 2s
Rate: 0.5 packet/s

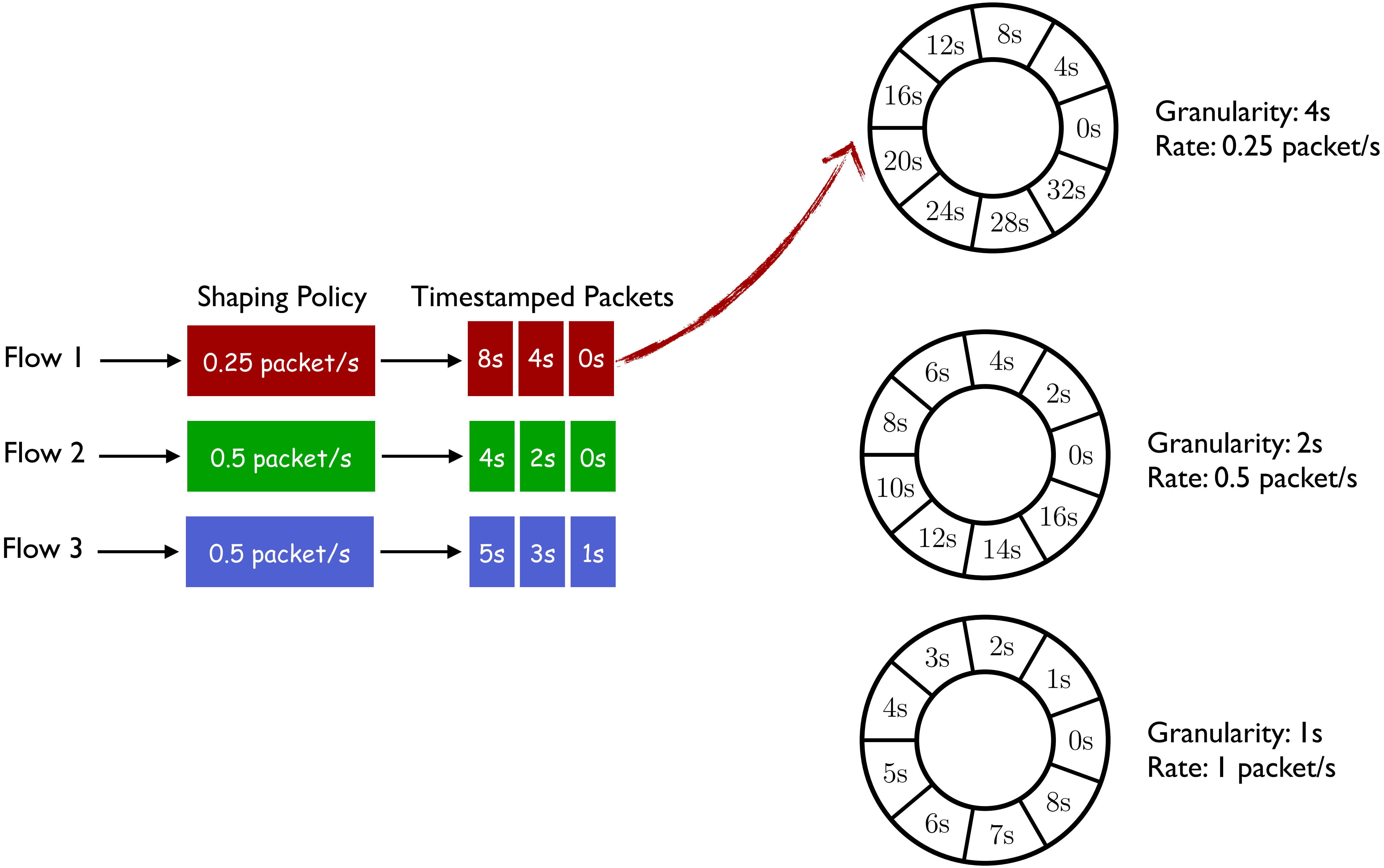


Granularity: 1s
Rate: 1 packet/s

Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow’s shaping rate

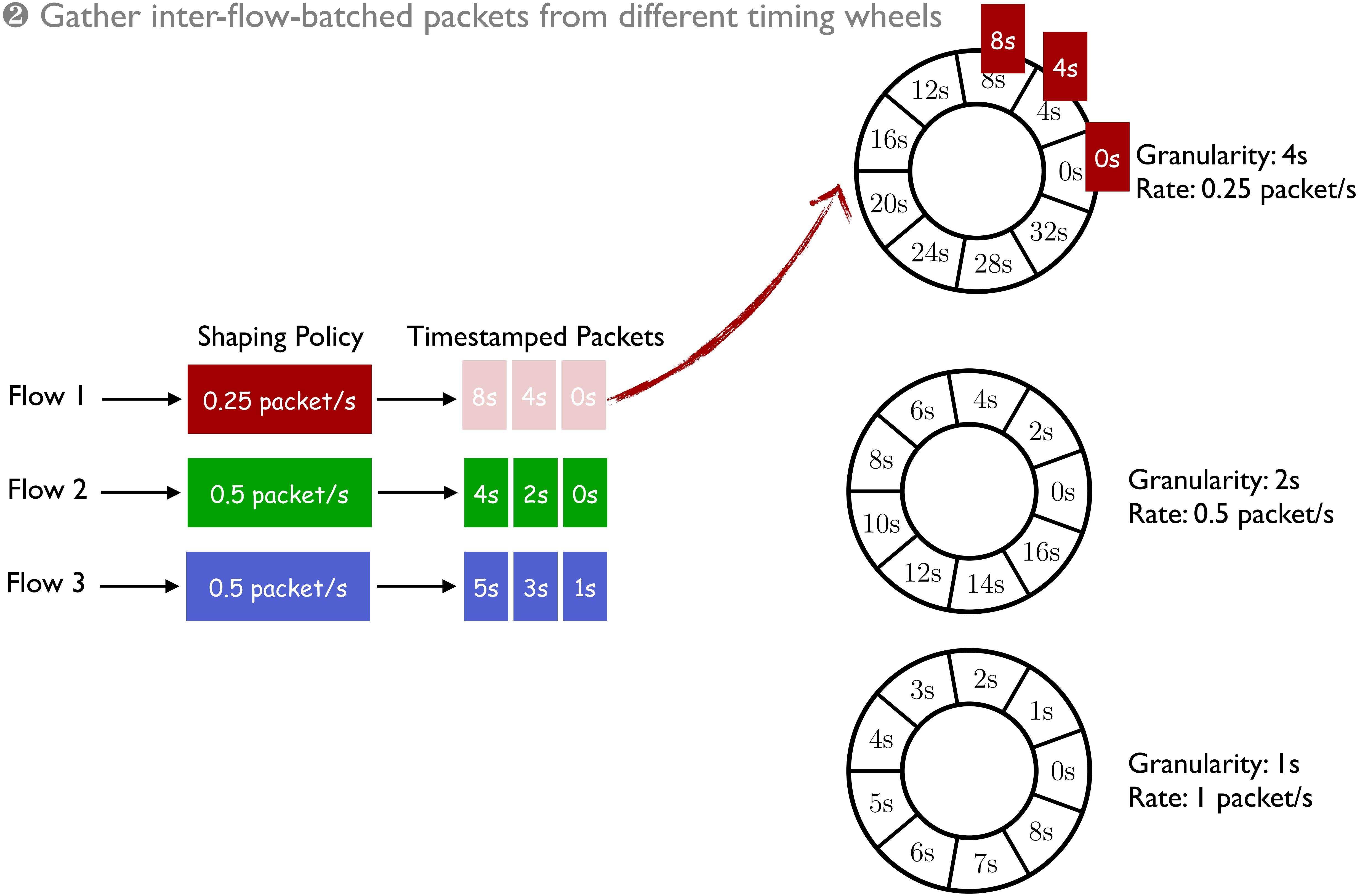
❷ Gather inter-flow-batched packets from different timing wheels



Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

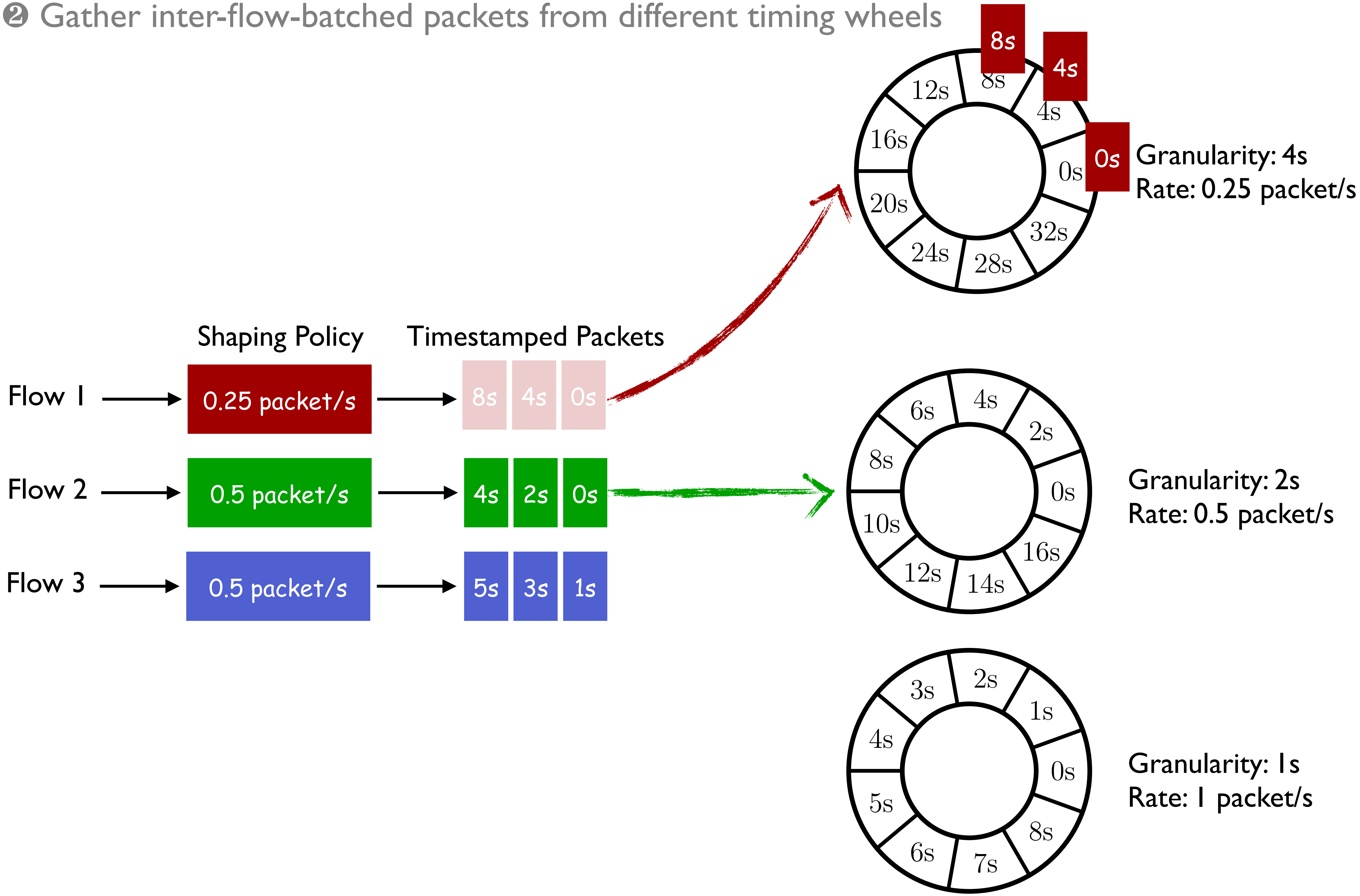
❷ Gather inter-flow-batched packets from different timing wheels



Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

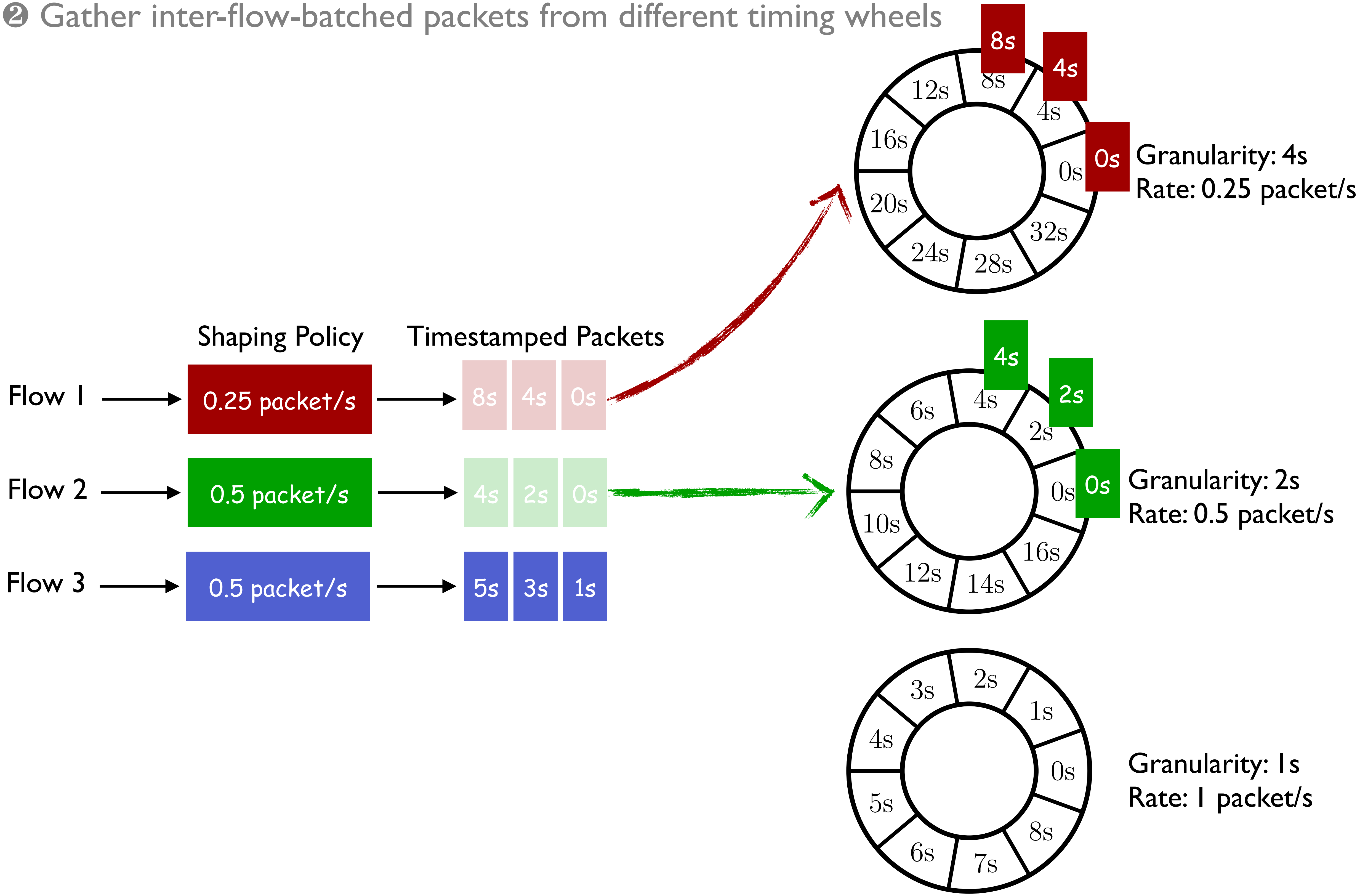
❷ Gather inter-flow-batched packets from different timing wheels



Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

❷ Gather inter-flow-batched packets from different timing wheels


Shaping Policy

Timestamped Packets

Flow 1 → 0.25 packet/s → 8s 4s 0s

Flow 2 → 0.5 packet/s → 4s 2s 0s

Flow 3 → 0.5 packet/s → 5s 3s 1s

Granularity: 4s
Rate: 0.25 packet/s

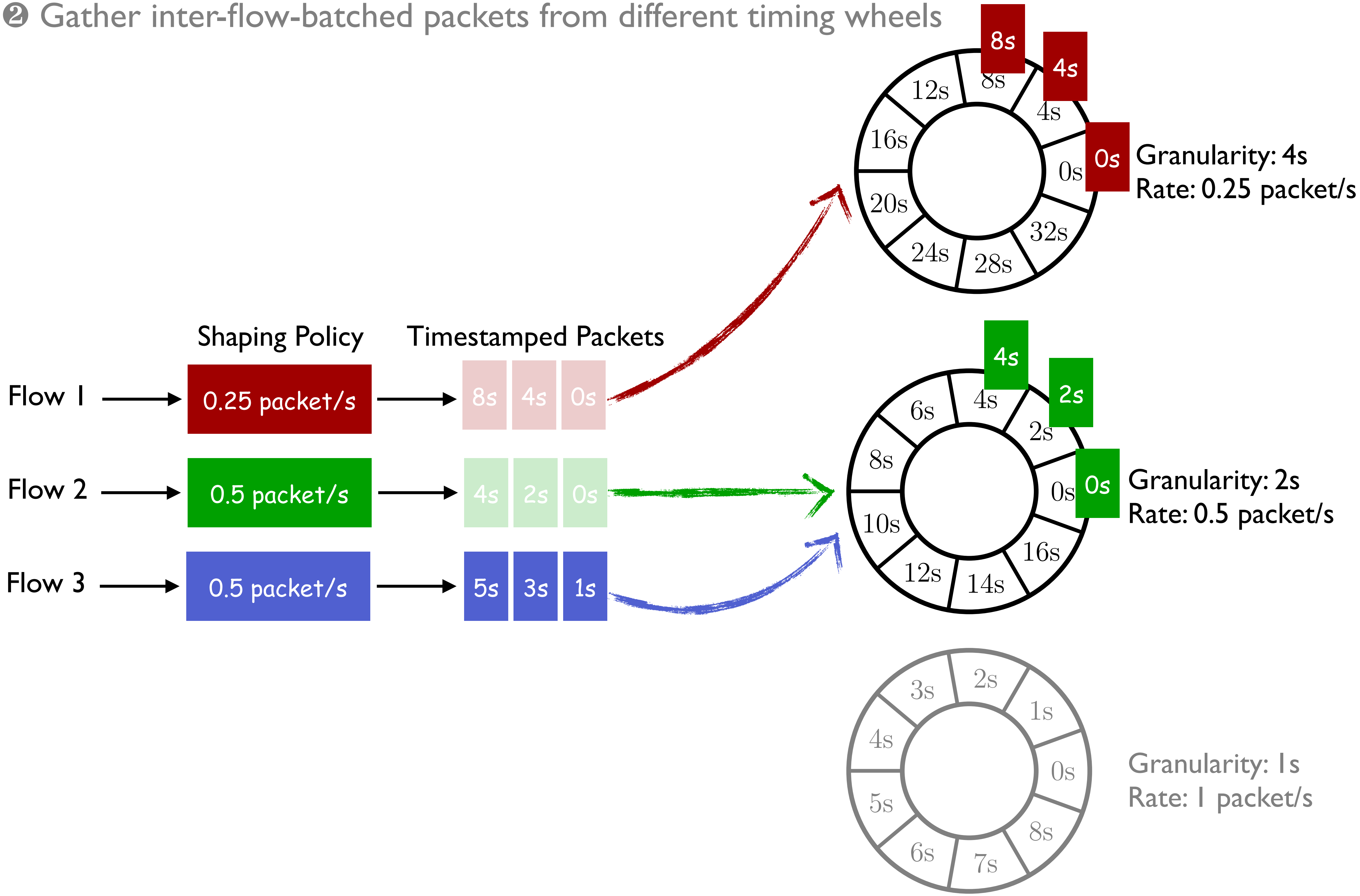
Granularity: 2s
Rate: 0.5 packet/s

Granularity: 1s
Rate: 1 packet/s

Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

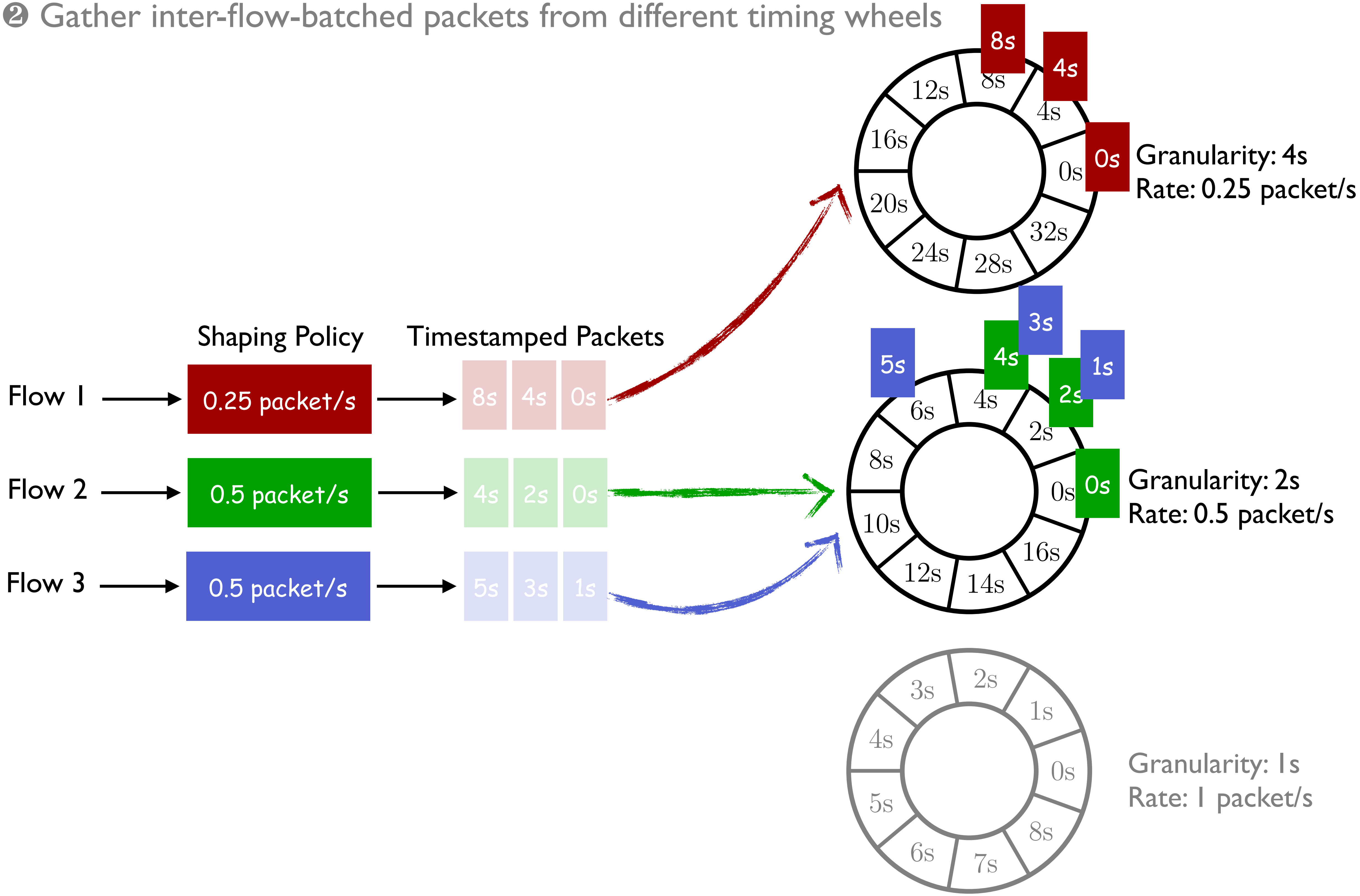
❷ Gather inter-flow-batched packets from different timing wheels



Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

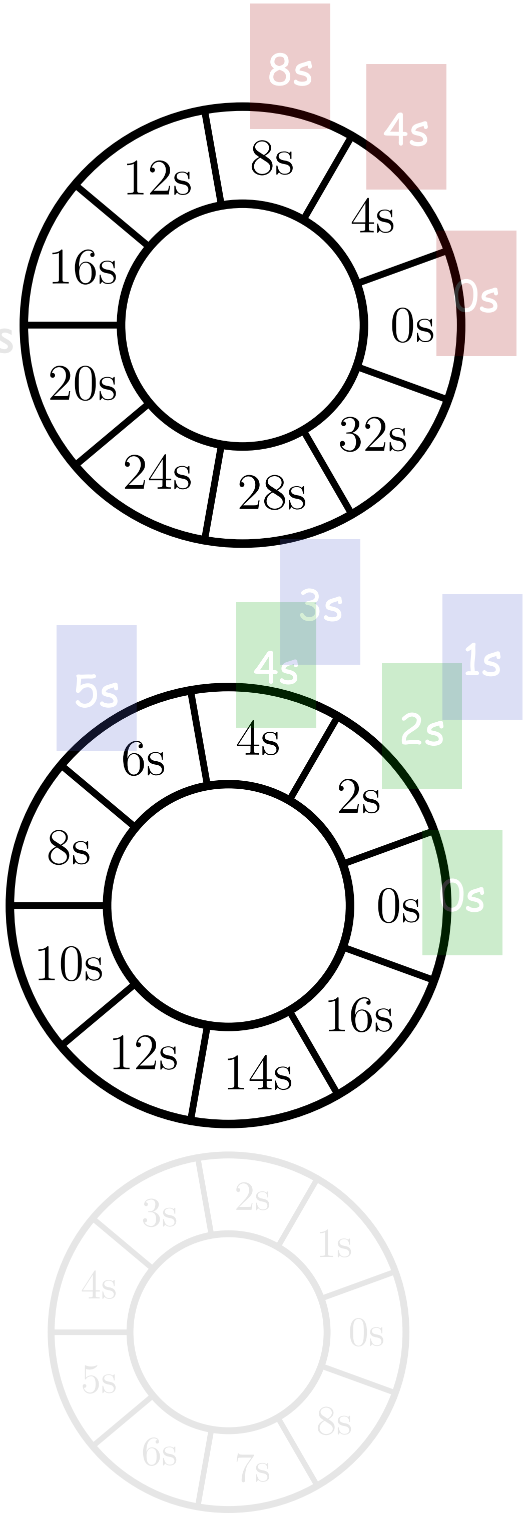
❷ Gather inter-flow-batched packets from different timing wheels



Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow’s shaping rate

❷ Gather inter-flow-batched packets from different timing wheels



Granularity: 4s
Rate: 0.25 packet/s

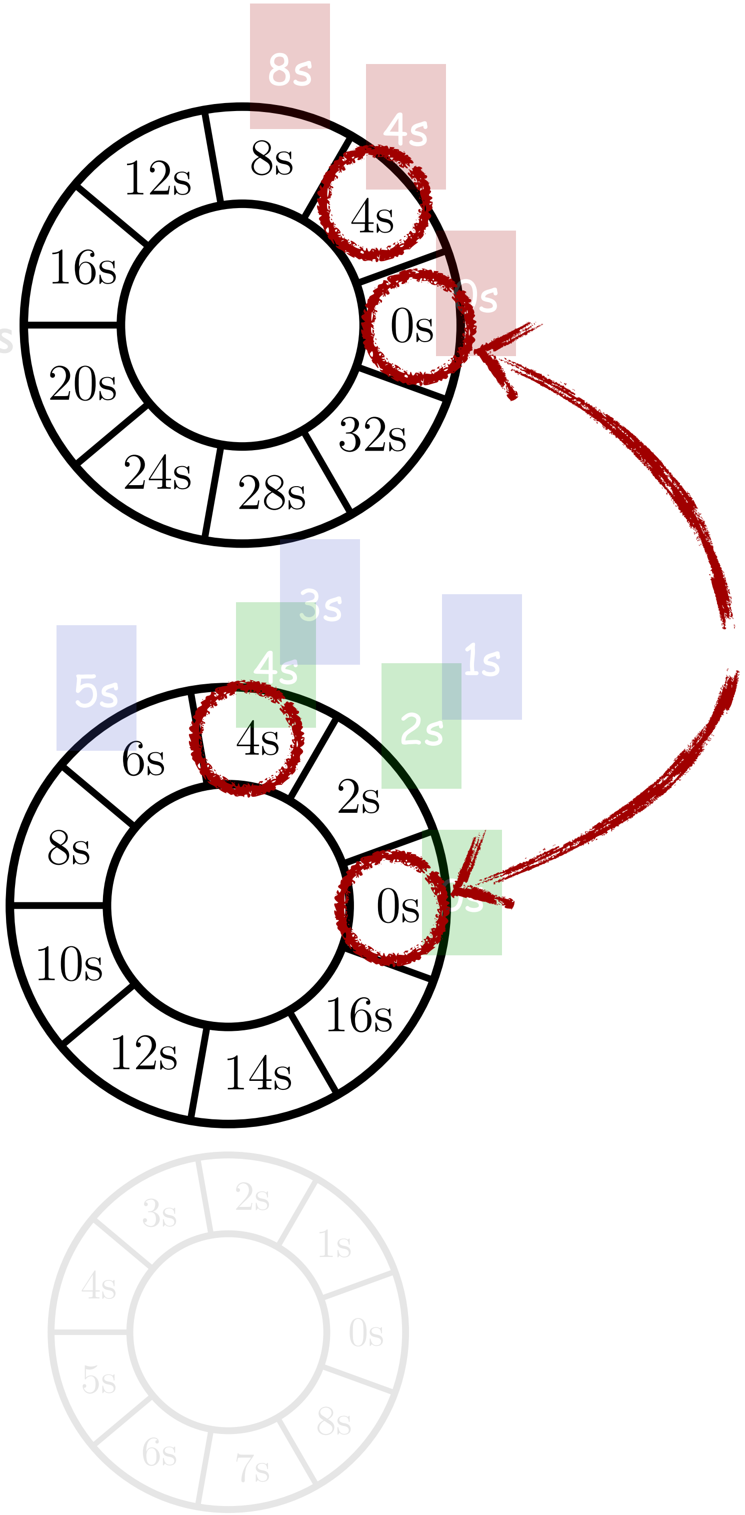
Granularity: 2s
Rate: 0.5 packet/s

Granularity: 1s
Rate: 1 packet/s

Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow’s shaping rate

❷ Gather inter-flow-batched packets from different timing wheels



Granularity: 4s
Rate: 0.25 packet/s

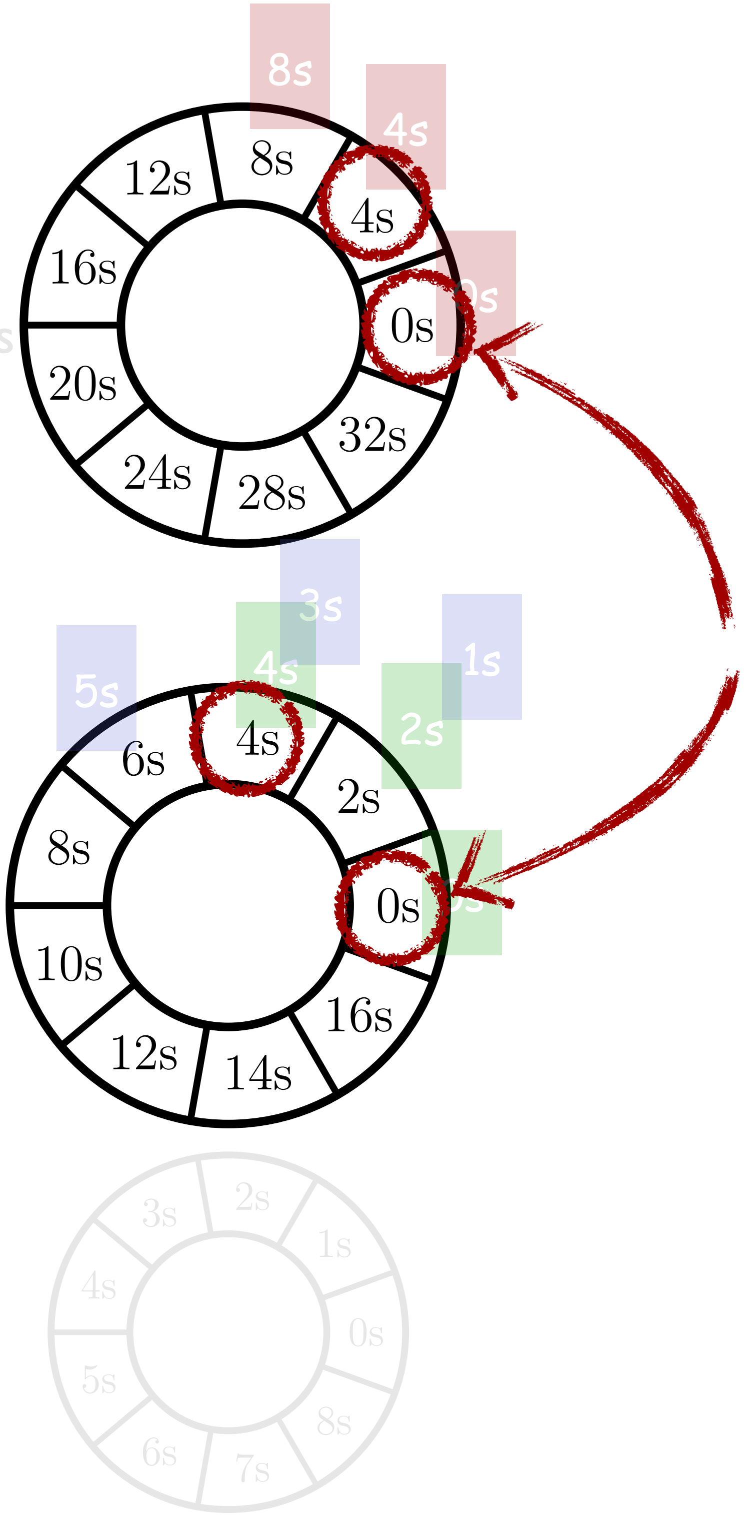
Granularity: 2s
Rate: 0.5 packet/s

Granularity: 1s
Rate: 1 packet/s

Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

❷ Gather inter-flow-batched packets from different timing wheels



Granularity: 4s
Rate: 0.25 packet/s

Granularity: 2s
Rate: 0.5 packet/s

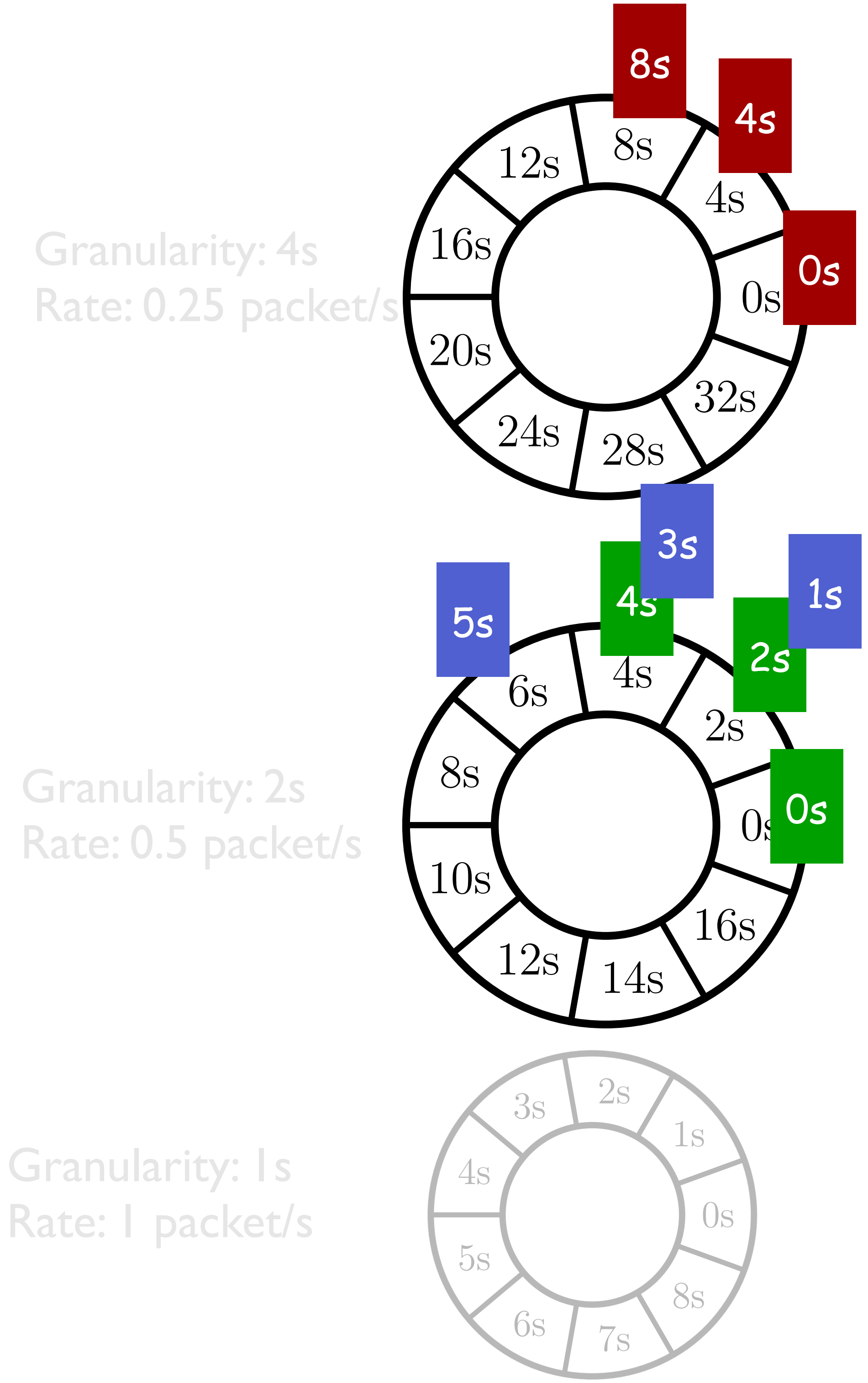
Granularity: 1s
Rate: 1 packet/s

- Have the same timestamp → Sent at the same time

Muti-level Timing Wheel (MLTW)

❶ Put packet into the queue whose granularity best matches the flow's shaping rate

❷ Gather inter-flow-batched packets from different timing wheels

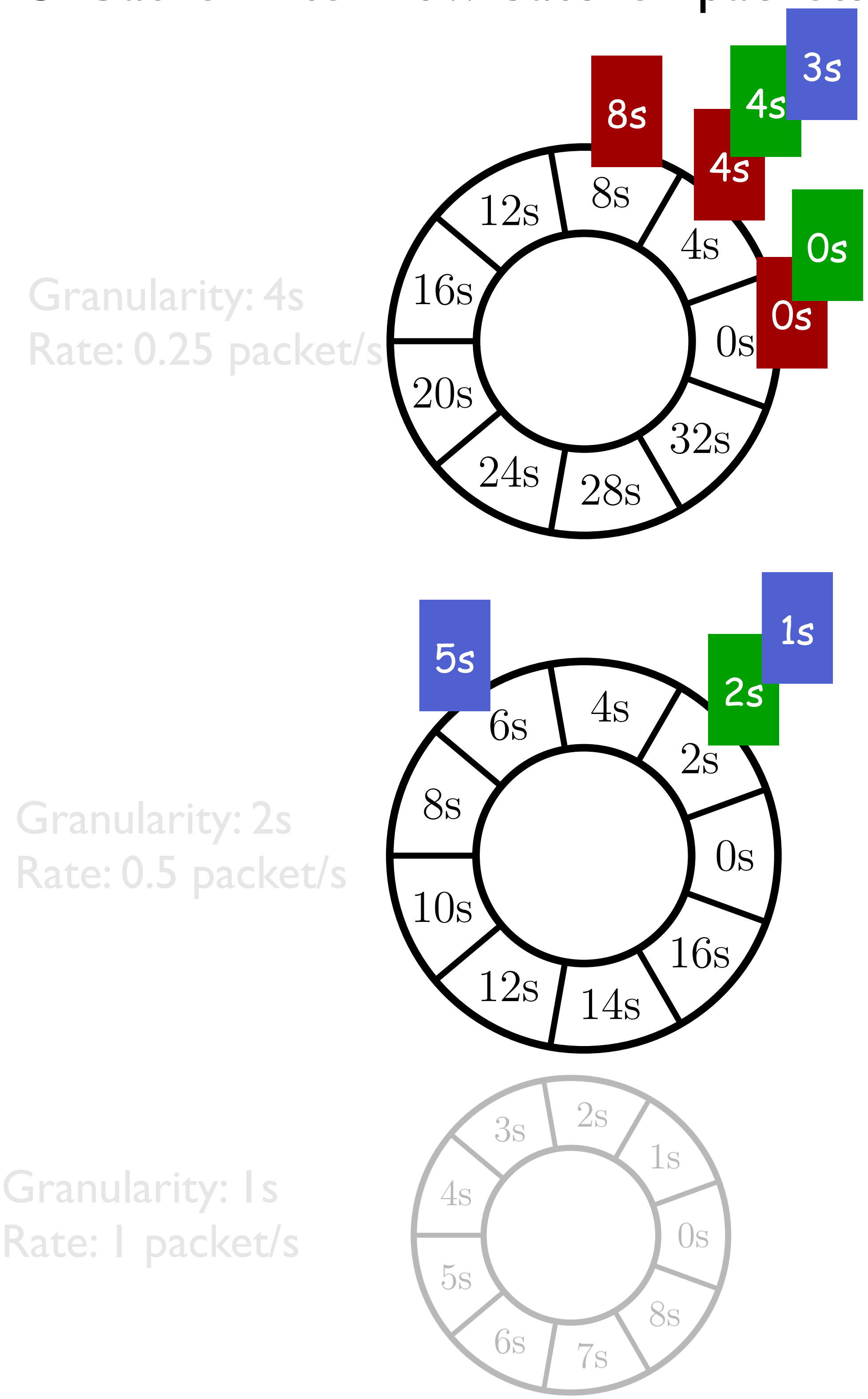


- Have the same timestamp → Sent at the same time
- Gather these packets into the same slot
  ✓ Place inter-flow-batched packets together
  ✓ Reduce # of dequeue operations

Muti-level Timing Wheel (MLTW)

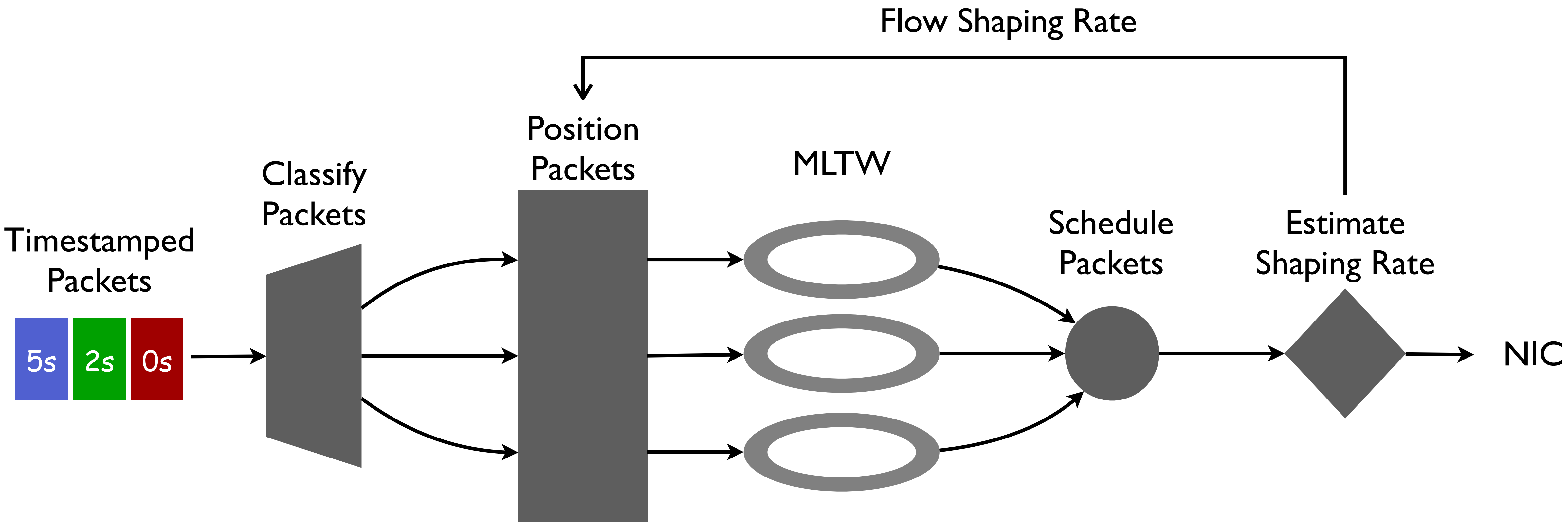
❶ Put packet into the queue whose granularity best matches the flow’s shaping rate

❷ Gather inter-flow-batched packets from different timing wheels

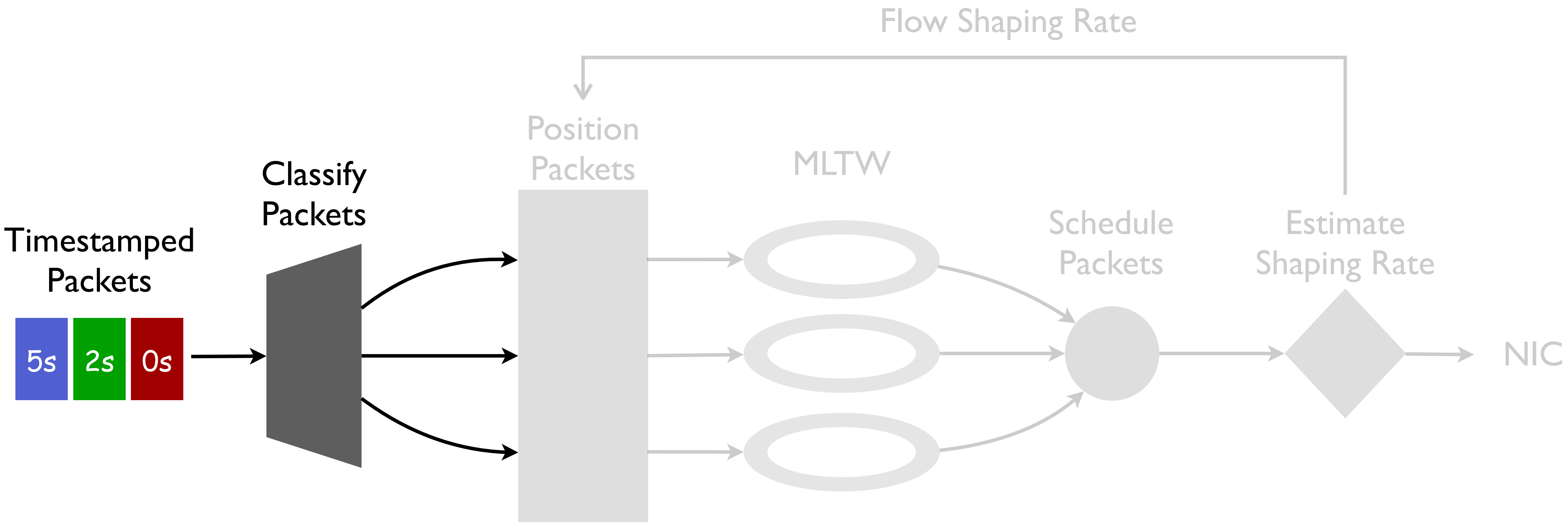


- Have the same timestamp → Sent at the same time
- Gather these packets into the same slot
  - ✓ Place inter-flow-batched packets together
  - ✓ Reduce # of dequeue operations

Putting Together: FlowBundler

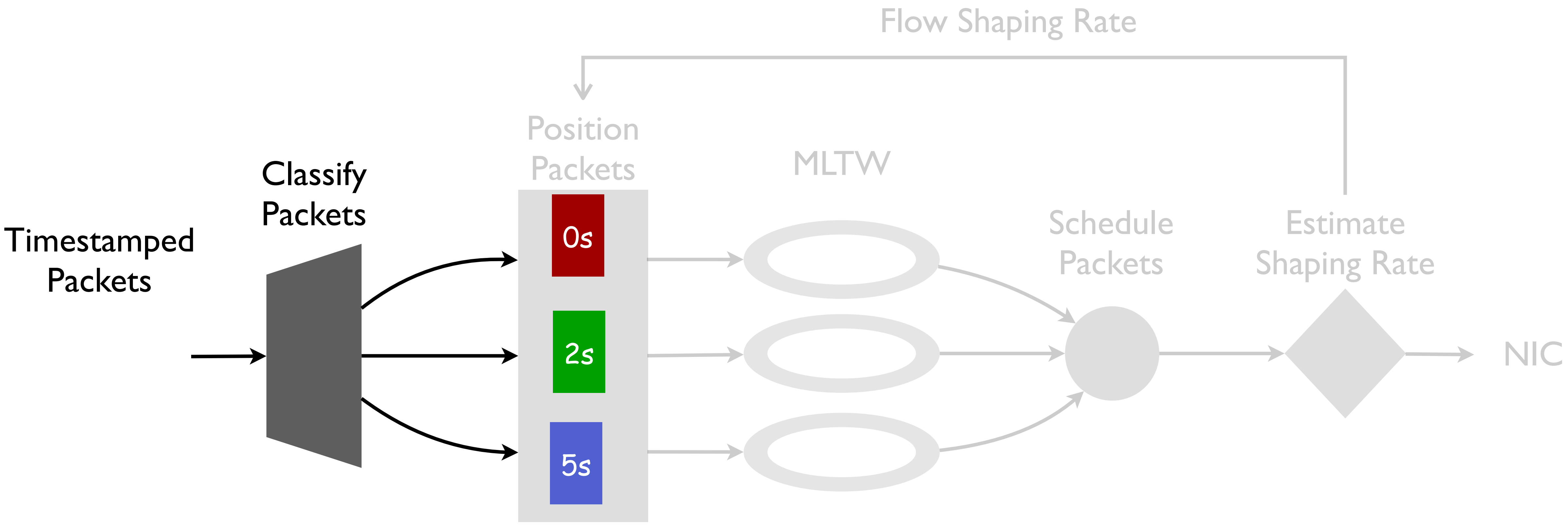


Putting Together: FlowBundler



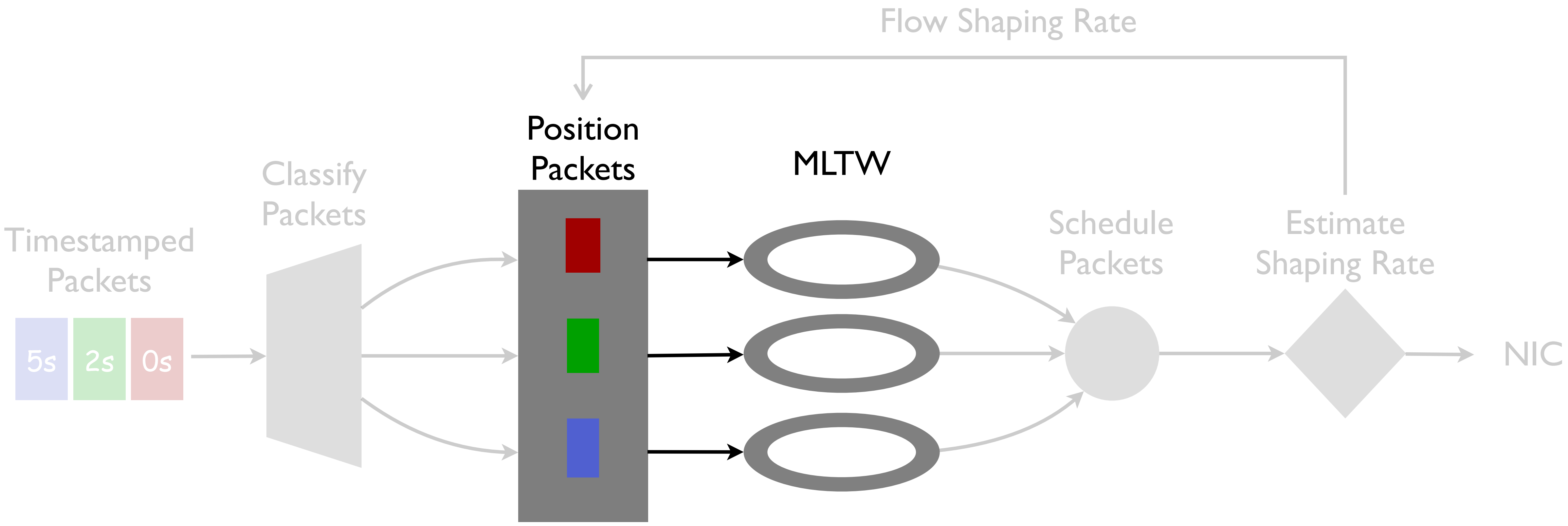
❶ Classify packets into flows

Putting Together: FlowBundler



❶ Classify packets into flows

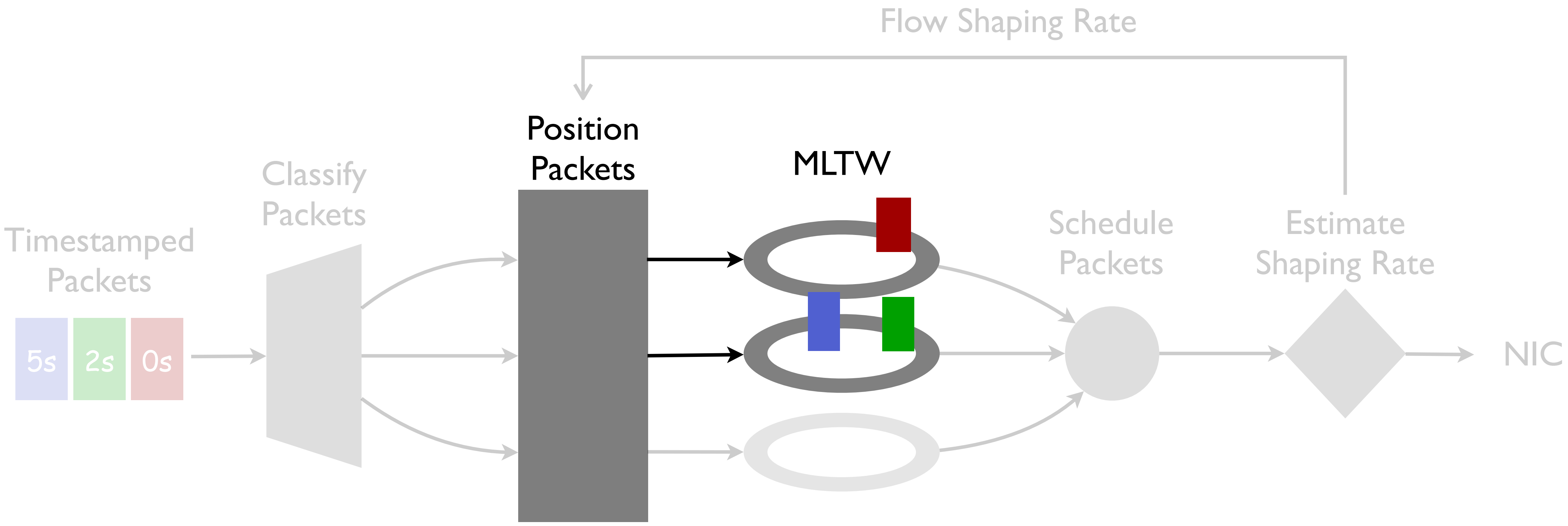
Putting Together: FlowBundler



❶ Classify packets into flows

❷ Place packets into MLTW based on flow shaping rate and timestamp

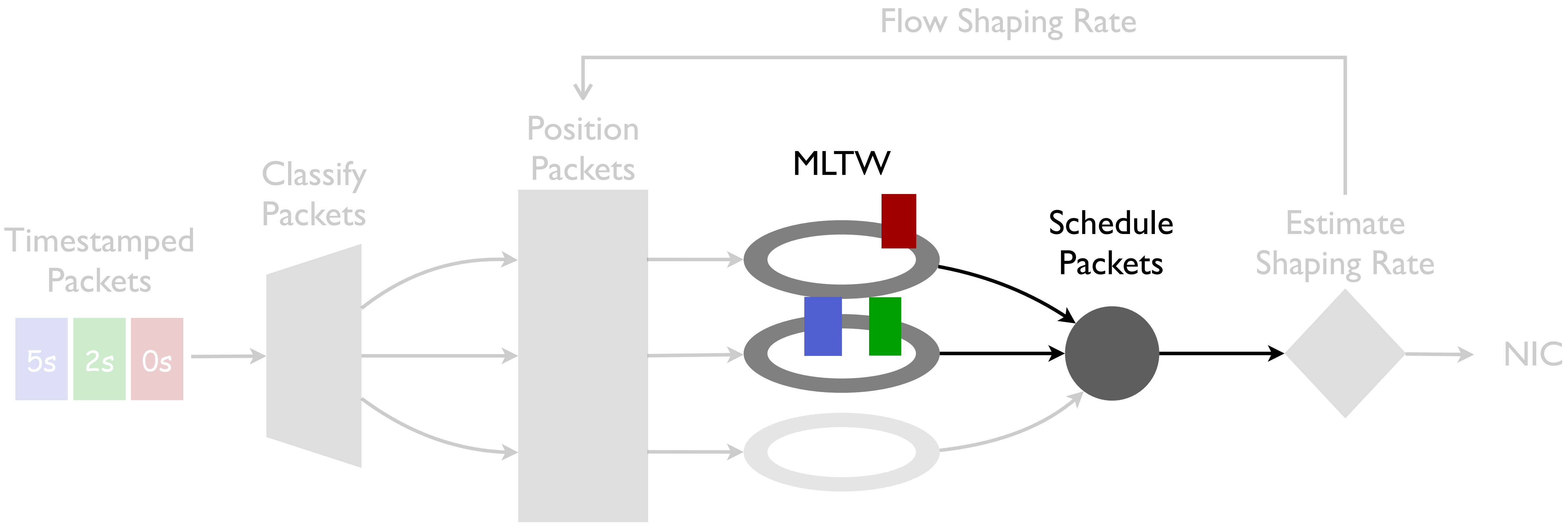
Putting Together: FlowBundler



❶ Classify packets into flows

❷ Place packets into MLTW based on flow shaping rate and timestamp

Putting Together: FlowBundler

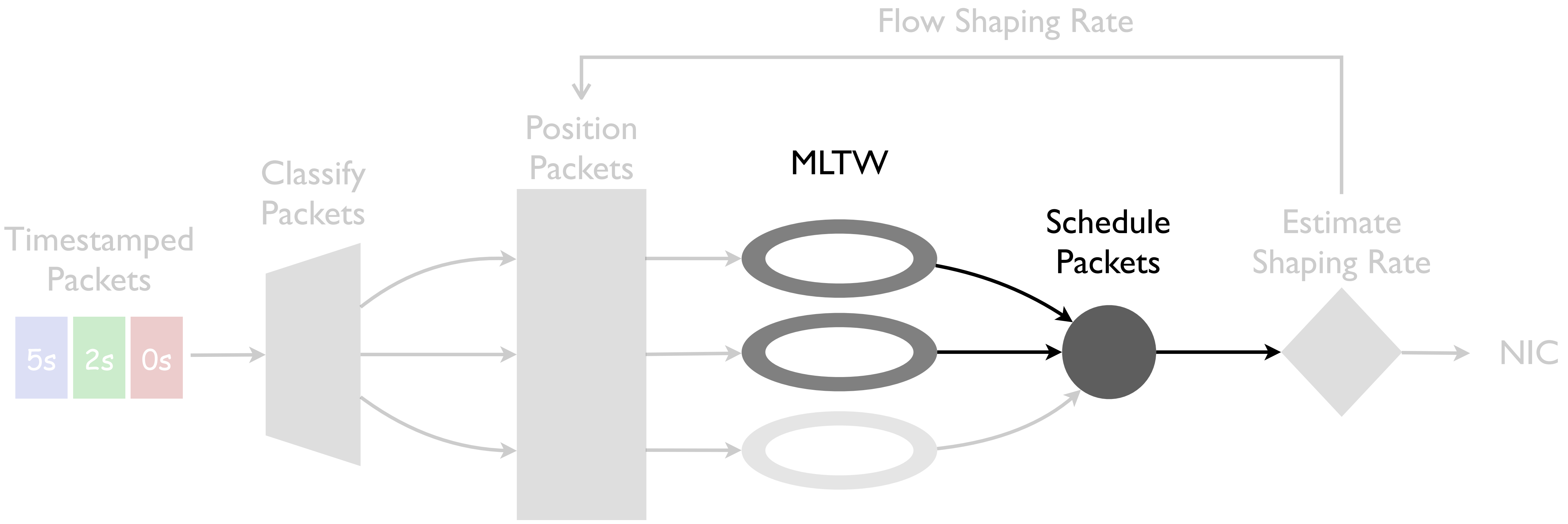


❶ Classify packets into flows

❷ Place packets into MLTW based on flow shaping rate and timestamp

❸ Dequeue packets from MLTW based on current time

Putting Together: FlowBundler

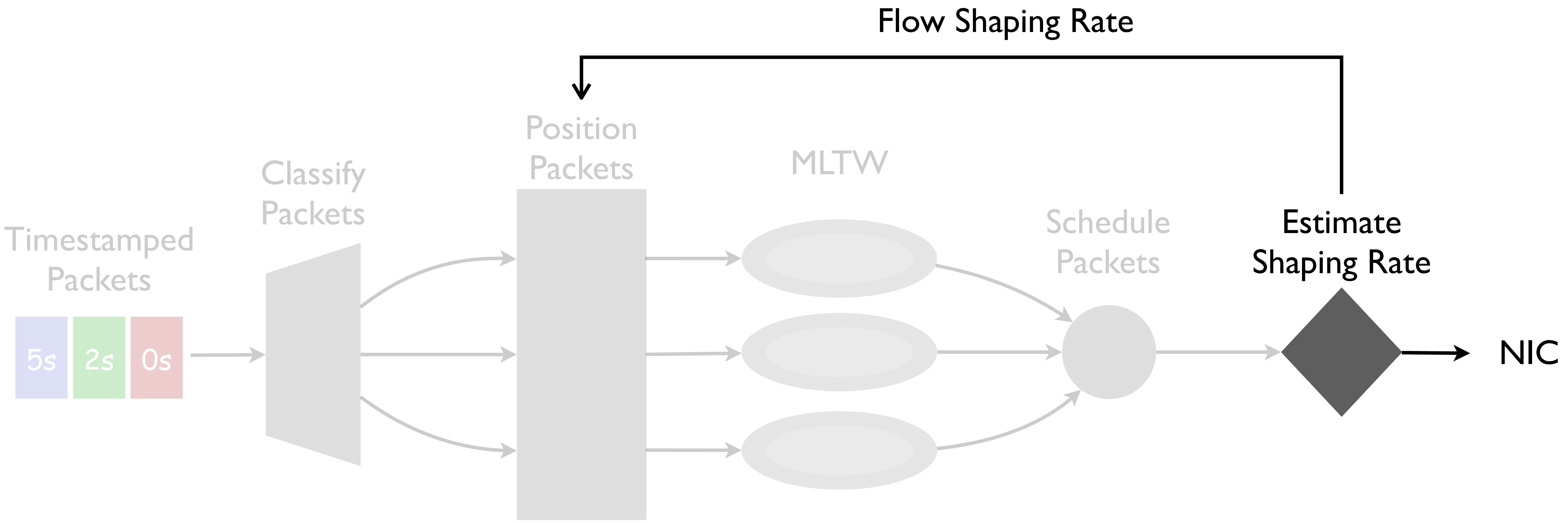


❶ Classify packets into flows

❷ Place packets into MLTW based on flow shaping rate and timestamp

❸ Dequeue packets from MLTW based on current time

Putting Together: FlowBundler



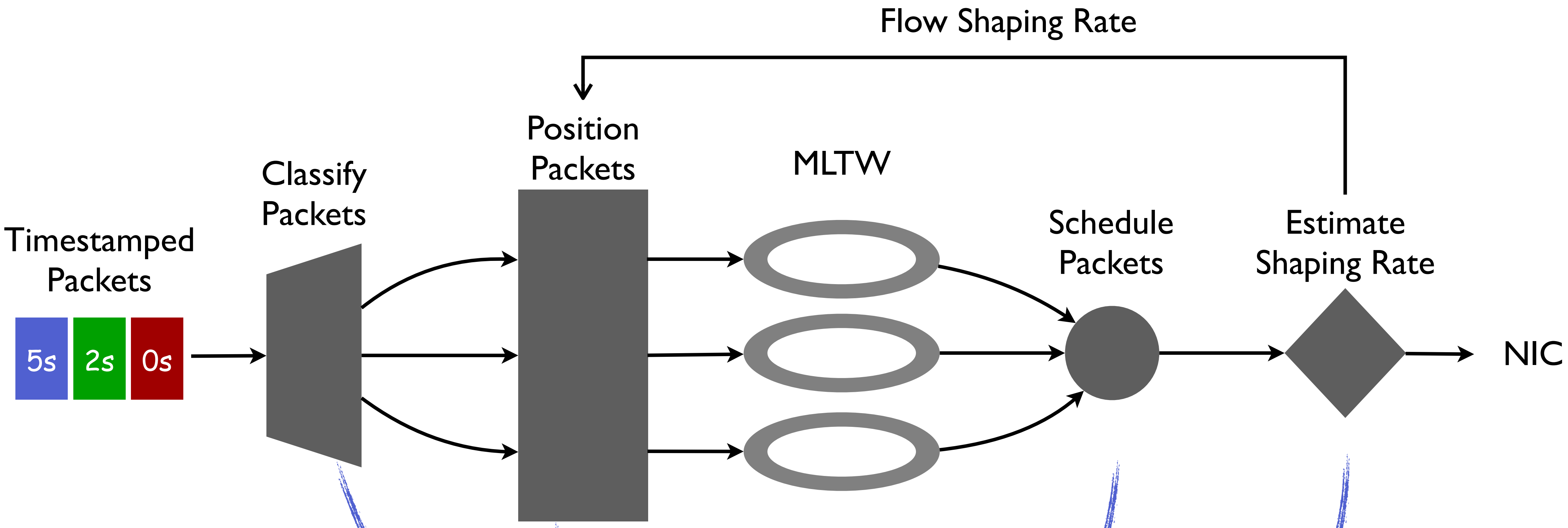
❶ Classify packets into flows

❷ Place packets into MLTW based on flow shaping rate and timestamp

❸ Dequeue packets from MLTW based on current time

❹ Estimate the shaping rate of each flow

Putting Together: FlowBundler



All operations can be achieved with O(1) time complexity
(More Details in the Paper)

Implementation

- Kernel
  - As a Linux queueing discipline

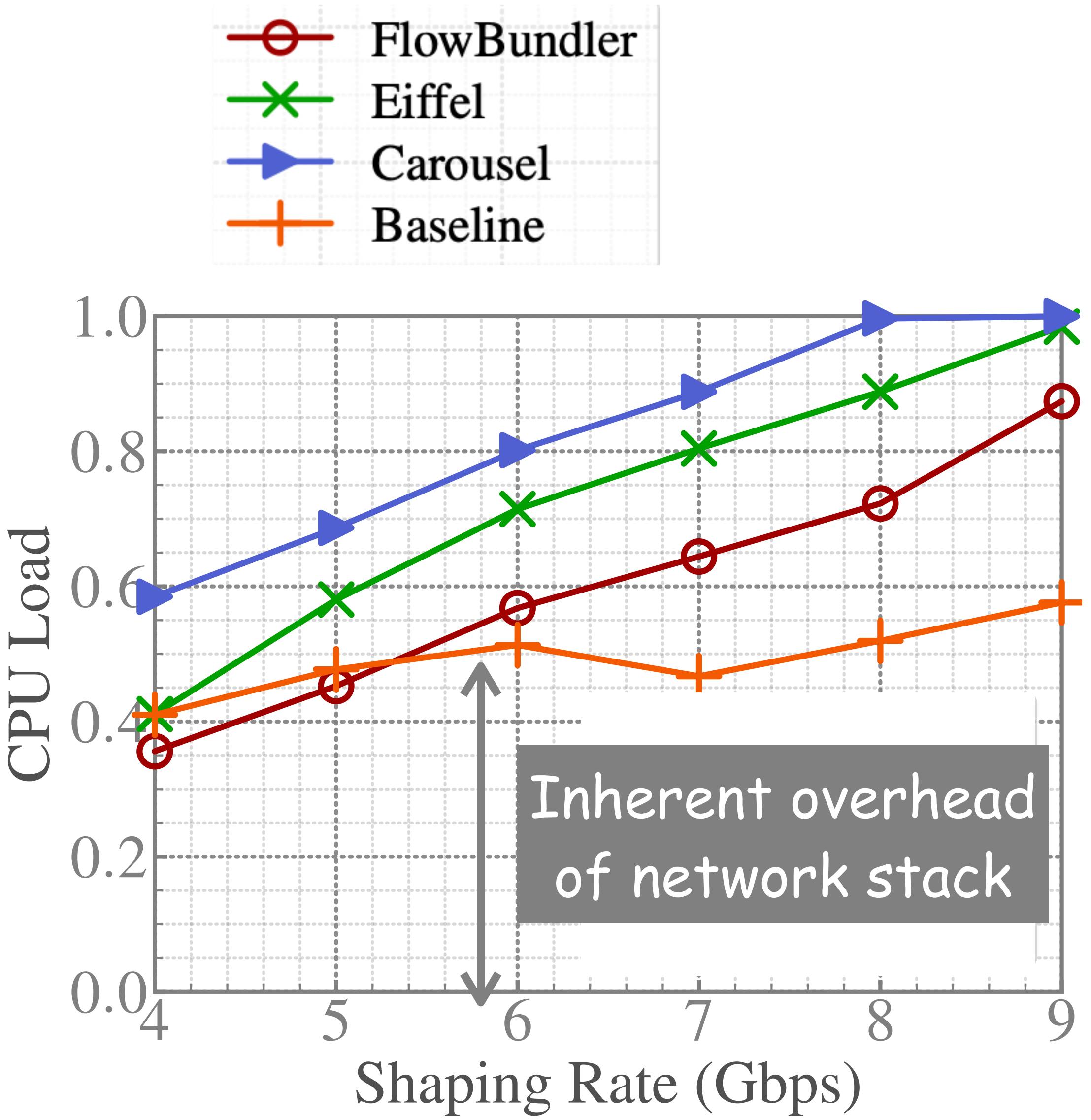
- Userspace
  - Based on BESS/DPDK (a kind of Software NIC)

Open source: https://github.com/ants-xjtu/FlowBundler

Evaluation

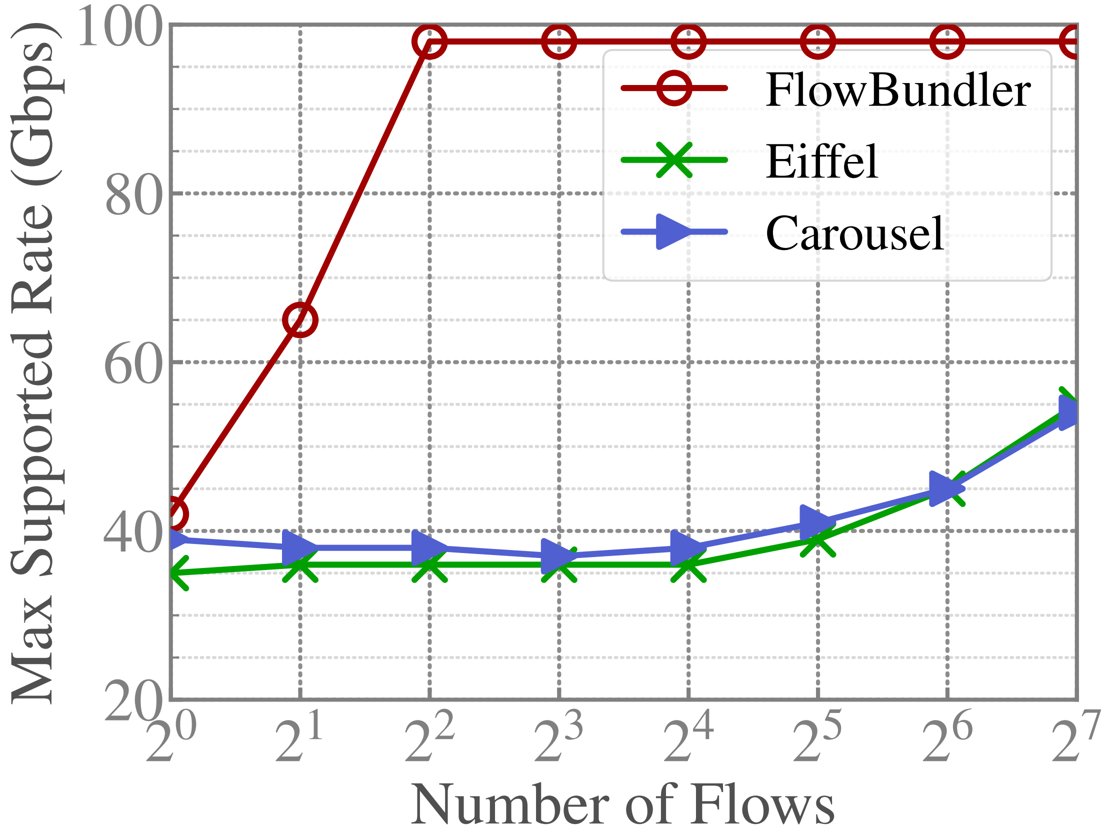
- Compared Schemes
  - Carousel[SIGCOMM'17]
  - Eiffel[NSDI'19]
- Metrics
  - CPU efficiency
  - Memory efficiency
  - Transmission performance

Evaluation — CPU Efficiency



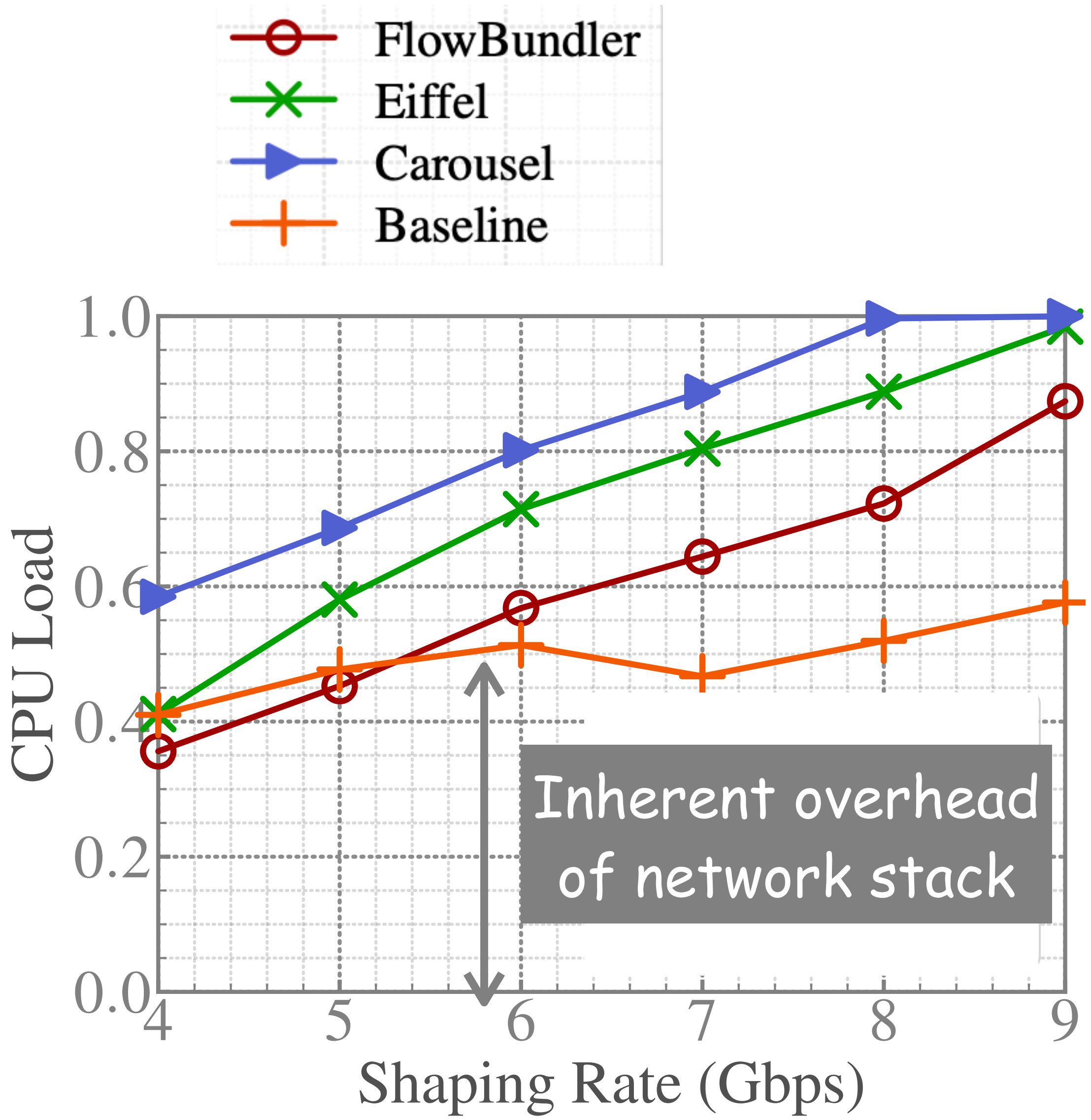
Inherent overhead of network stack

Kernel



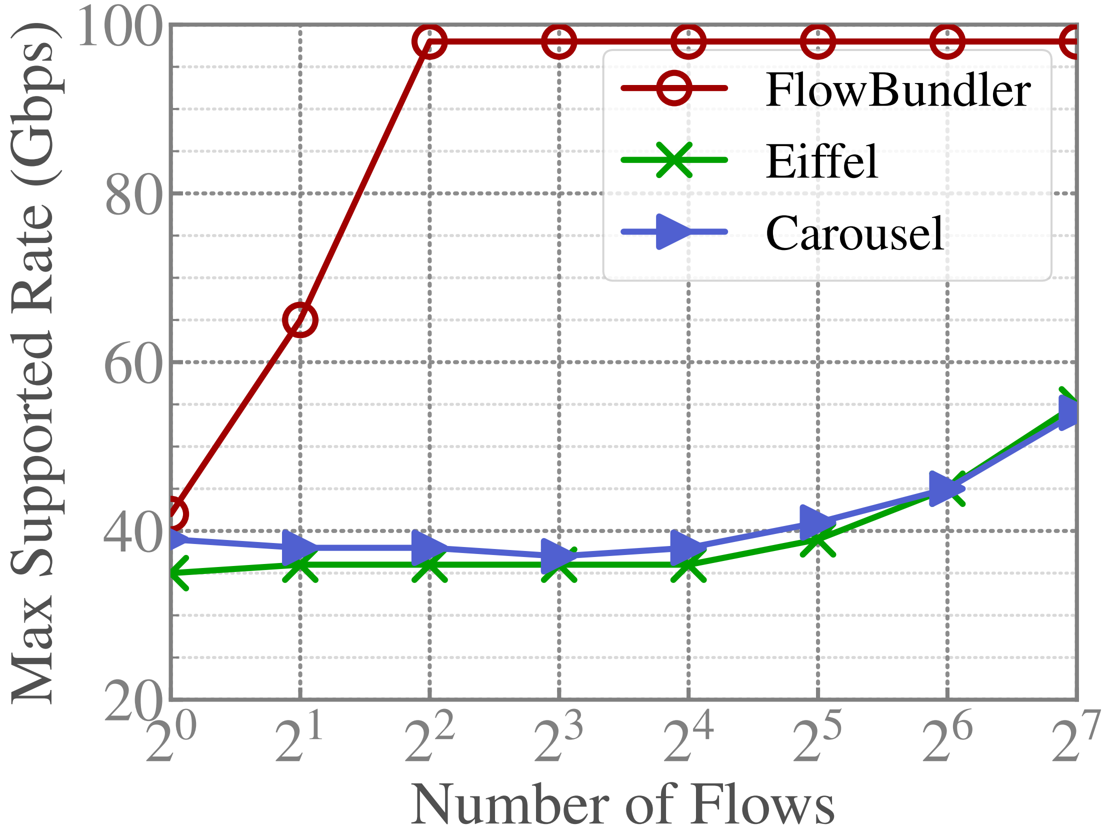
Userspace

Evaluation — CPU Efficiency



Inherent overhead of network stack

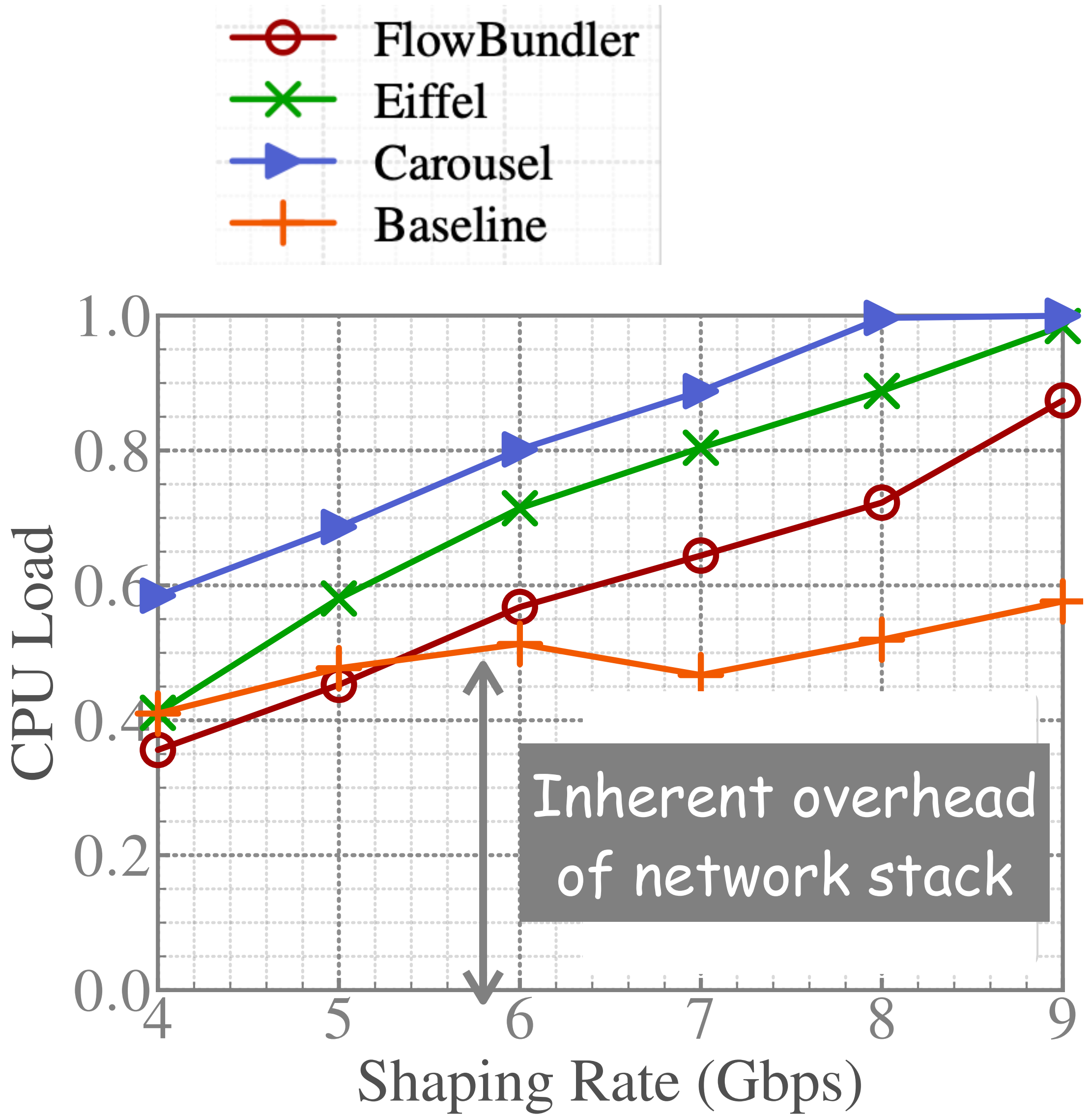
Kernel



Userspace

~20% lower cpu load

Evaluation — CPU Efficiency



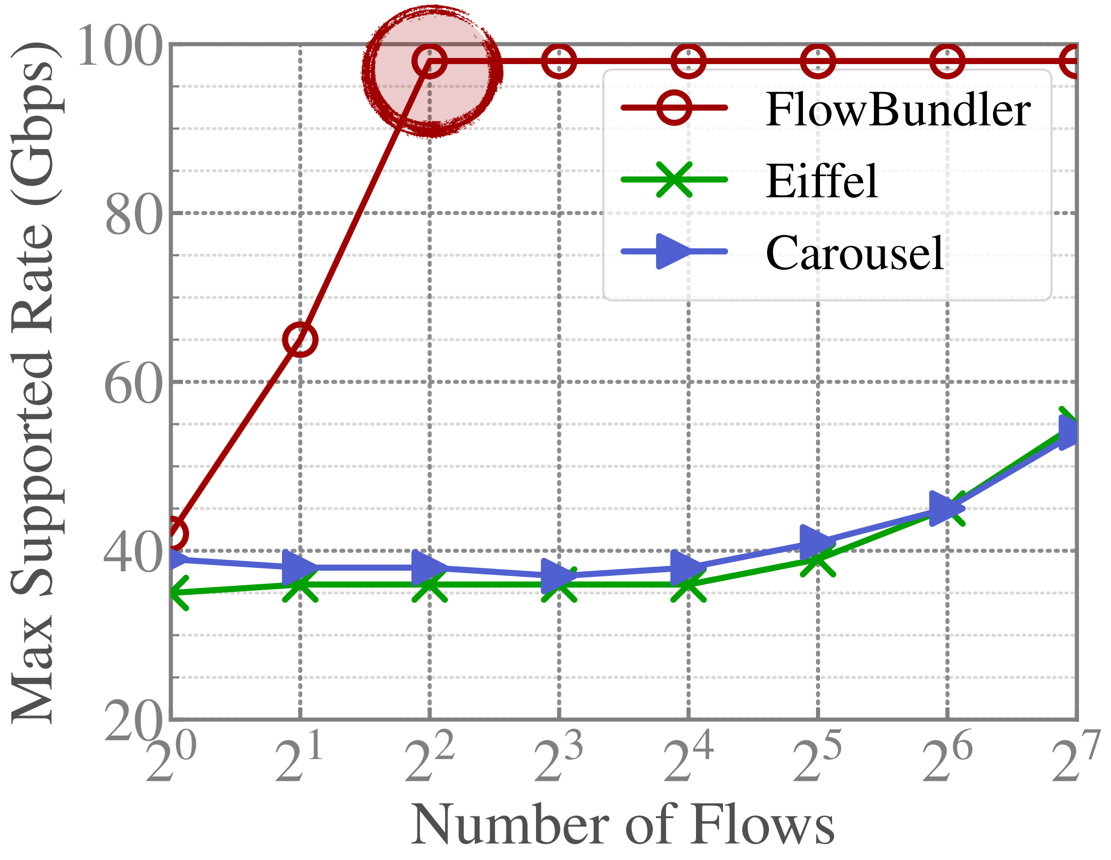
Inherent overhead of network stack

Kernel

~20% lower cpu load

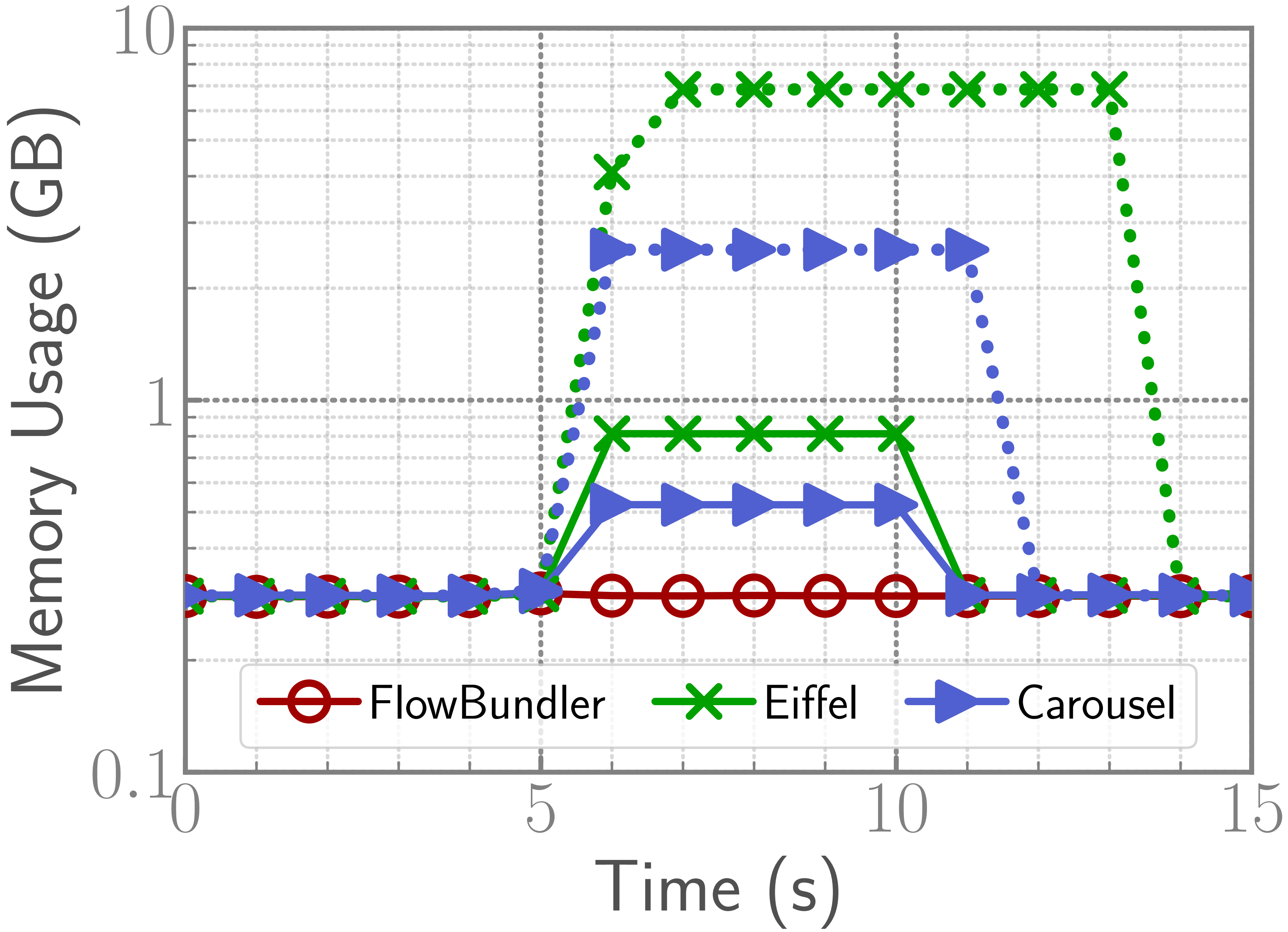
Near 100Gbps shaping speed with 4 flows

~2.6x higher shaping speed

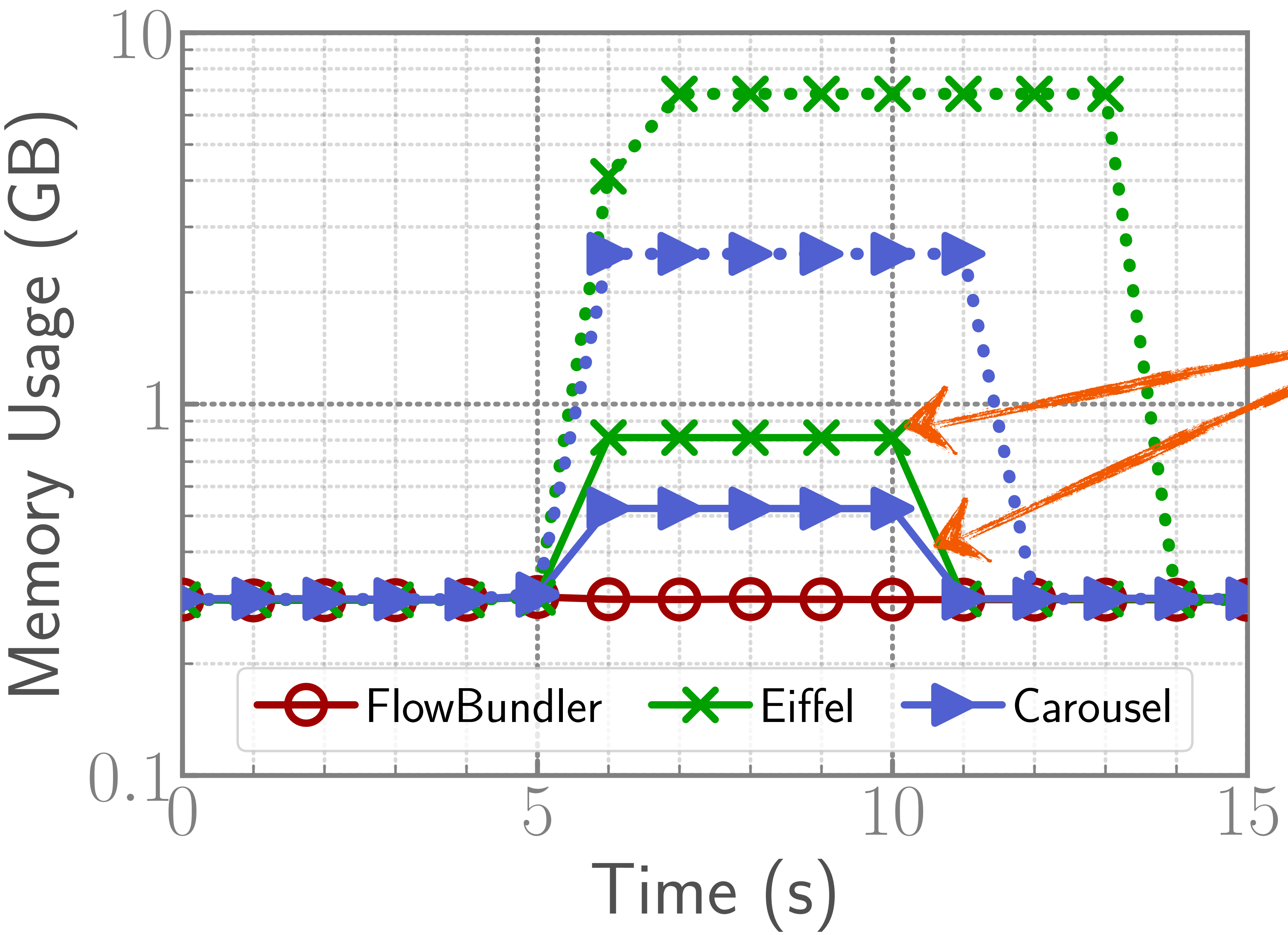


Userspace

Evaluation — Memory Efficiency (kernel)



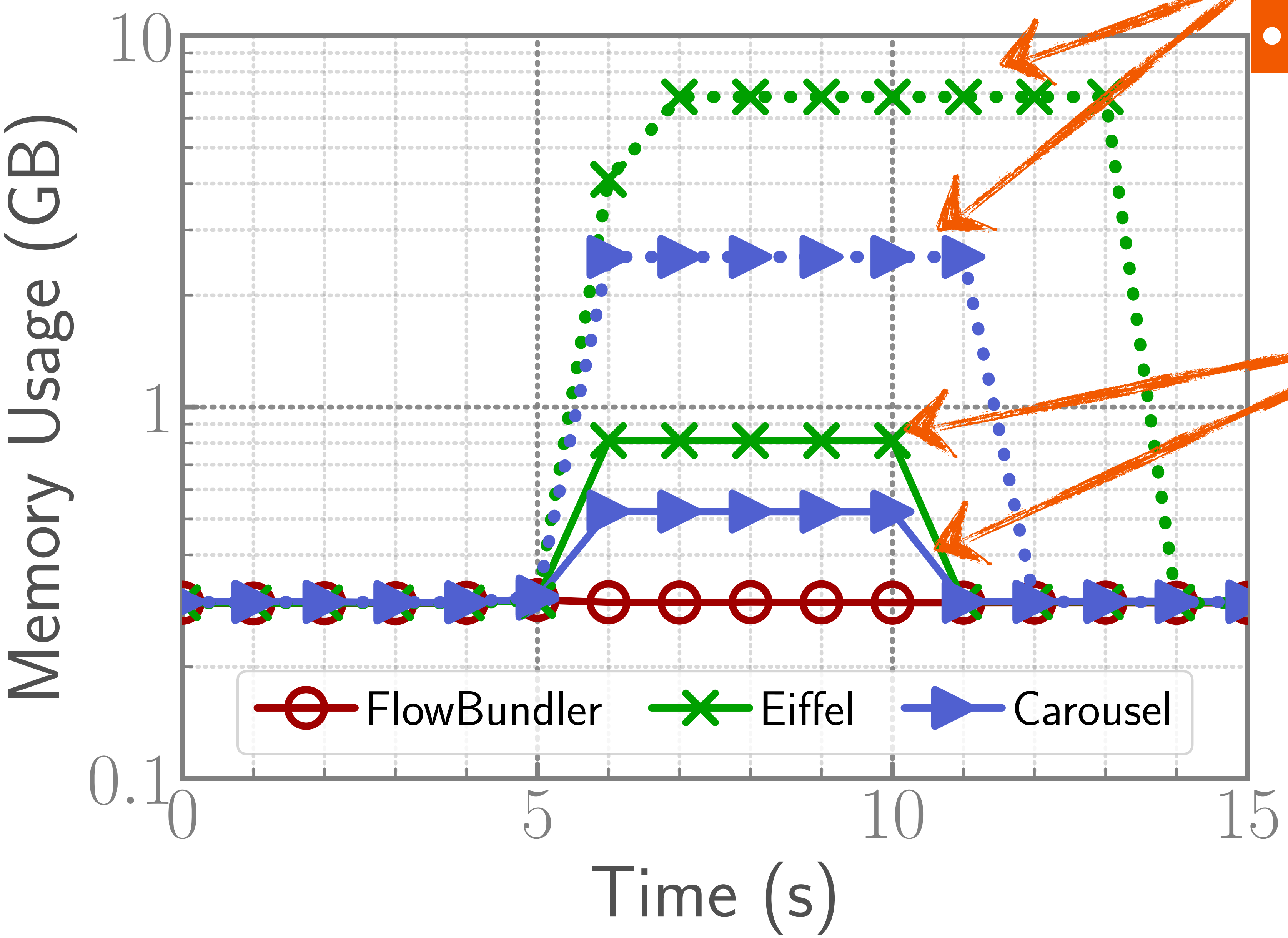
Evaluation — Memory Efficiency (kernel)



Eiffel/Carousel:

- 1µs granularity
- 0.5GB/0.2GB memory usage

Evaluation — Memory Efficiency (kernel)



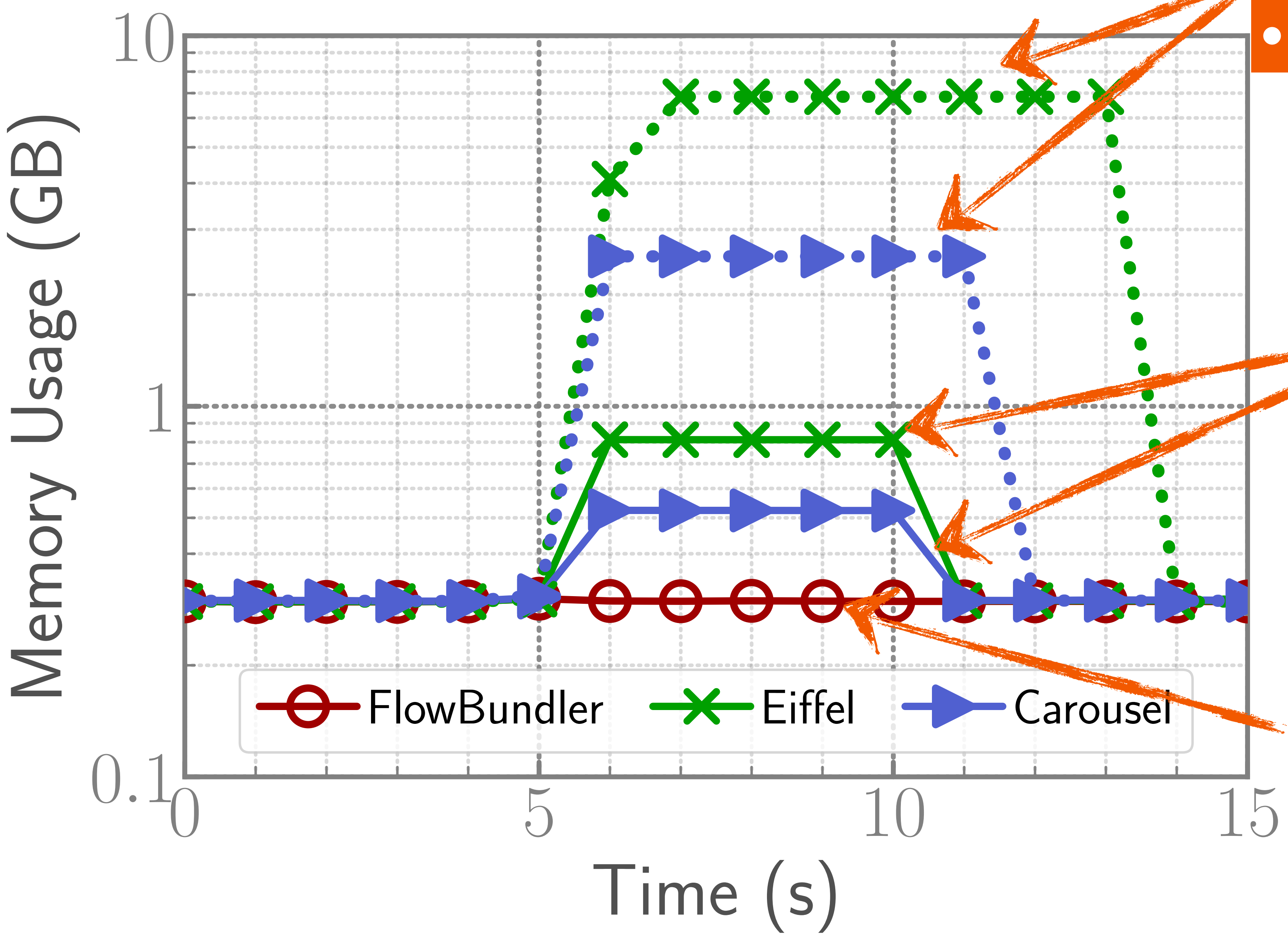
Eiffel/Carousel:

- 0.1 µs granularity
- 6.9GB/2.3GB memory usage

Eiffel/Carousel:

- 1 µs granularity
- 0.5GB/0.2GB memory usage

Evaluation — Memory Efficiency (kernel)



Eiffel/Carousel:

- 0.1µs granularity
- 6.9GB/2.3GB memory usage

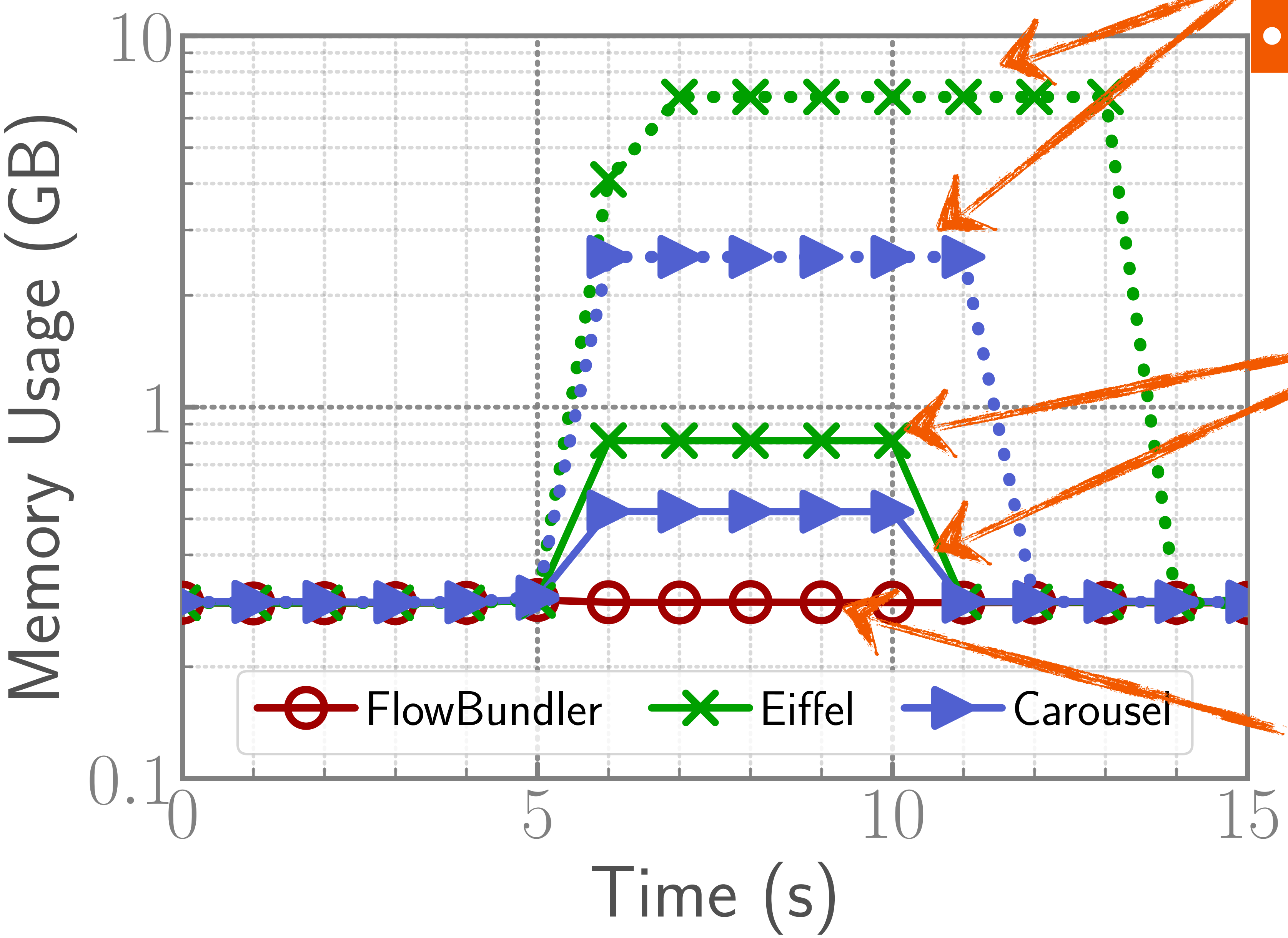
Eiffel/Carousel:

- 1µs granularity
- 0.5GB/0.2GB memory usage

FlowBundler:

- 1ns granularity
- 1.1MB memory usage

Evaluation — Memory Efficiency (kernel)



Eiffel/Carousel:

- 0.1µs granularity
- 6.9GB/2.3GB memory usage

Eiffel/Carousel:

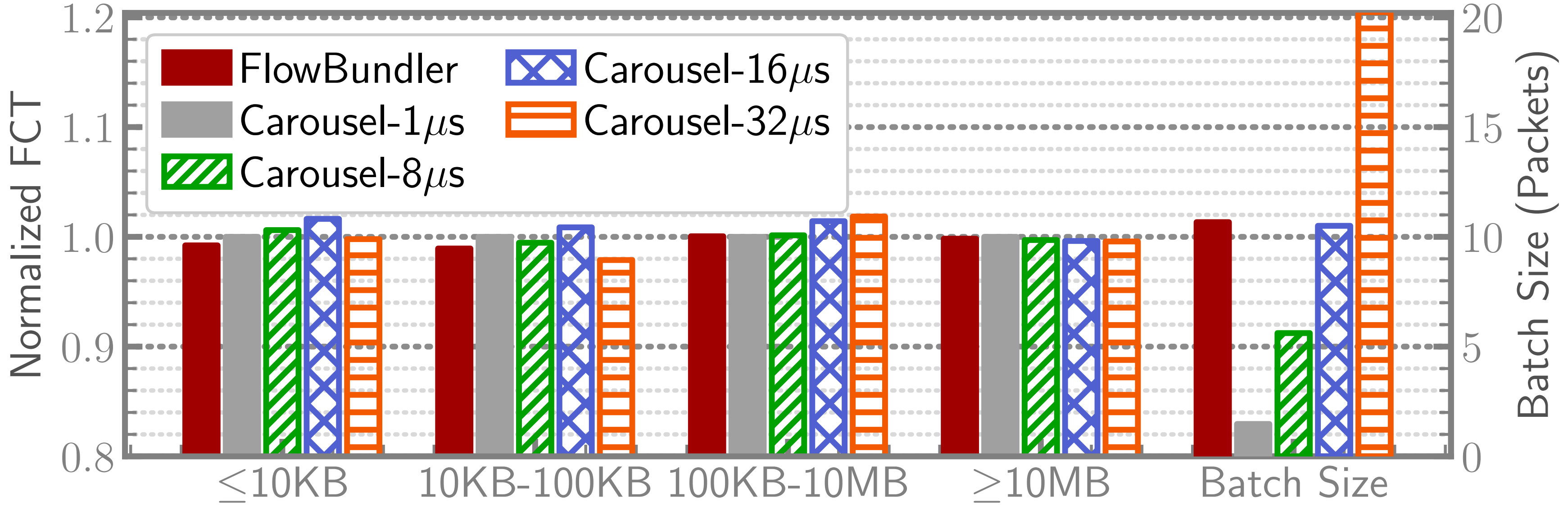
- 1µs granularity
- 0.5GB/0.2GB memory usage

FlowBundler:

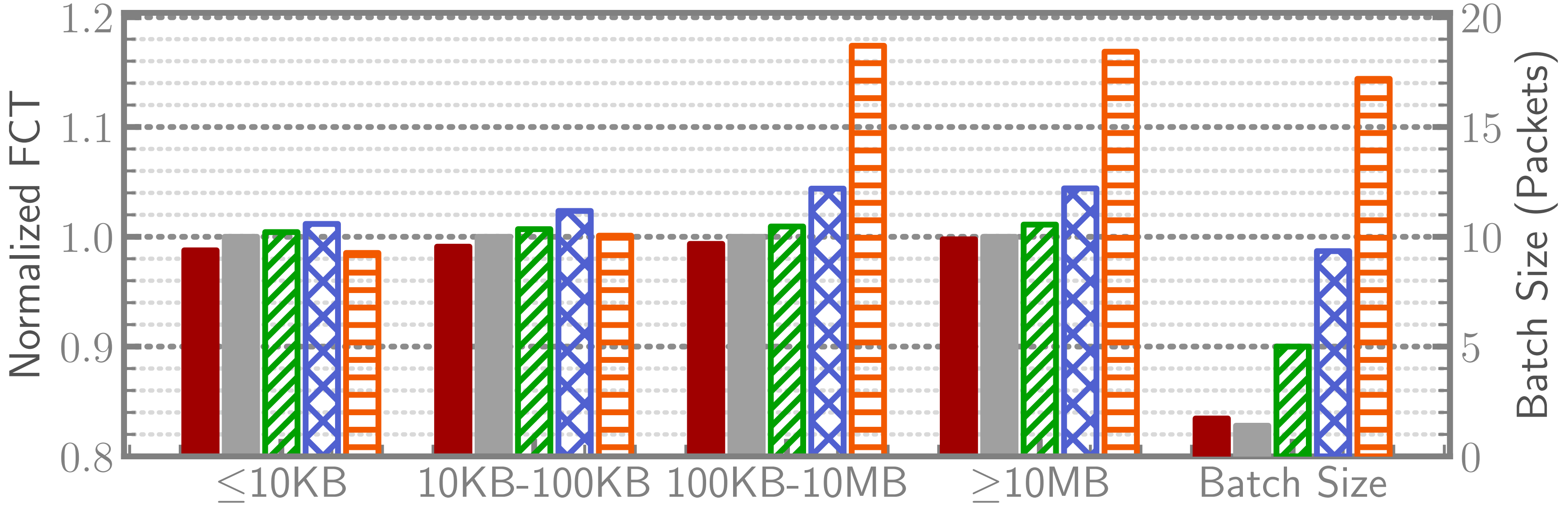
- 1ns granularity
- 1.1MB memory usage

Three orders of magnitude less memory usage

Evaluation — Transmission Performance



Fan-out Flows



One-to-one Flows

Batch packet transmissions without harming transmission performance

Conclusion

- FlowBundler utilizes inter-flow batching to achieve efficient traffic shaping

- FlowBundler utilizes Multi-level Timing Wheel, which can achieve fine-grained shaping while accommodating wide-time-range packets

- FlowBundler can achieve near 100Gbps shaping speed

Questions?