Efficient Policy-Rich Rate Enforcement with Phantom Queues

Ammar Tahir, Prateesh Goyal, Ilias Marinos, Mike Evans, Radhika Mittal

Rate Enforcement on Today's Networks

ISPs

rate-limit users to their subscribed rates

Cellular Service Providers

rate-limit video traffic to manage limited radio resources

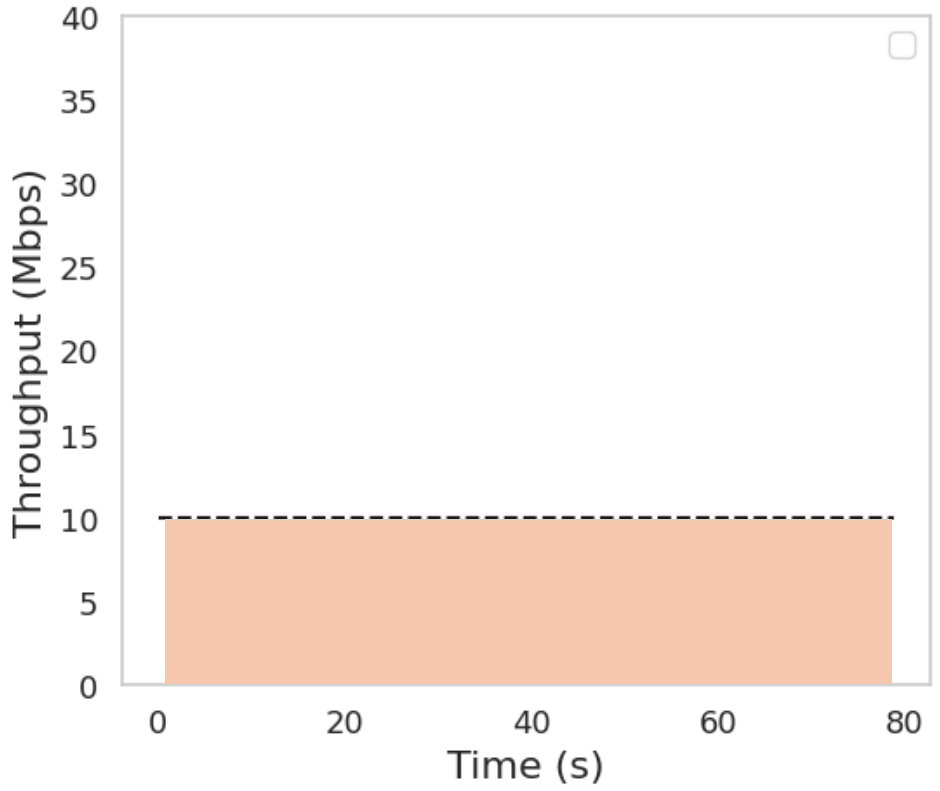
Datacenters

use rate enforcement to actuate TE decisions

Expectations from Rate Enforcement

Effective Enforcement

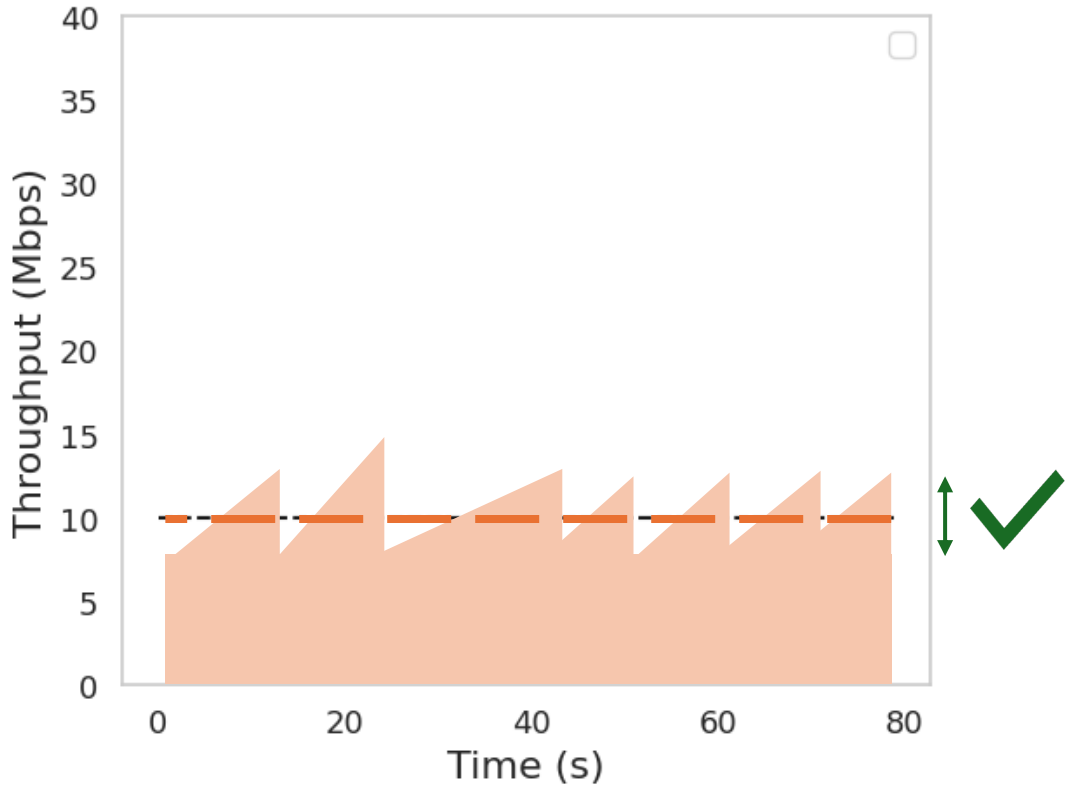
- Correct Rate Enforcement



Expectations from Rate Enforcement

Effective Enforcement

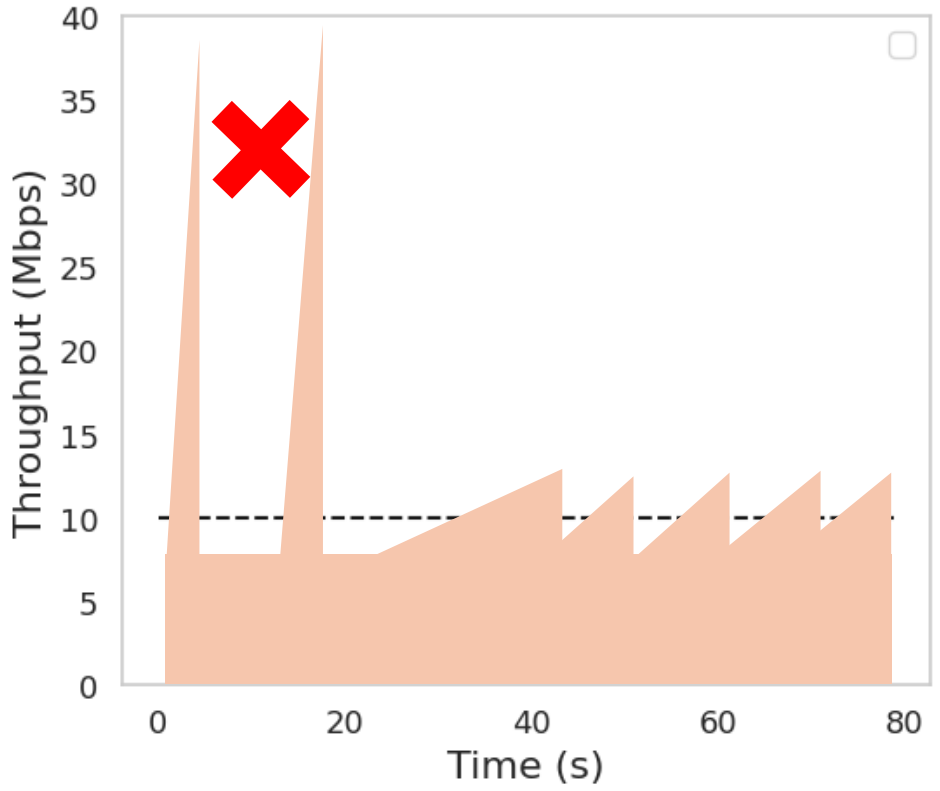
- Correct Rate Enforcement



Expectations from Rate Enforcement

Effective Enforcement

- Correct Rate Enforcement
- No bursts



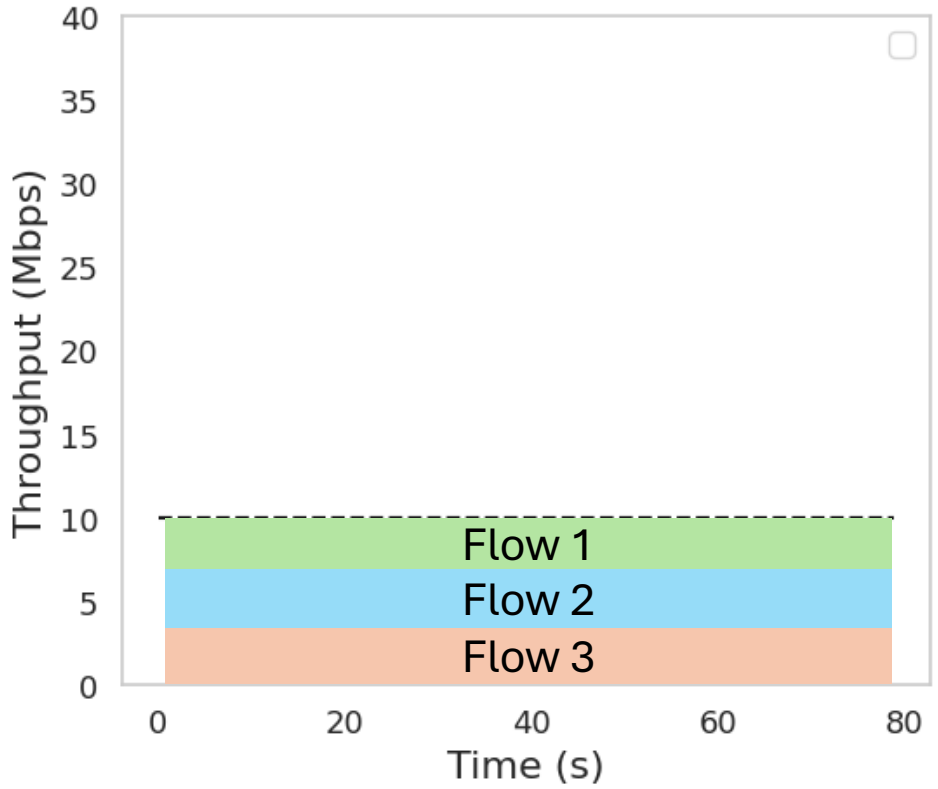
Expectations from Rate Enforcement

Effective Enforcement

- Correct Rate Enforcement
- No bursts

Policy Richness

- Per-flow Fairness,
- Weighted Fairness, Prioritization



Expectations from Rate Enforcement

Effective Enforcement

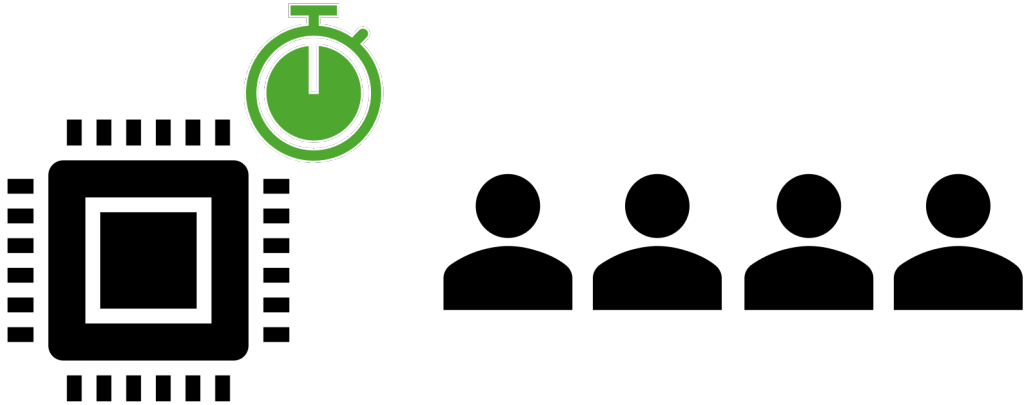
- Correct Rate Enforcement
- No bursts

Policy Richness

- Per-flow Fairness,
- Weighted Fairness, Prioritization etc.

Computational Efficiency

- Low cost
- High scalability



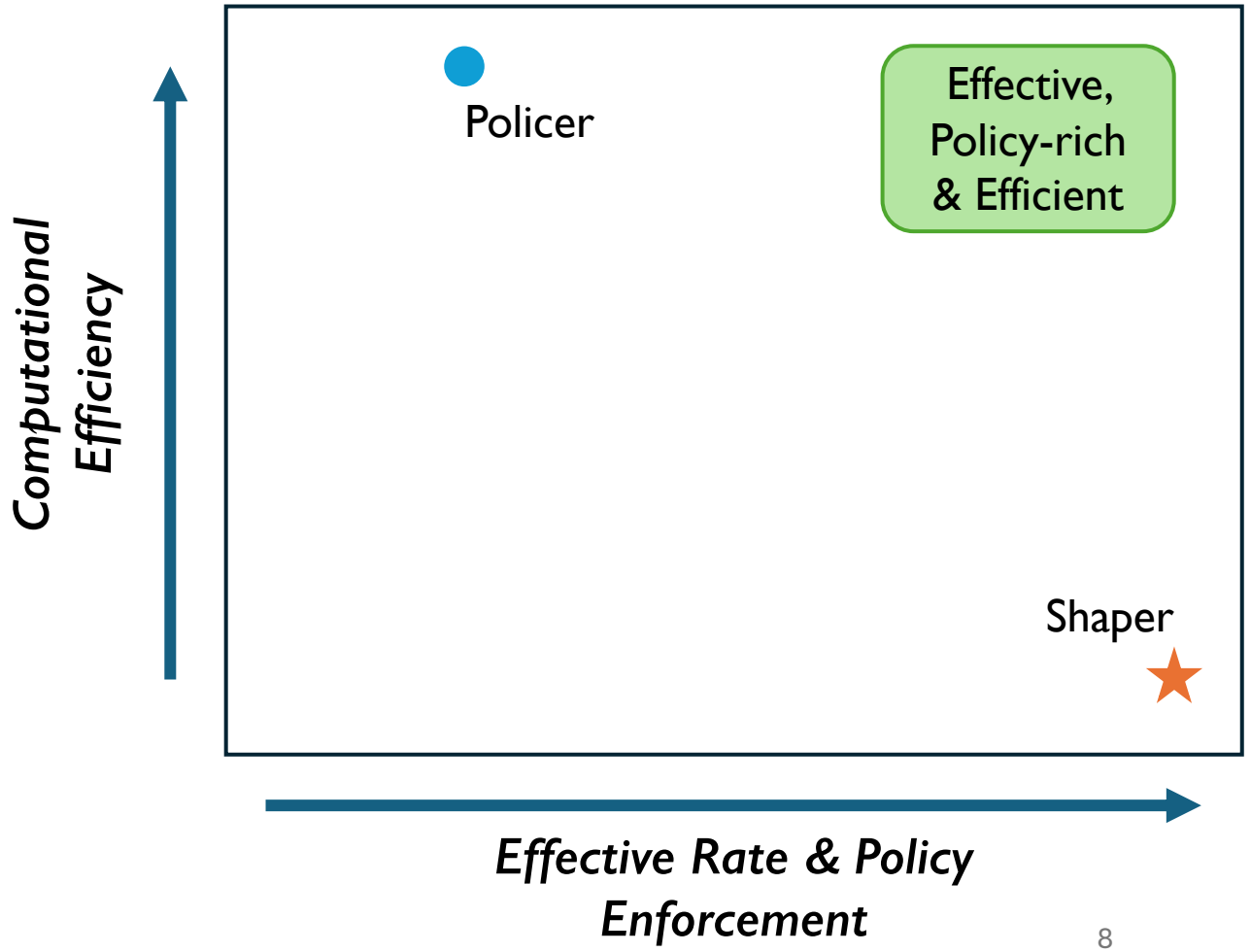
Rate Enforcement Mechanisms

Traffic Shaper

Buffers packets in queues and dequeues them at the desired rate

Traffic Policer

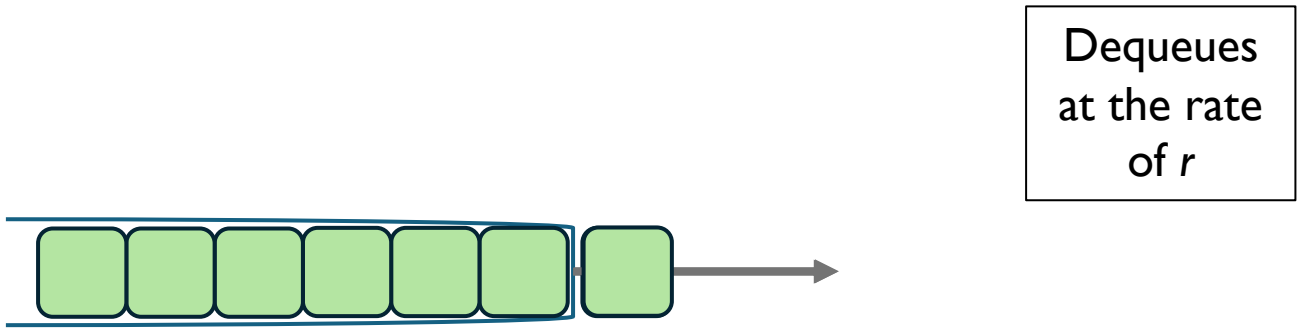
Drops packets exceeding the desired rate without buffering packets



Traffic Shapers

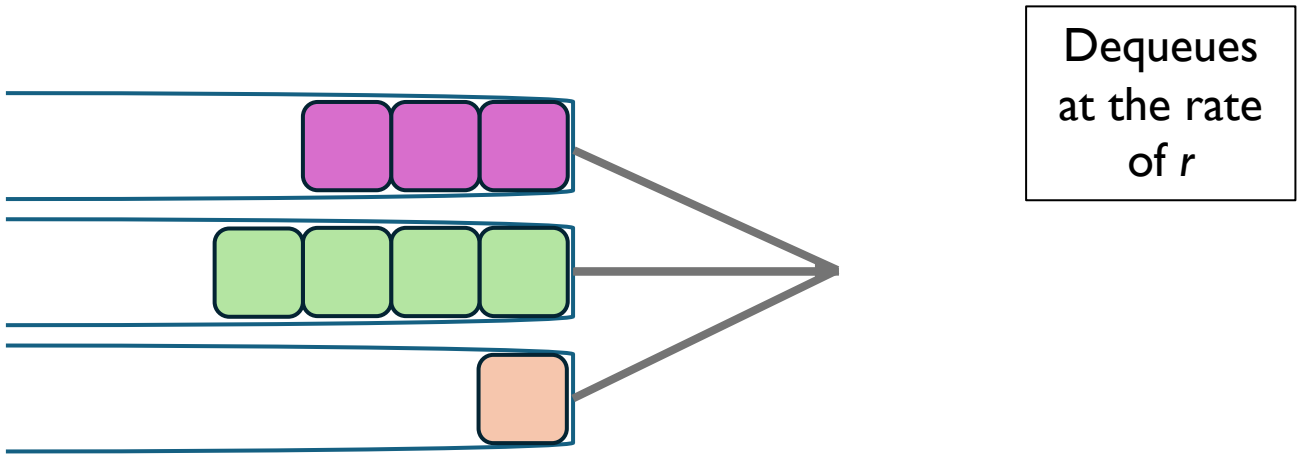
Incoming packets are buffered into a queue.

Traffic Shapers



Packets are dequeued from the queue at the given rate *r*.

Traffic Shapers

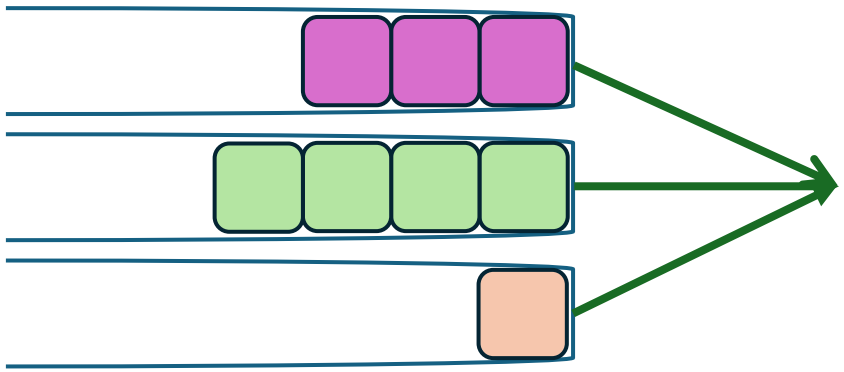


r can be divided across flows using multiple queues to support rate sharing policies.

Traffic Shapers

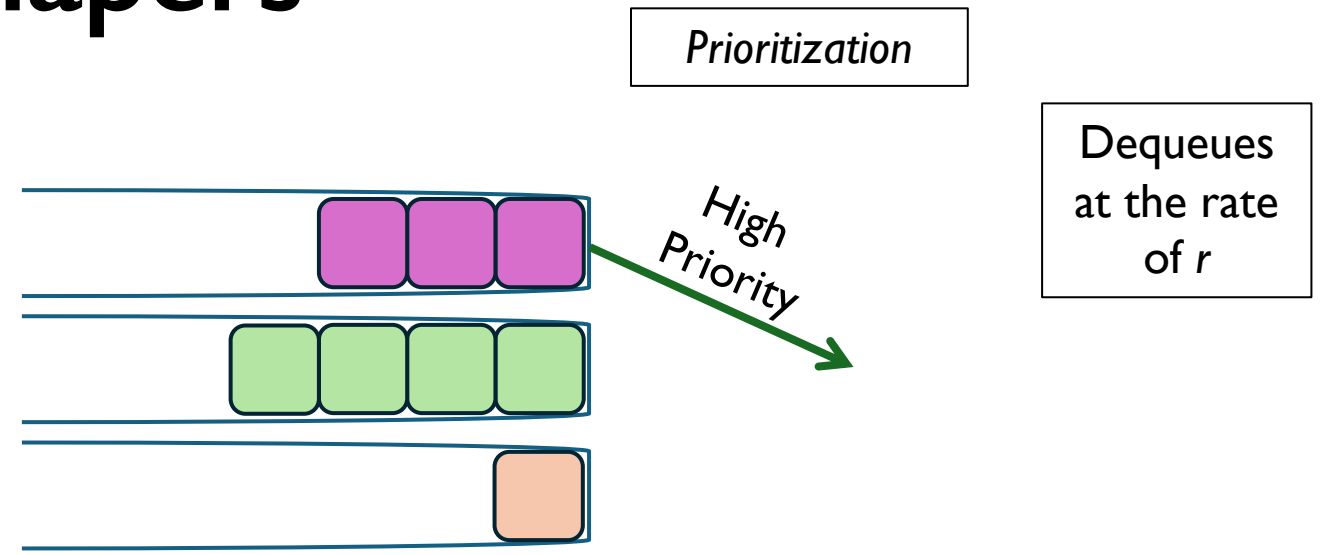
Round-robin for fairness

Dequeues at the rate of *r*



r can be divided across flows using multiple queues to support rate sharing policies.

Traffic Shapers

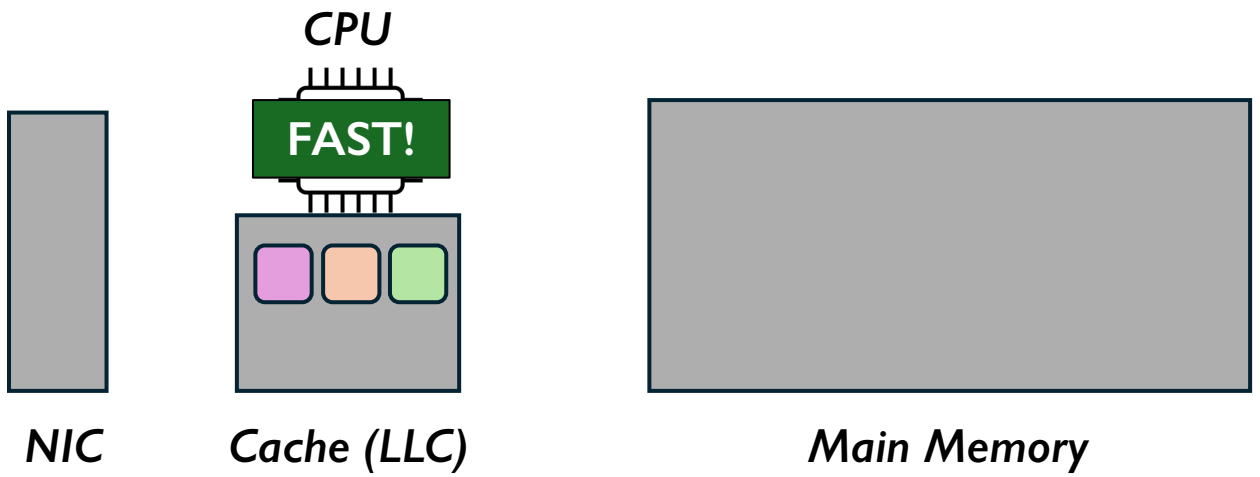


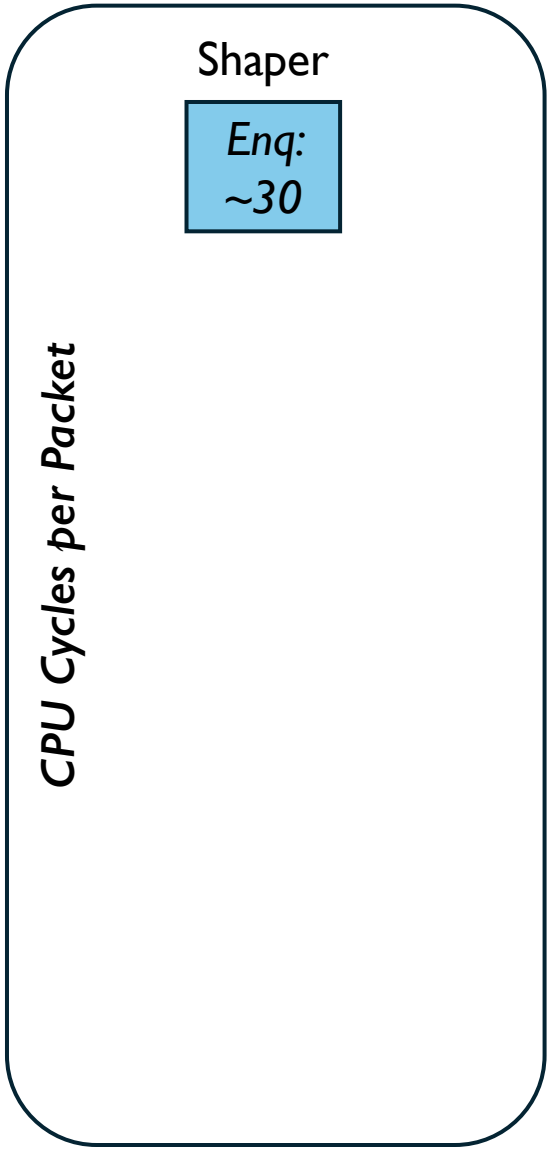
r can be divided across flows using multiple queues to support rate sharing policies.

Traffic Shapers: Inefficiencies

Enqueue Routine

Classify packets from the incoming batch into shapers and queues





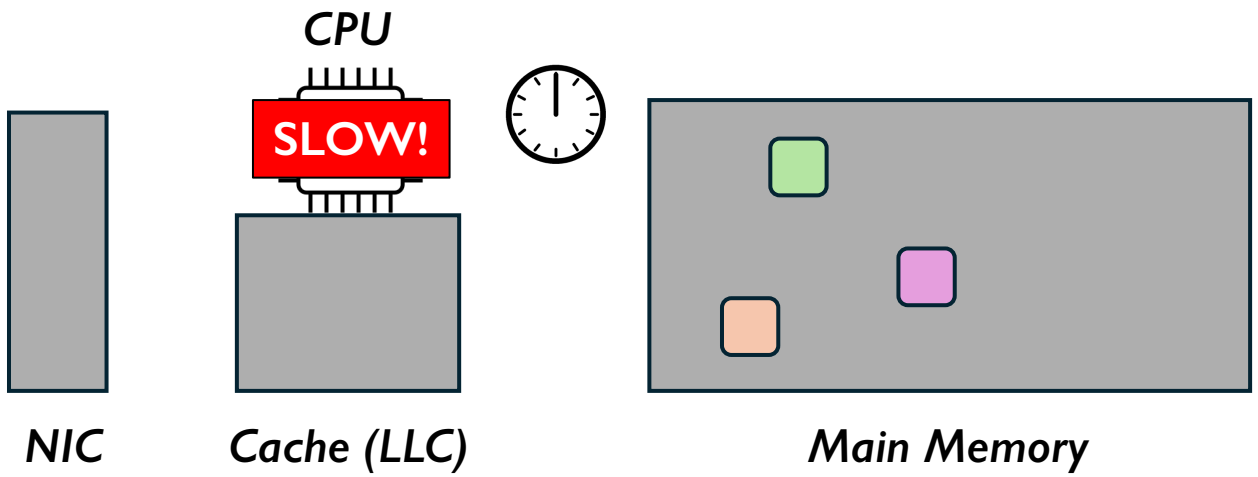
Traffic Shapers: Inefficiencies

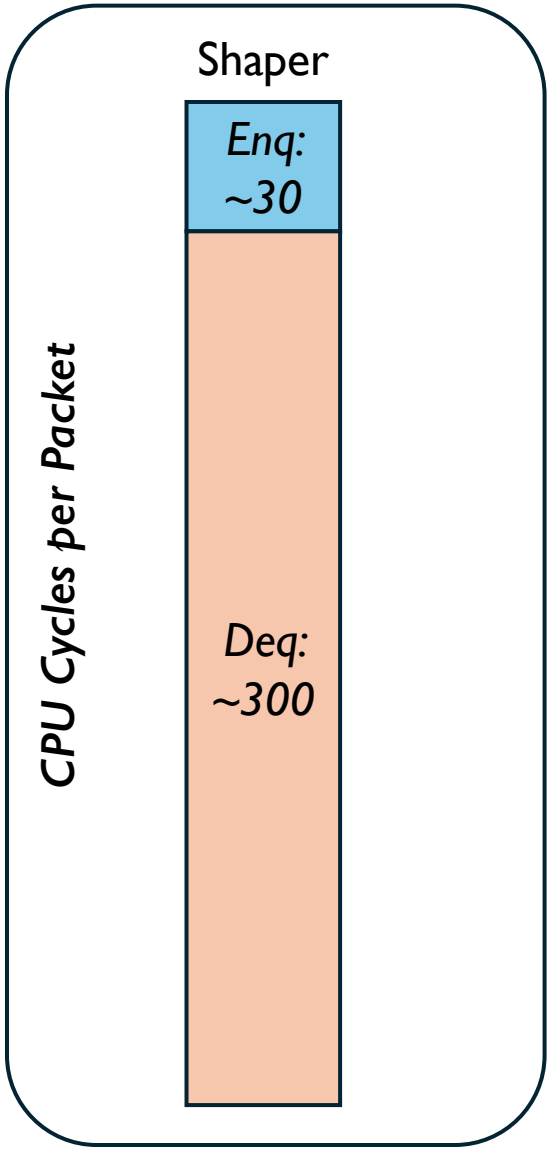
Enqueue Routine

Classify packets from the incoming batch into shapers and queues

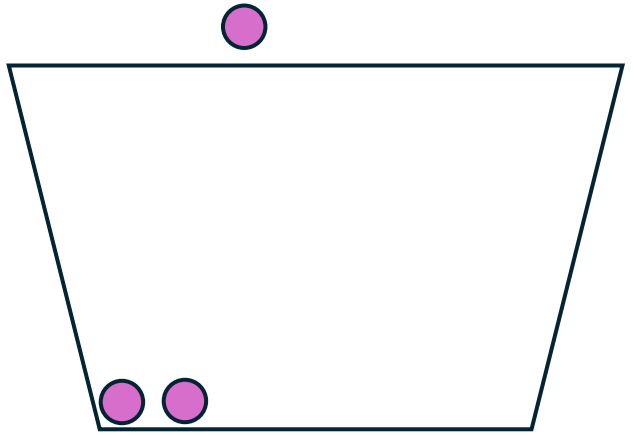
Dequeue Routine

Prepare a batch of packets to dequeue from all shapers



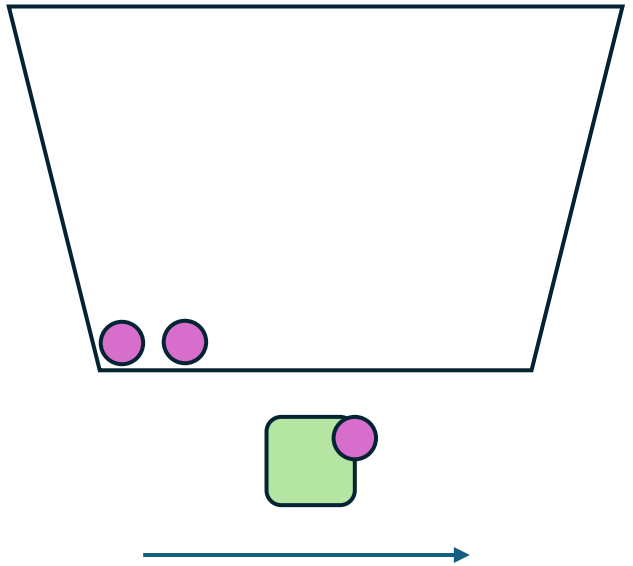


Traffic Policers (with token buckets)



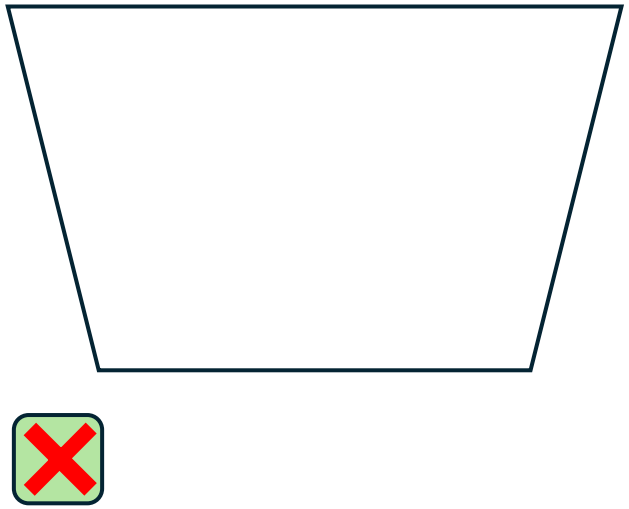
Tokens are added to a bucket at desired rate *r*

Traffic Policers (with token buckets)



Incoming packet is transmitted if there are tokens in the bucket

Traffic Policers (with token buckets)

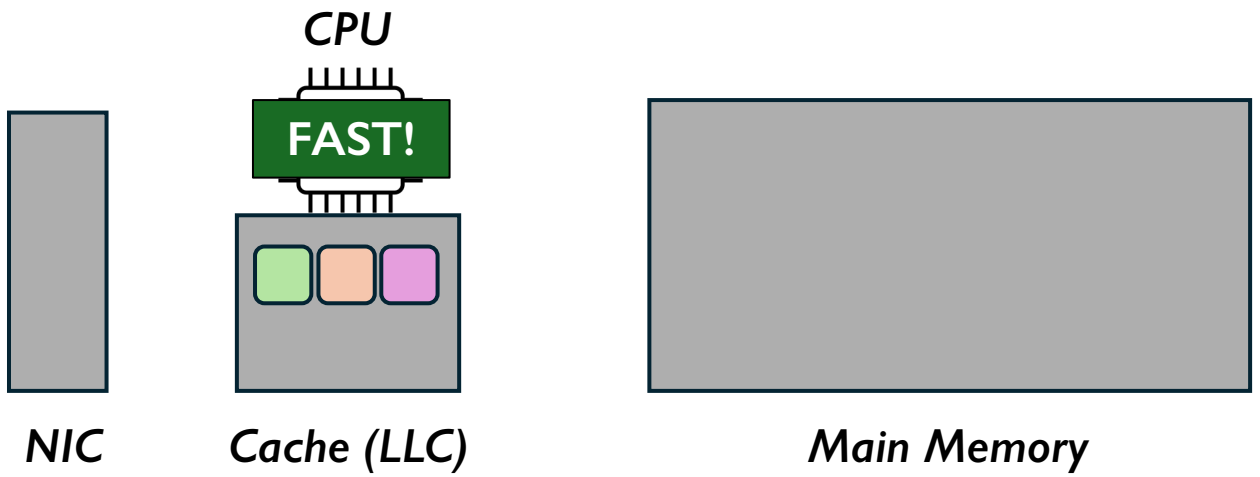


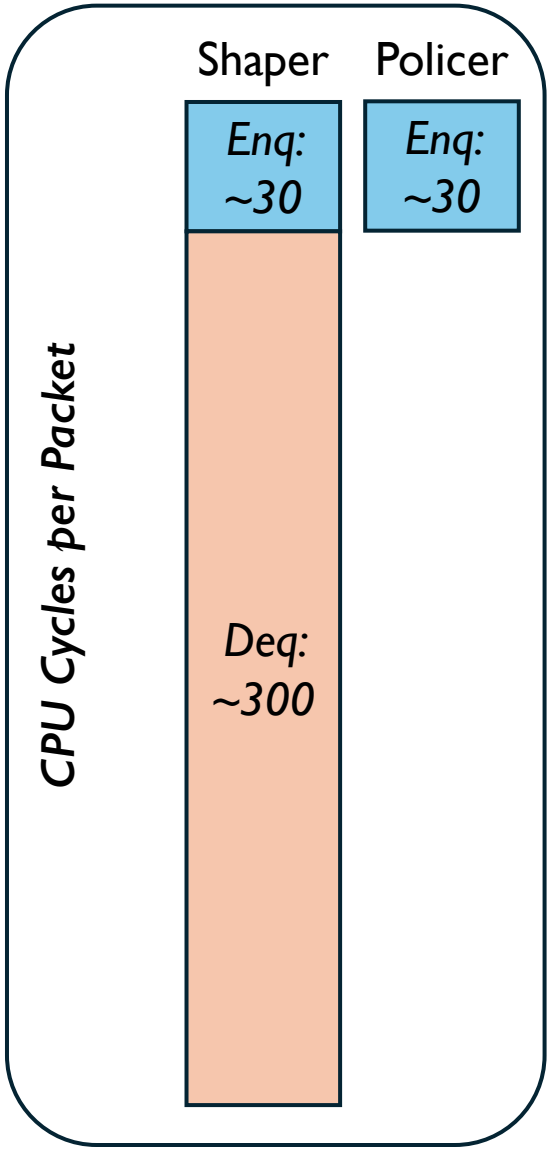
Incoming packet is dropped if there are no tokens in the bucket

Traffic Policers: CPU Cycles

Policers do not need a dequeue routine!

Decision about dropping or transmitting the packet is made at the time of enqueue.

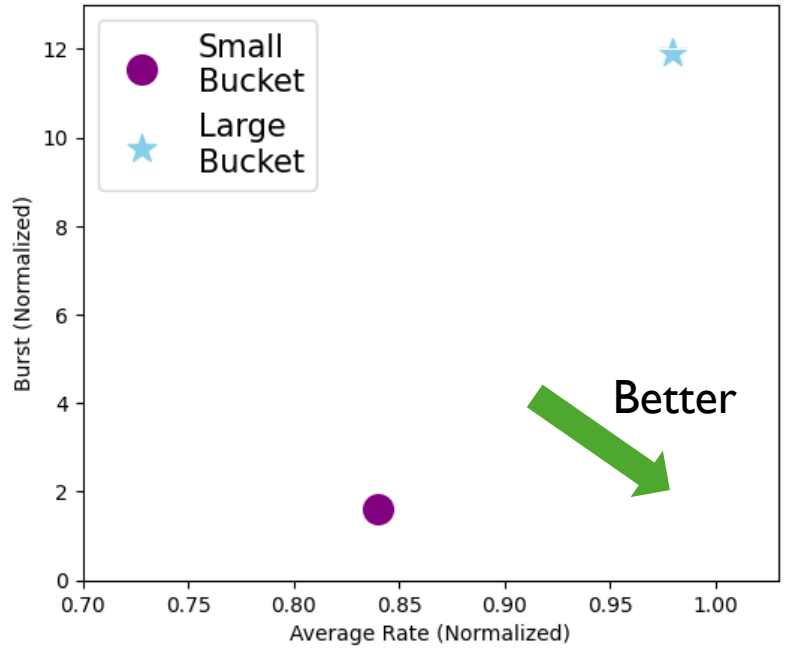




Traffic Policers: Drawbacks

Poor Rate Enforcement

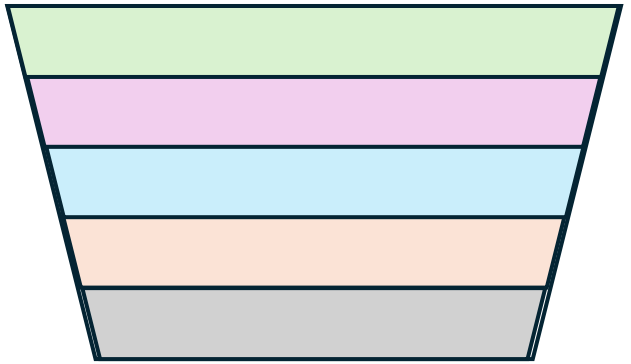
- Configuring a token bucket is hard…
- Choose one:
  - ***Under Enforced Average Rate***
  - *OR* ***Large Bursts***



Traffic Policers: Drawbacks

Poor Rate Enforcement

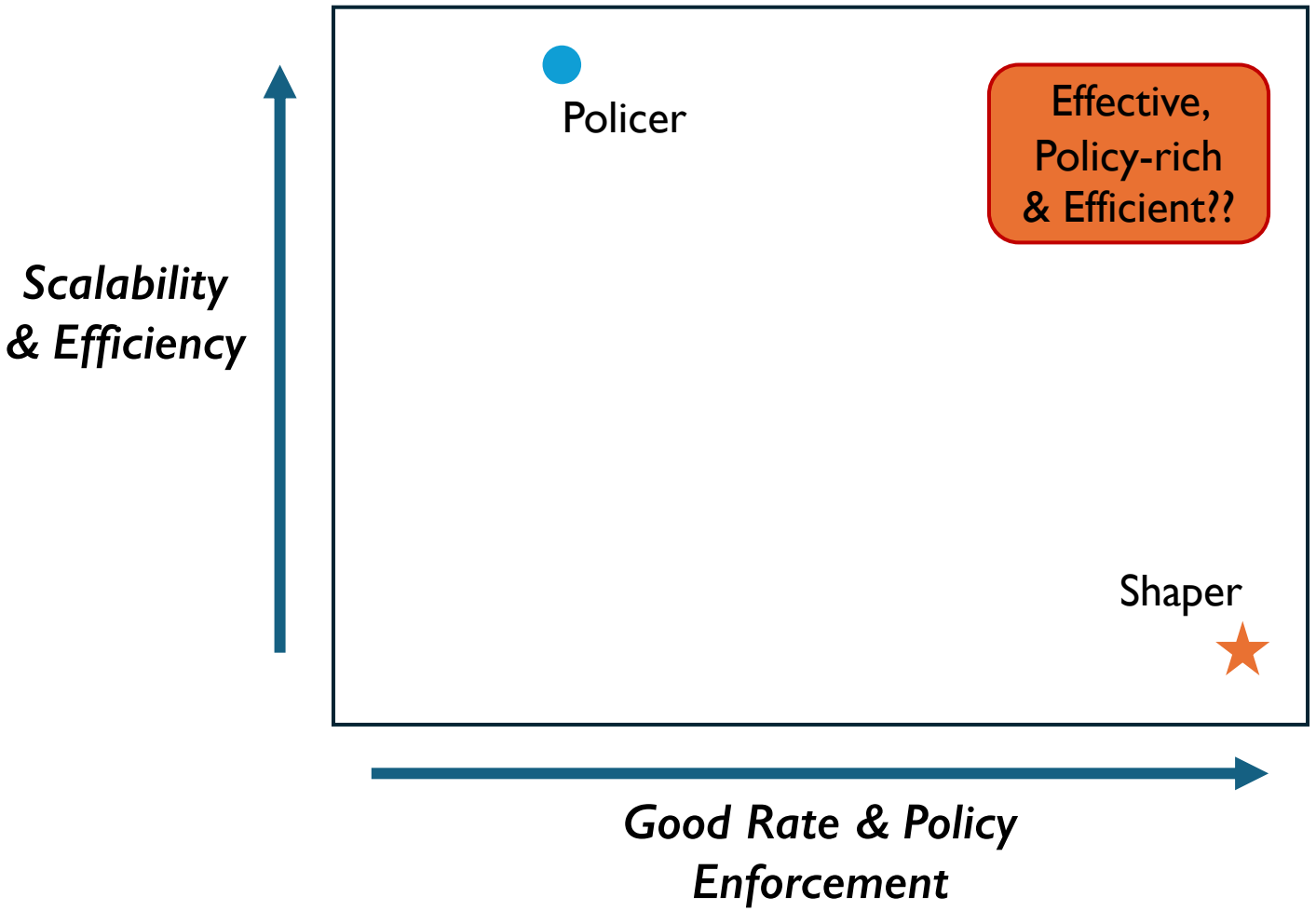
- Configuring a token bucket is hard…
- Choose one:
  - ***Under Enforced Average Rate***
  - *OR* ***Large Bursts***



No Policy Richness

- Lack of support for rich rate-sharing policies
- FairPolicer: Point solution to make token buckets fair
  - Does not generalize to other policies

Can we get efficiency, good rate enforcement and policy richness together?



Our System: BC-PQP

BC-PQP: *Burst Controlled – Phantom Queue Policer*

Policy Rich

e.g. fairness, prioritization

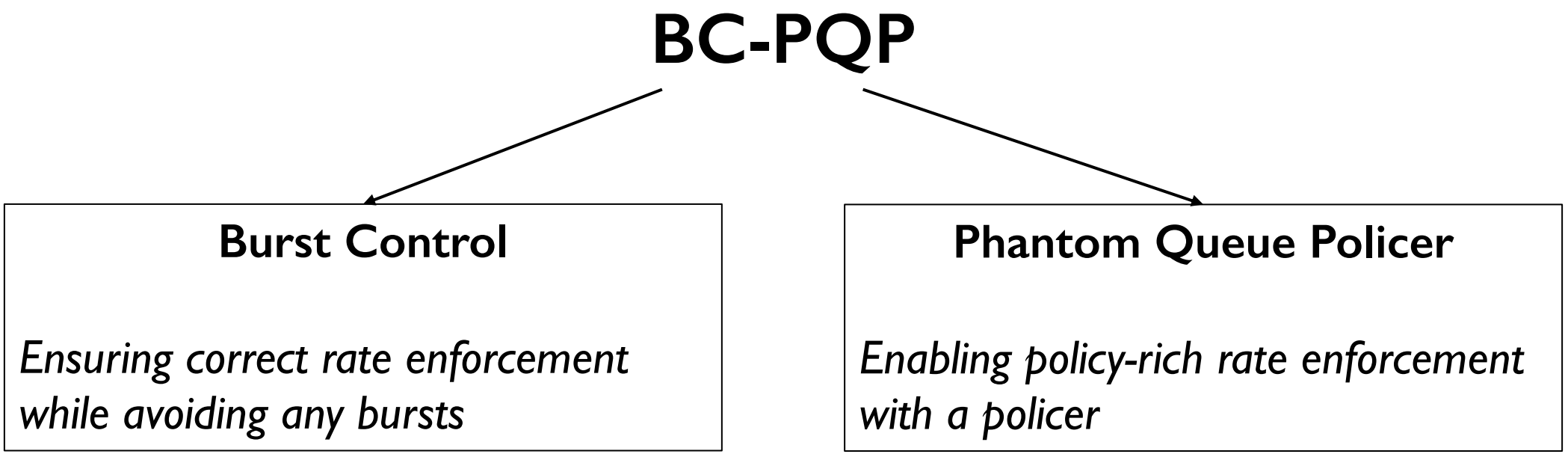
Burst Free & Correct

Smaller than 2x burst

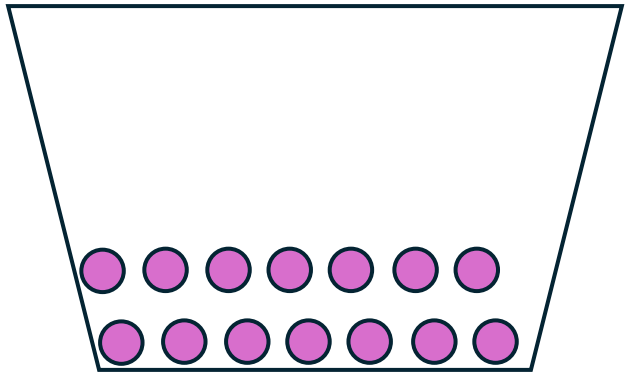
Efficient

7x better than shaper

Our System

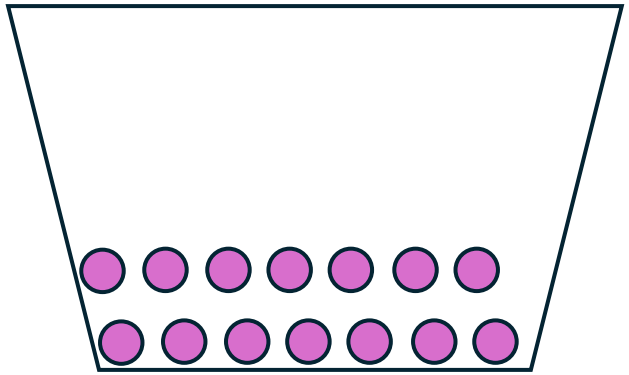

BC-PQP
Burst Control
Ensuring correct rate enforcement while avoiding any bursts
Phantom Queue Policer
Enabling policy-rich rate enforcement with a policer

BC-PQP: Enabling Policy-richness

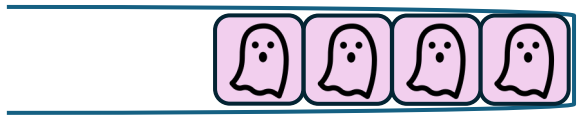


Token Bucket

BC-PQP: Enabling Policy-richness



Token Bucket

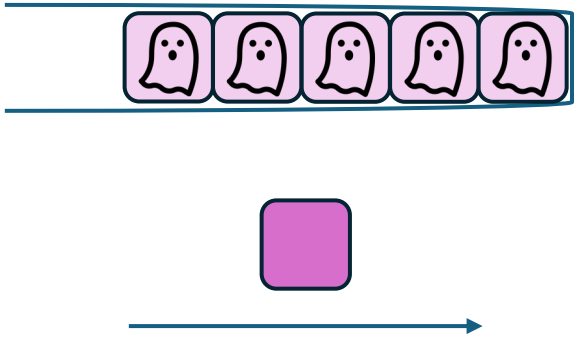


Phantom Queue

Phantom Queue

Phantom queue is a *byte-counter* that simulates a real queue

Phantom Queue



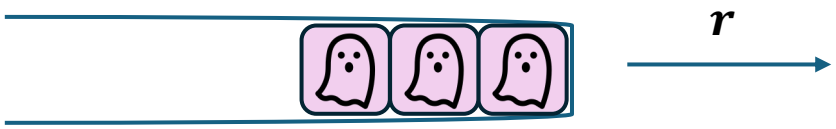
Real packets are never stored!

Upon arrival of a packet, a phantom packet is added to the queue but real packet is transmitted immediately.

Phantom Queue

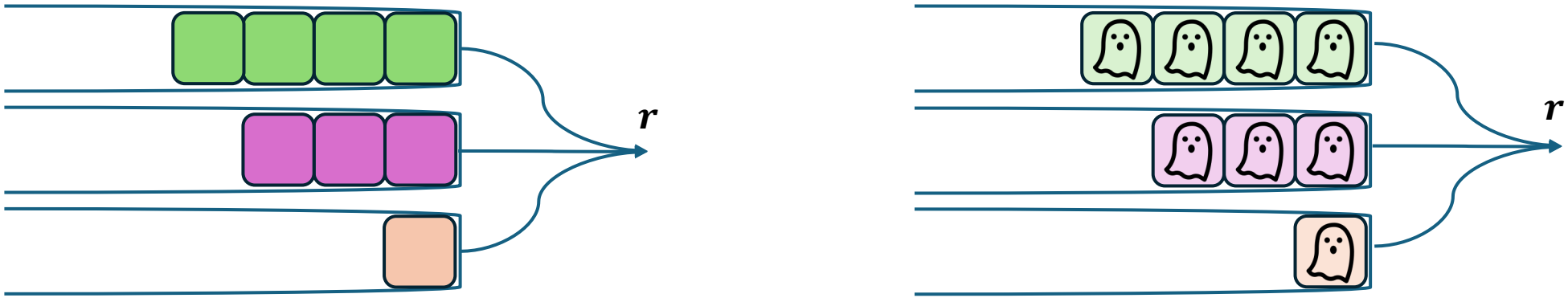
If the phantom queue is full, incoming packet is dropped

Phantom Queue



Phantom queue is dequeued at rate r, only phantom packets are dequeued

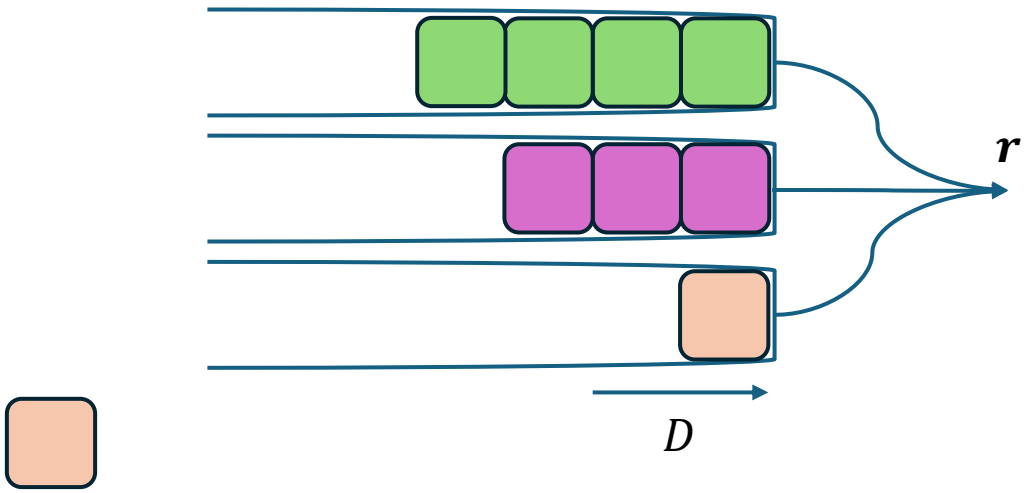
BC-PQP: Enabling Policy-richness

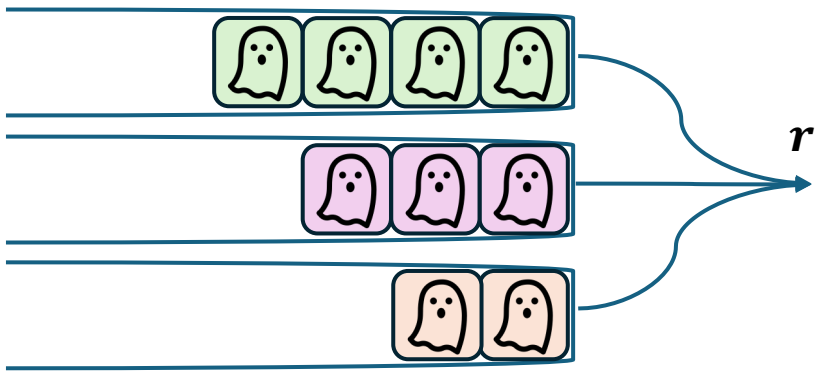


Just like a shaper, we can combine multiple queues to support different rate-sharing policies

e.g. round-robin for fairness

BC-PQP: vs Shaper





Enqueue time: t_e

Shaper

Dequeue time: $t_e + D$

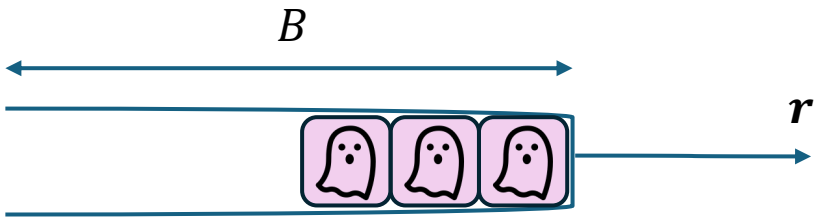
PQP

Dequeue time: t_e

No dequeue routine → High efficiency

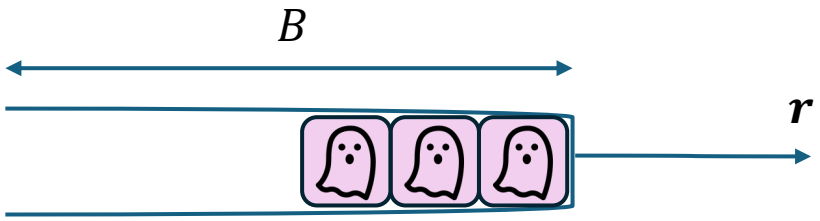
Error in rate and policy enforcement

BC-PQP: Rate Enforcement Guarantees



Over any time window Δt,

BC-PQP: Rate Enforcement Guarantees



$$max(0, r \pm \frac{B}{\Delta t})$$

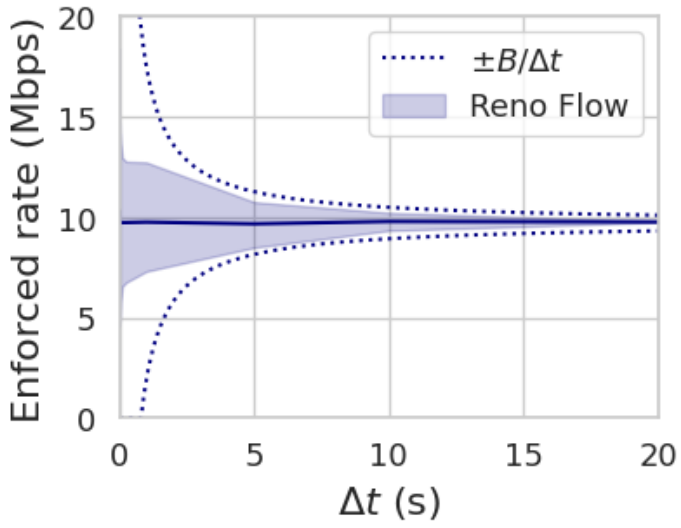
Over any time window Δt,

if phantom queue occupancy does not go to zero

BC-PQP: Rate Enforcement Guarantees

$$max(0, r \pm \frac{B}{\Delta t})$$

Error amortizes as we increase Δt



BC-PQP: Rate Enforcement Guarantees

$$max(0, r \pm \frac{B}{\Delta t})$$

- A large B results in larger bursts
- A small B can result in under enforcement

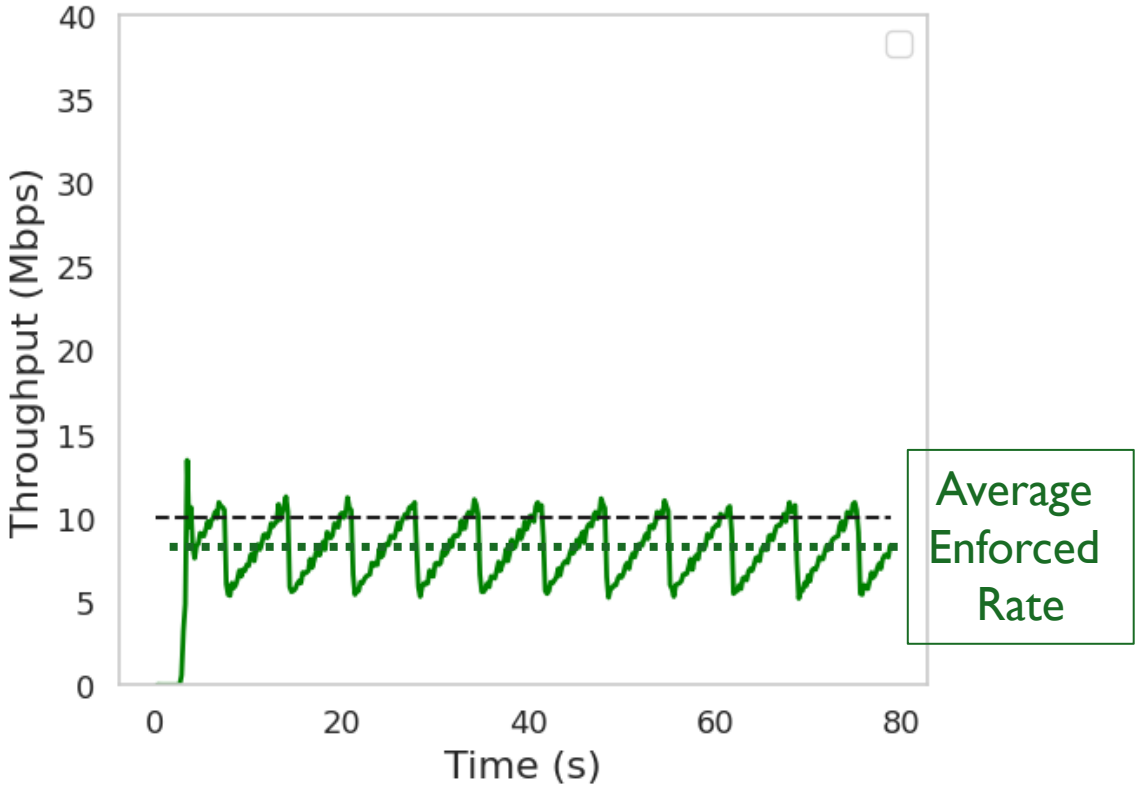
BC-PQP: Rate Enforcement Guarantees

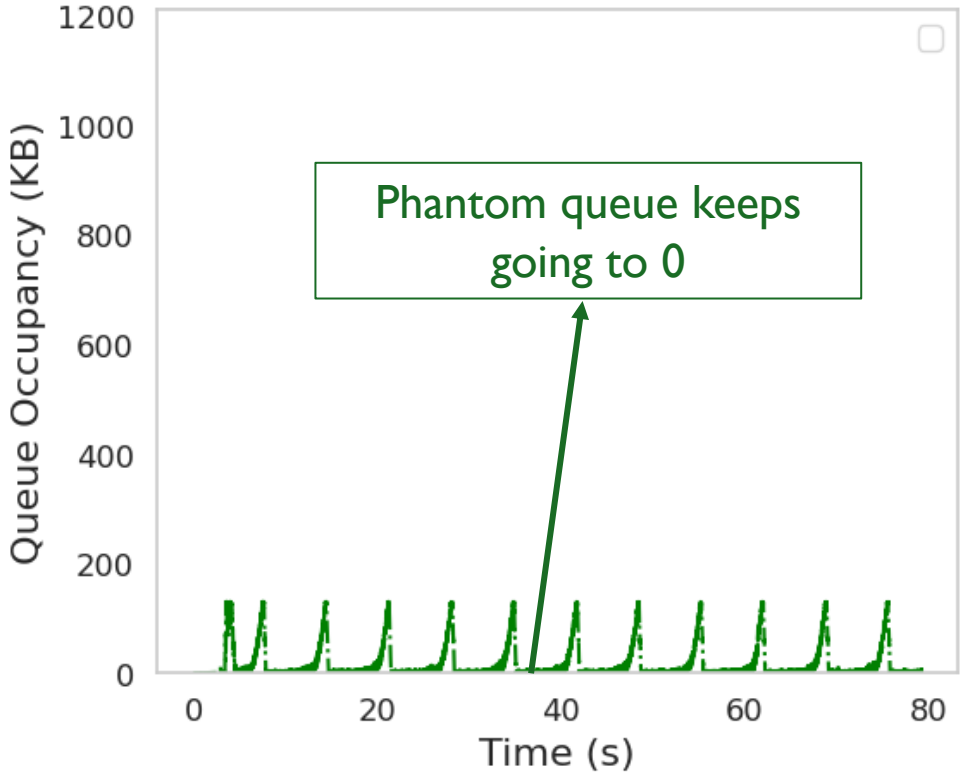
$$max(0, r \pm n\frac{B}{\Delta t})$$

- A large B results in larger bursts
- A small B can result in under enforcement
- Error scales with n, the number of phantom queues

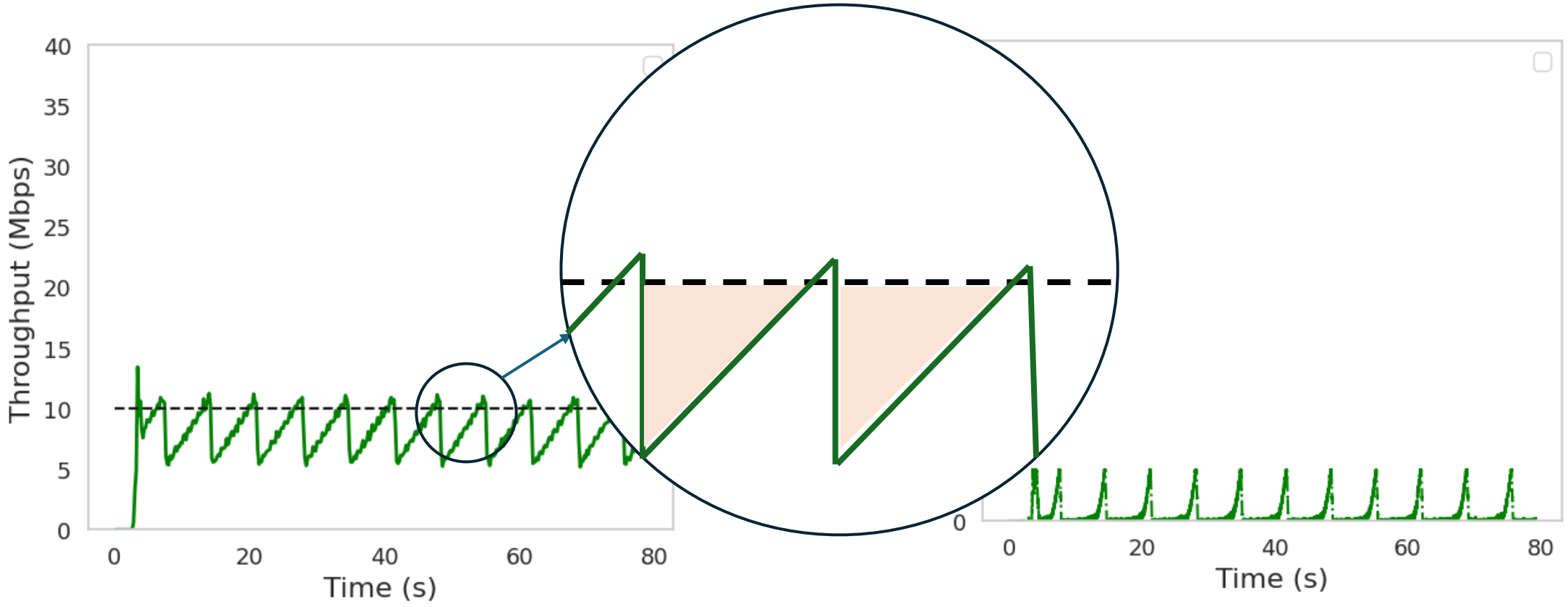
BC-PQP: Sizing Phantom Queues

- Size by $BDP = rate \times RTT$ (as for shaper queues)?

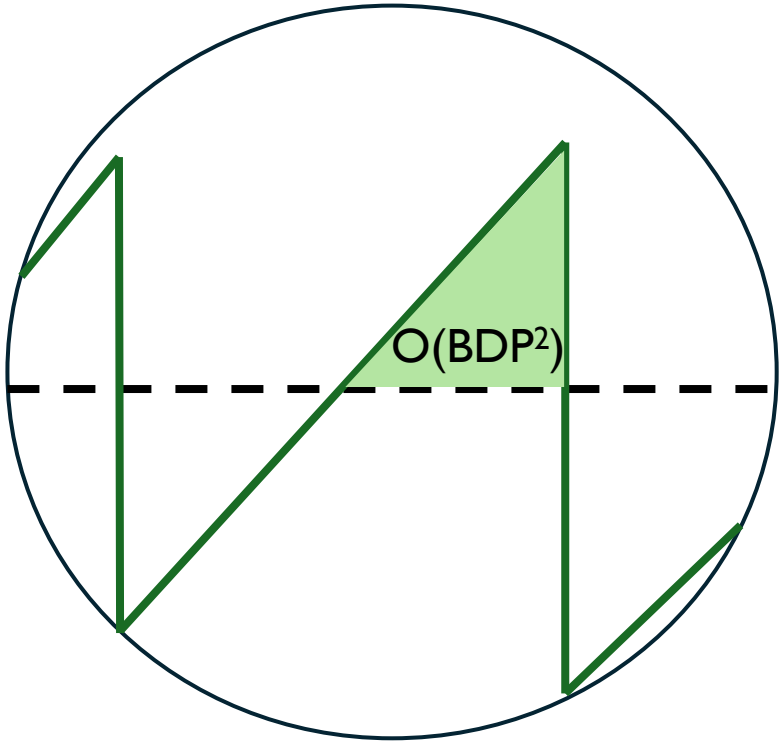




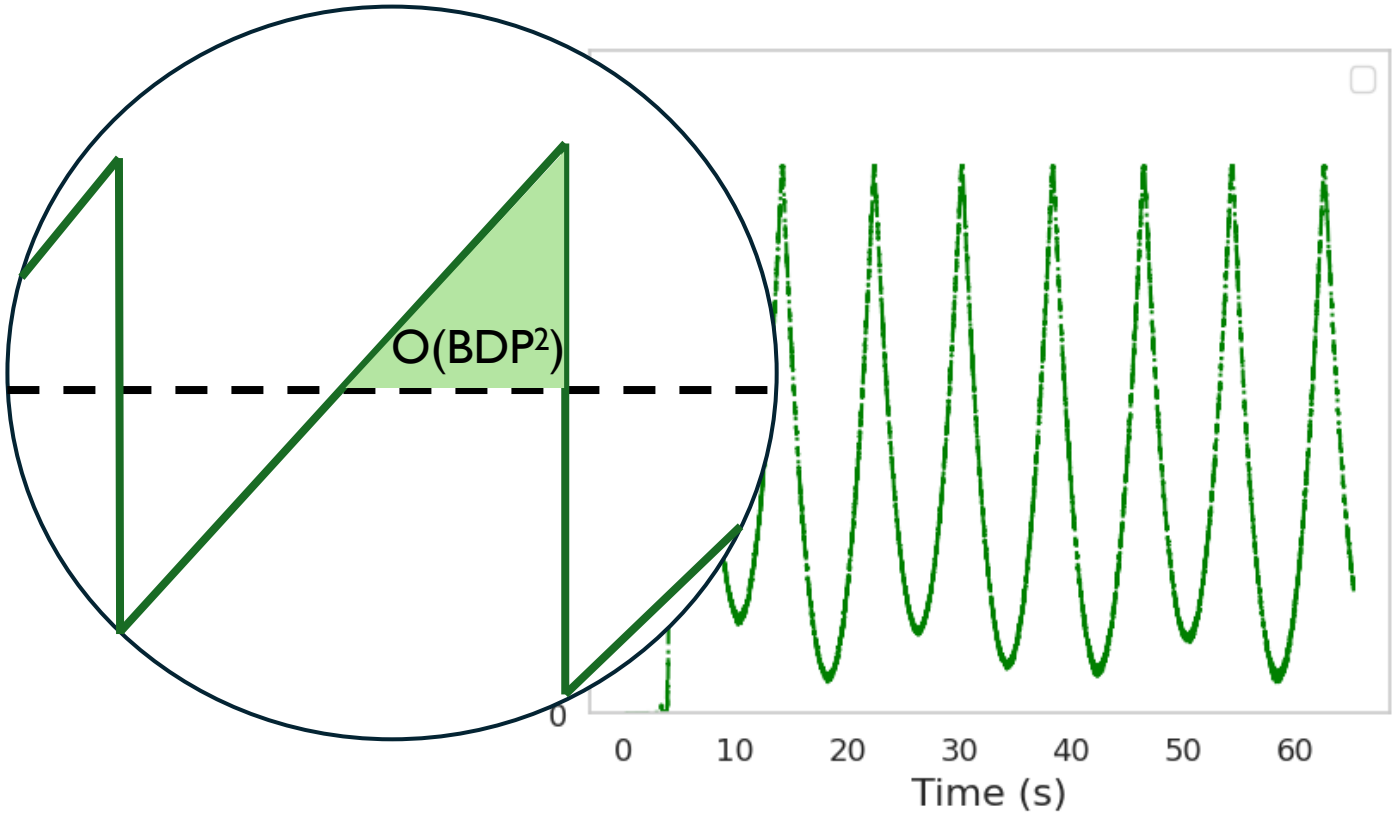
BC-PQP: Sizing Phantom Queues



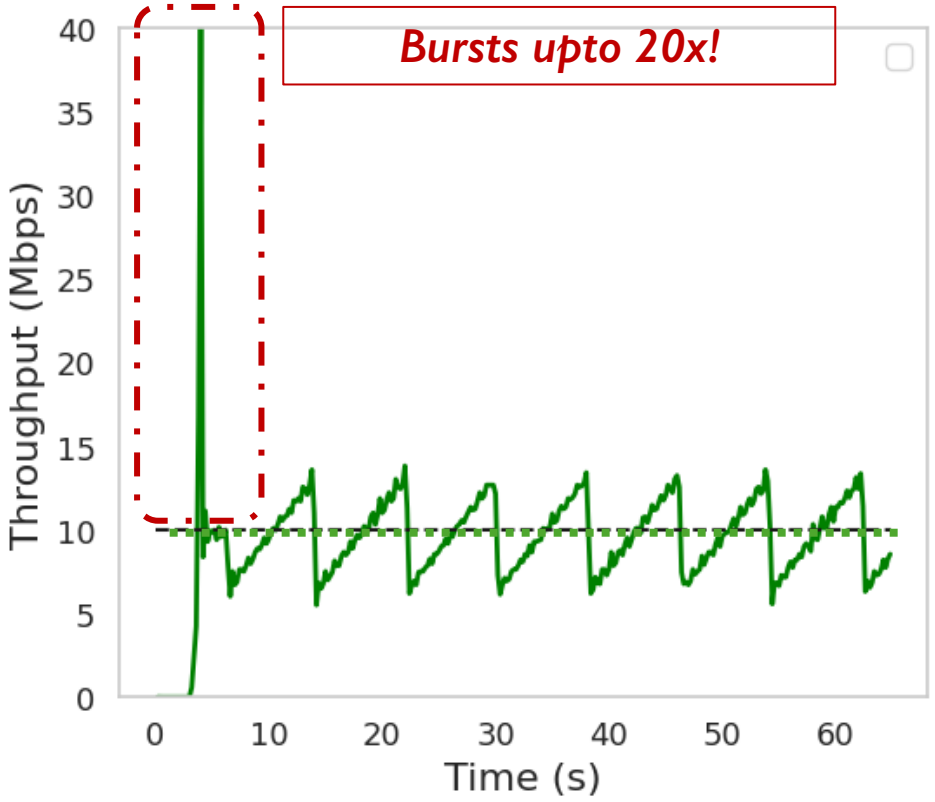
BC-PQP: Sizing Phantom Queues

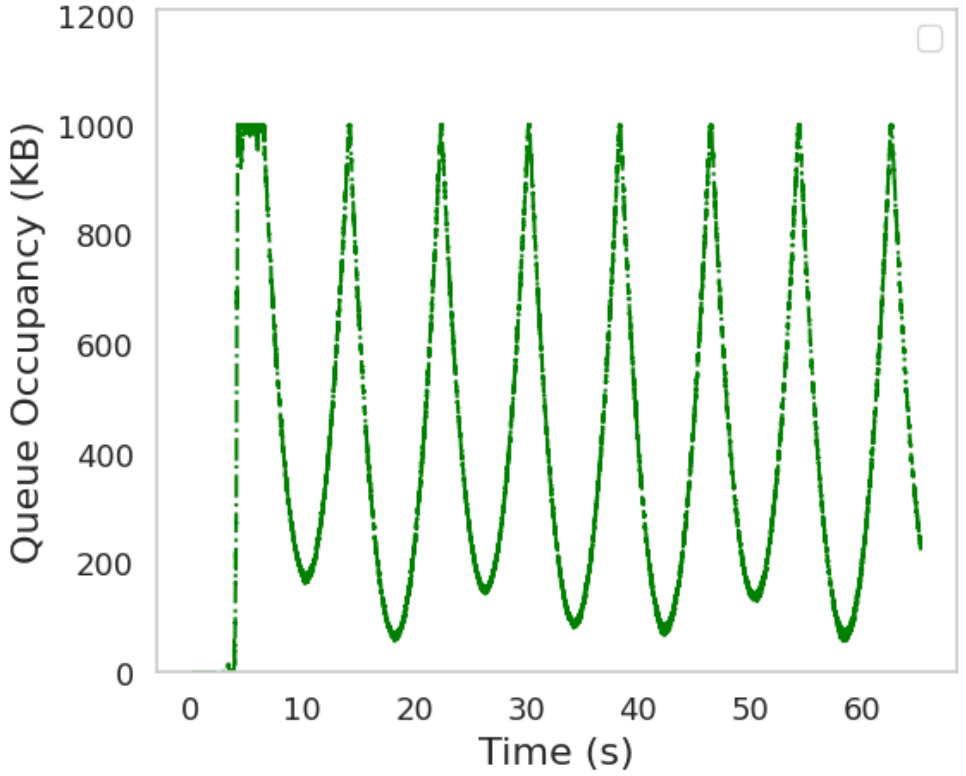


BC-PQP: Sizing Phantom Queues

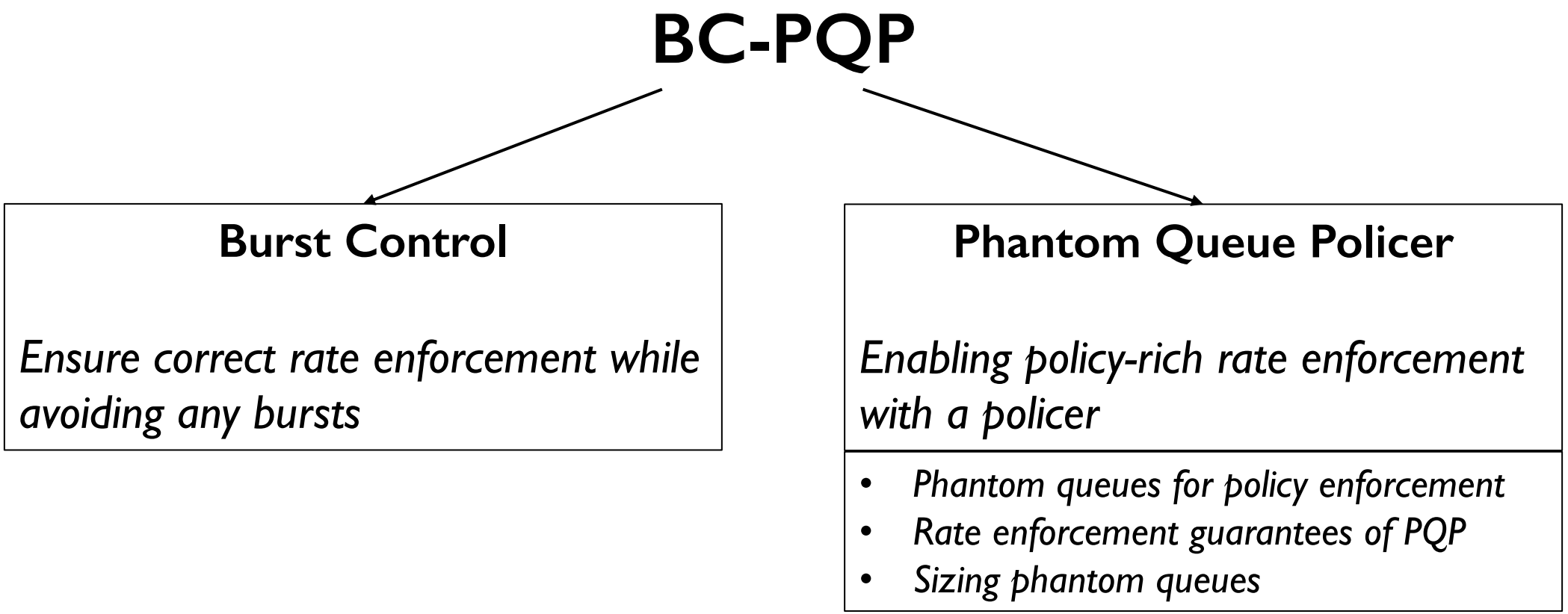


BC-PQP: Sizing Phantom Queues



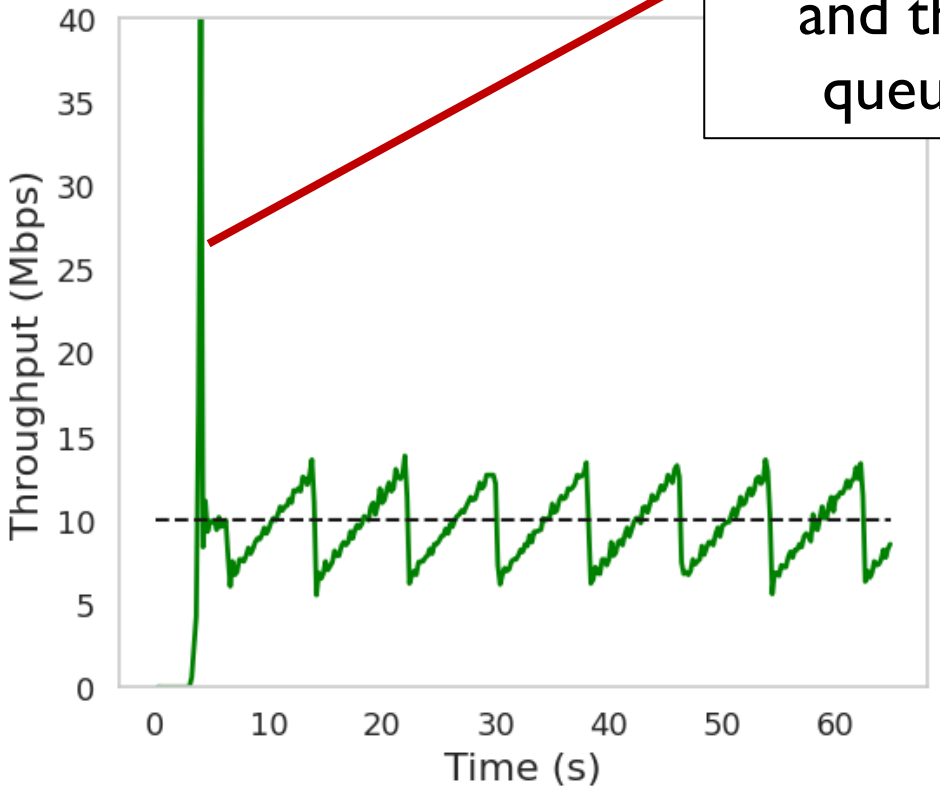


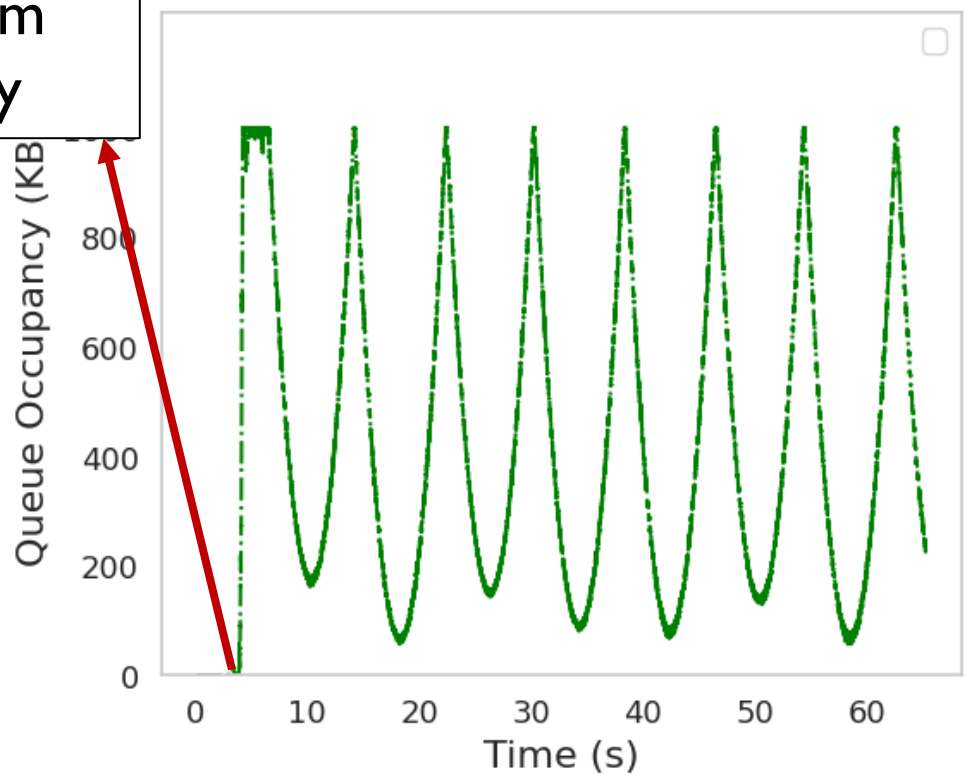
Our System


BC-PQP
Burst Control
Ensure correct rate enforcement while avoiding any bursts
Phantom Queue Policer
Enabling policy-rich rate enforcement with a policer
• Phantom queues for policy enforcement
• Rate enforcement guarantees of PQP
• Sizing phantom queues

BC-PQP: Burst Control

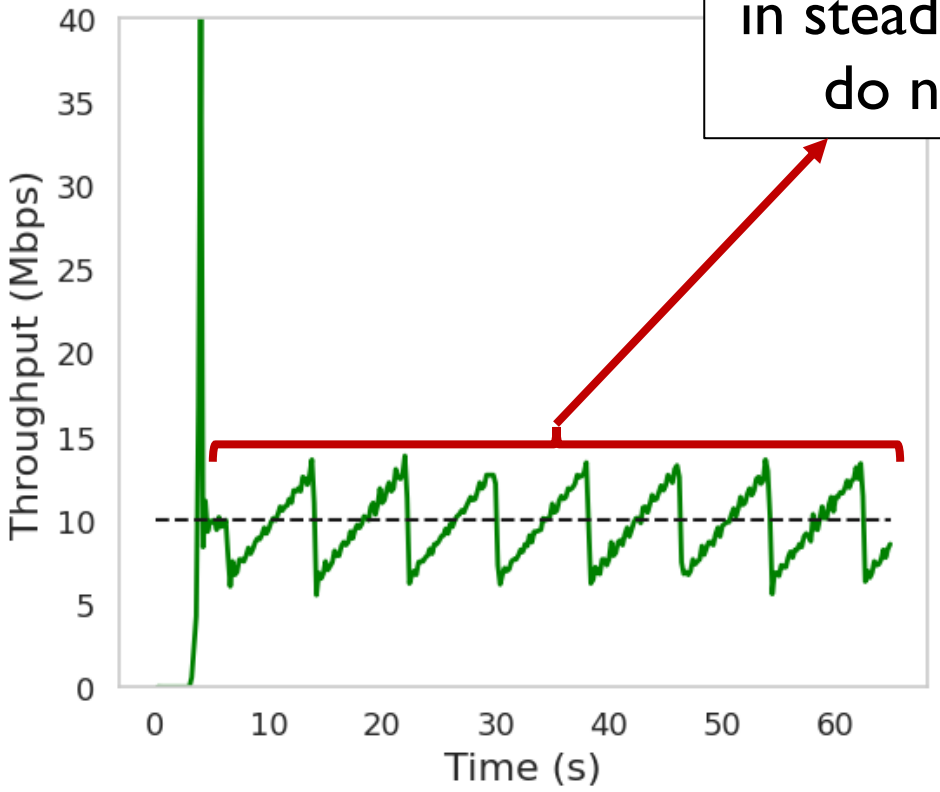
Bursts occur when flow is starting out and the phantom queue is empty

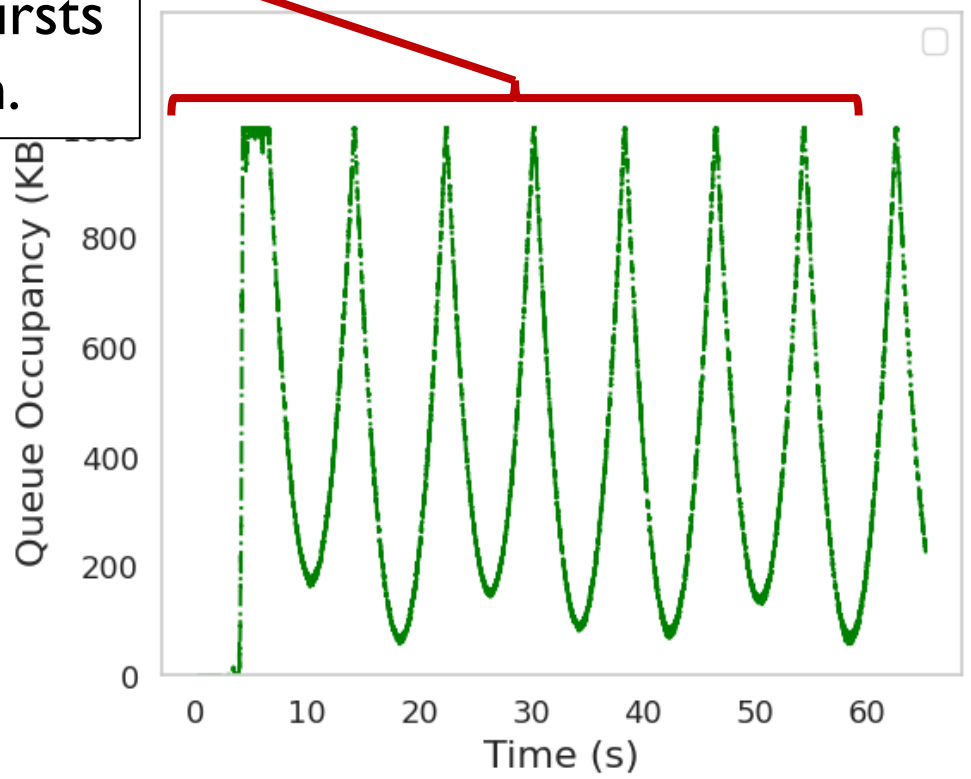




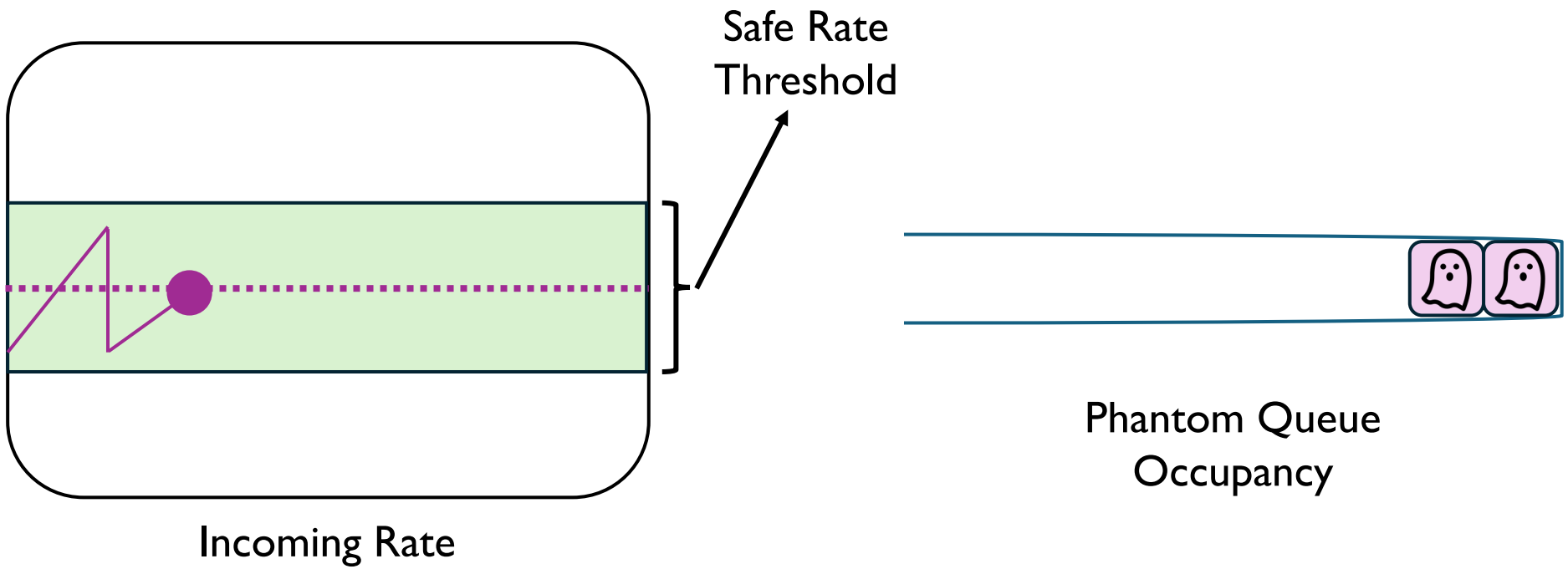
BC-PQP: Burst Control

Once the phantom queue is full and flow in steady state, bursts do not happen.

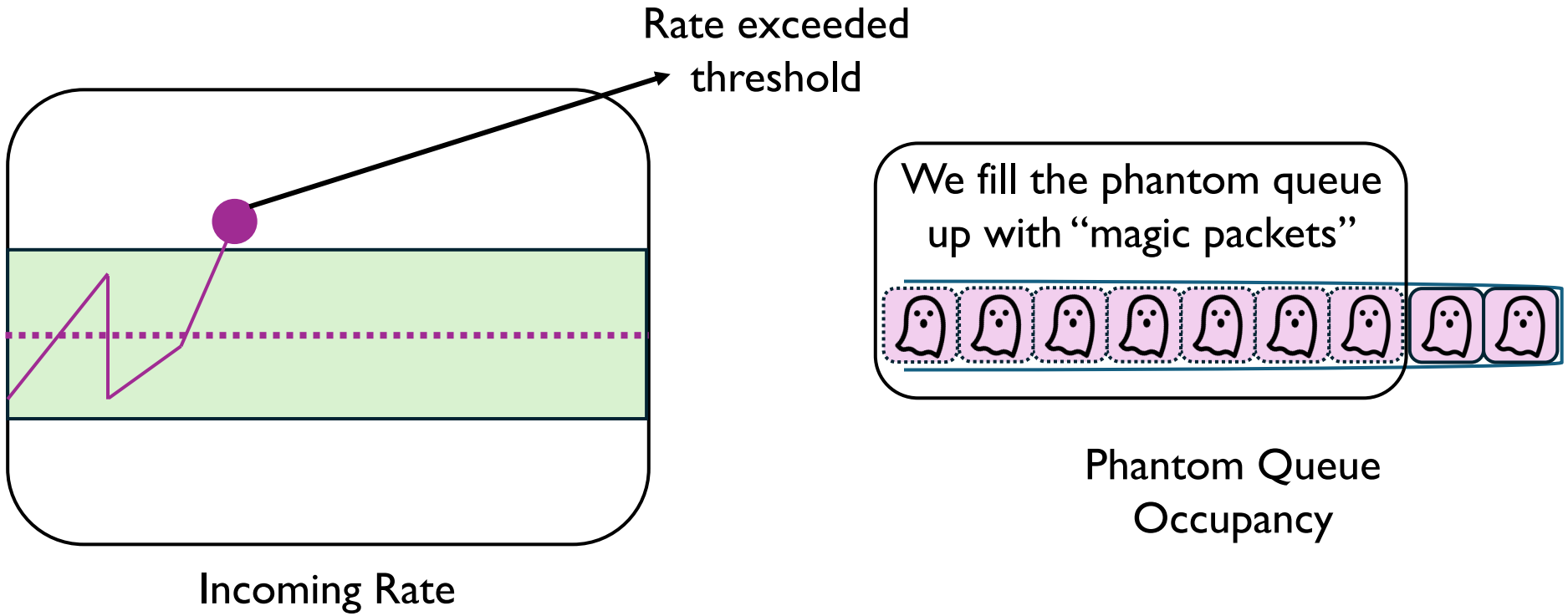




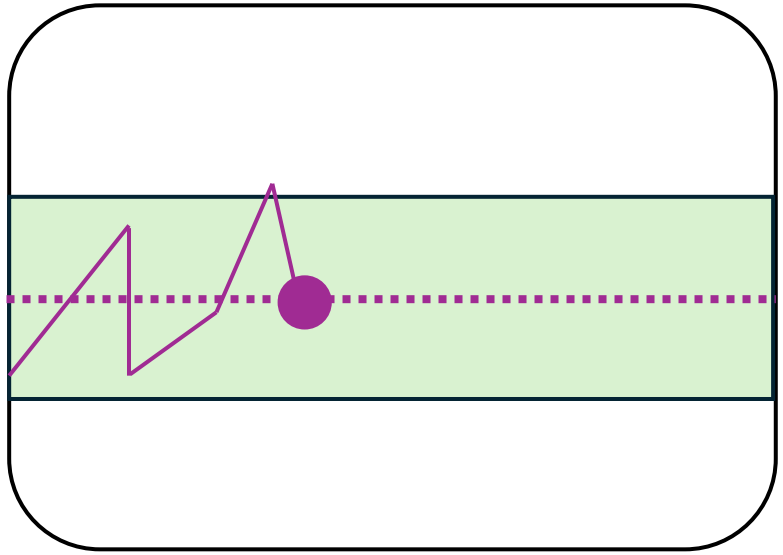
BC-PQP: Burst Control



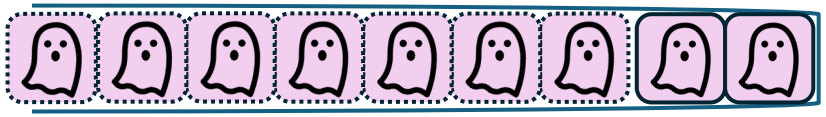
BC-PQP: Burst Control



BC-PQP: Burst Control

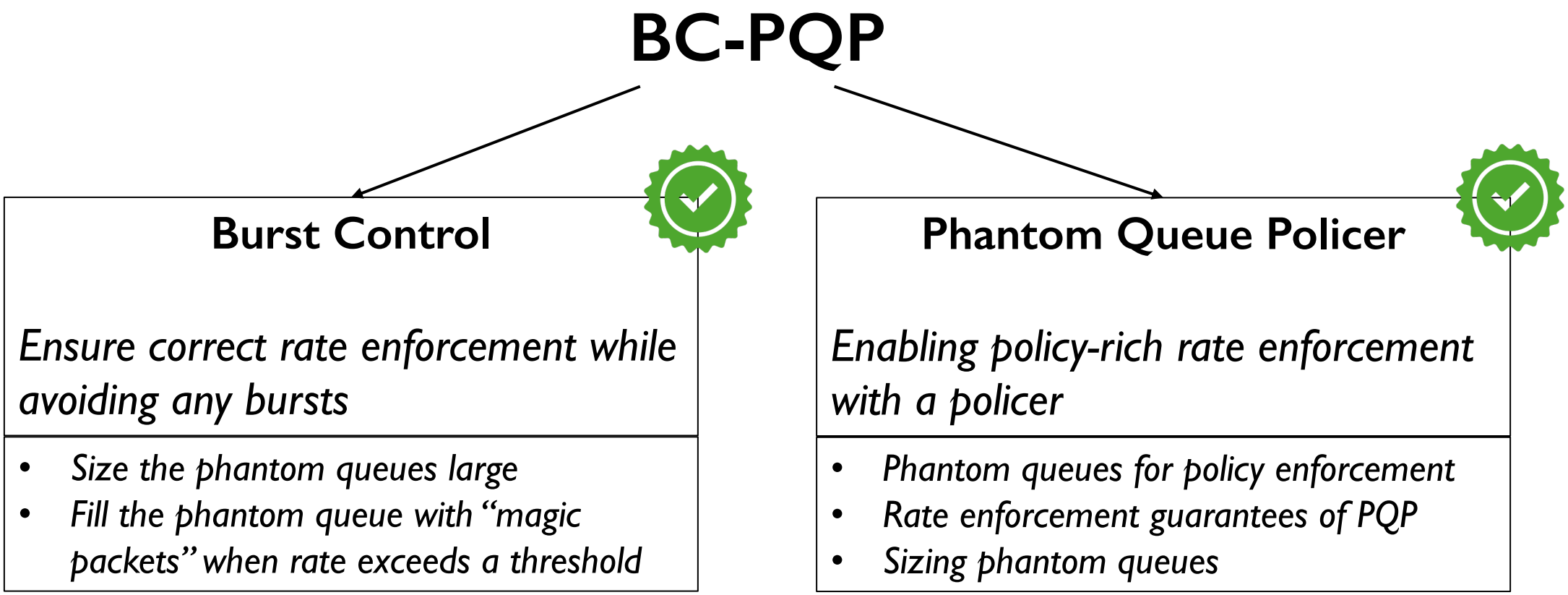


Incoming Rate

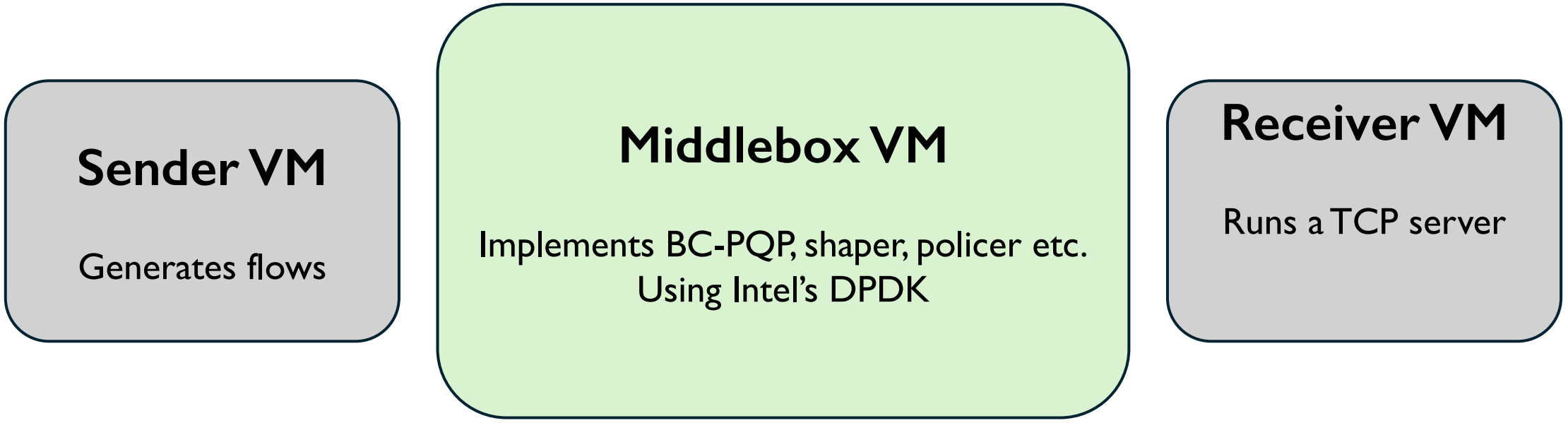


Phantom Queue
Occupancy

Our System


BC-PQP
Burst Control
Ensure correct rate enforcement while avoiding any bursts
• Size the phantom queues large
• Fill the phantom queue with "magic packets" when rate exceeds a threshold
Phantom Queue Policer
Enabling policy-rich rate enforcement with a policer
• Phantom queues for policy enforcement
• Rate enforcement guarantees of PQP
• Sizing phantom queues

Implementation



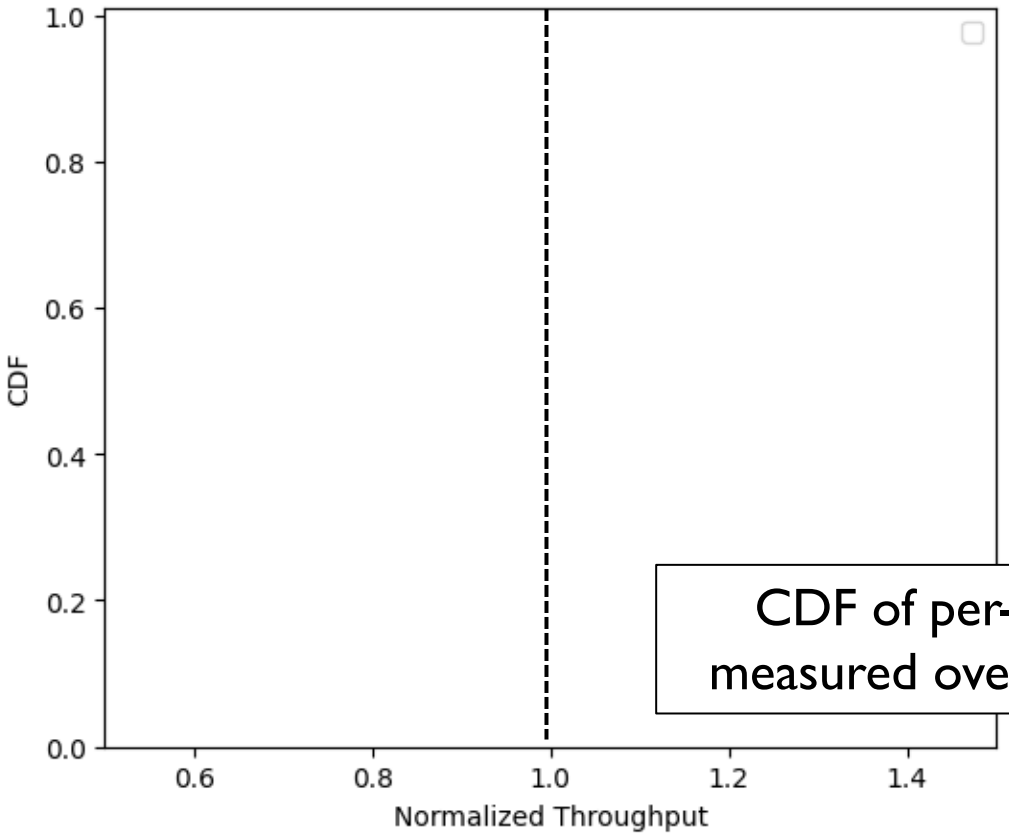
Measure CPU cycles spent per packets

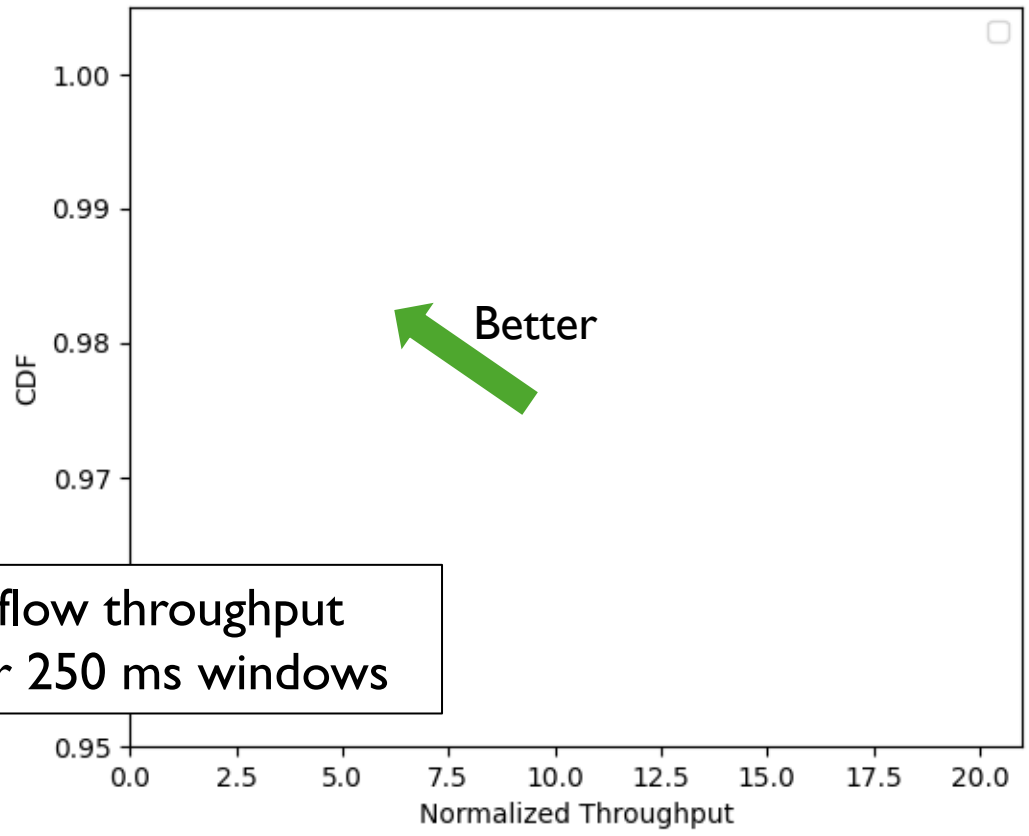
Measure per-flow throughput

CCs: *Reno, Cubic, BBR, Vegas*
RTTs: *2 – 50 ms*
Flow sizes: *10 KB – 100 MB*

Rates: *1.5 – 200 Mbps*
Policy: *Fairness*

Evaluation: Rate Enforcement





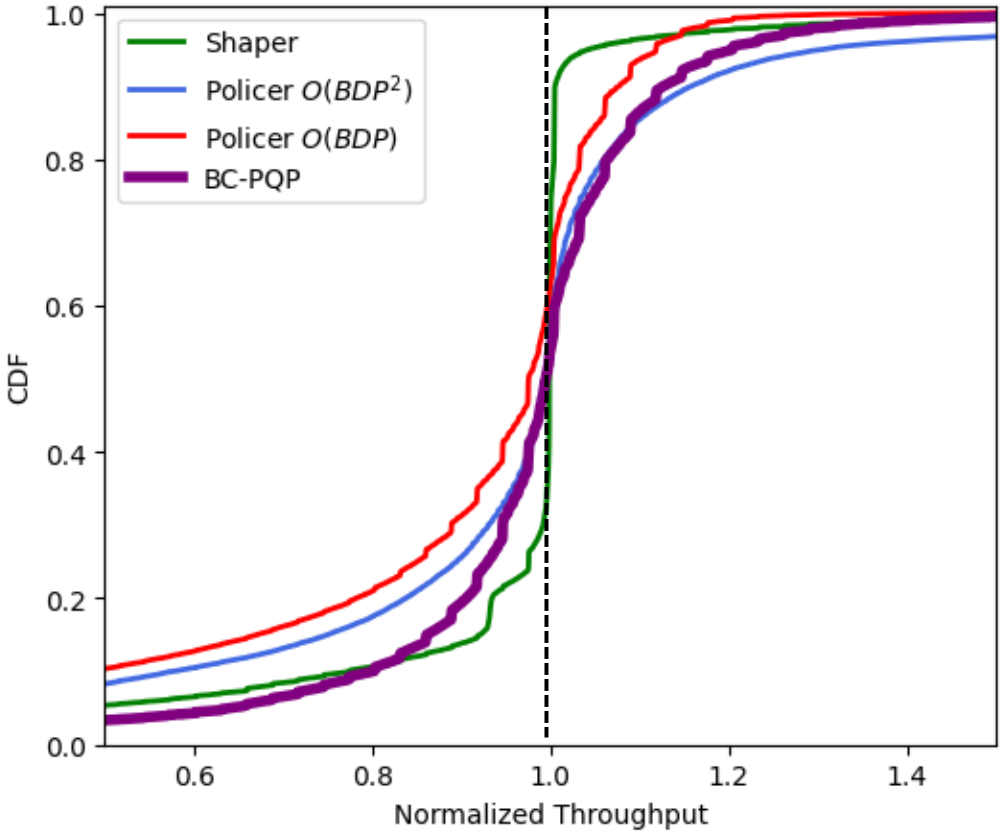
Better

CDF of per-flow throughput measured over 250 ms windows

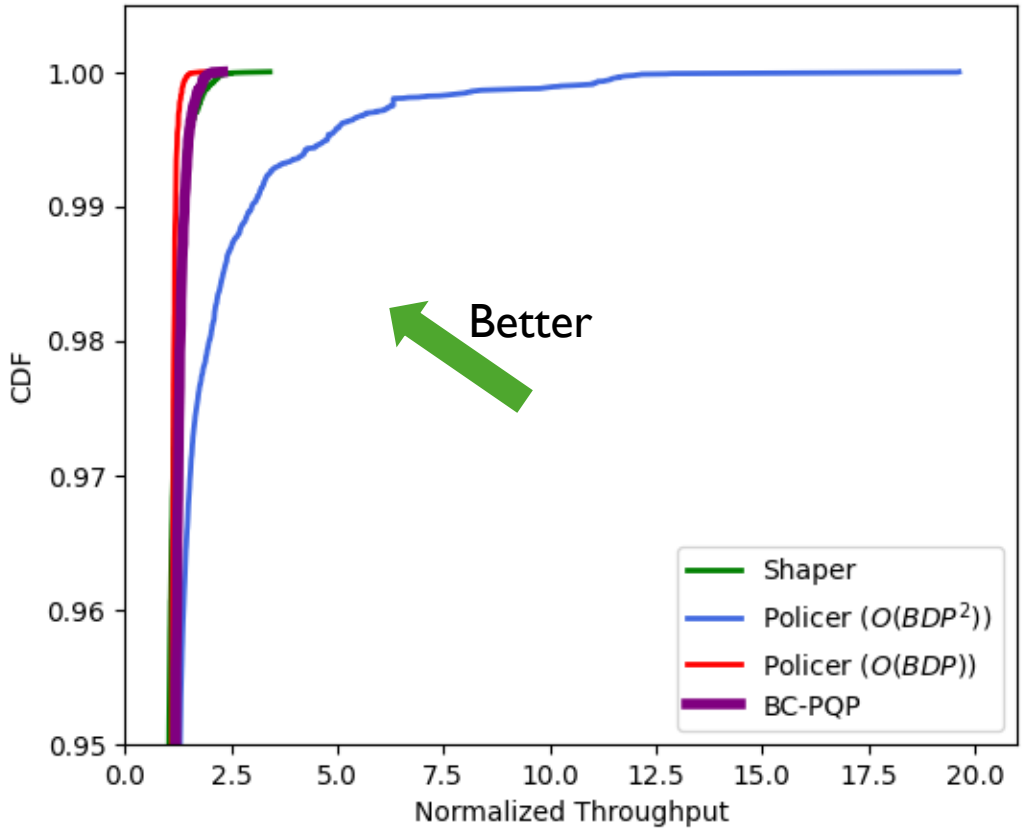
Average Rate Enforcement

Burst

Evaluation: Rate Enforcement

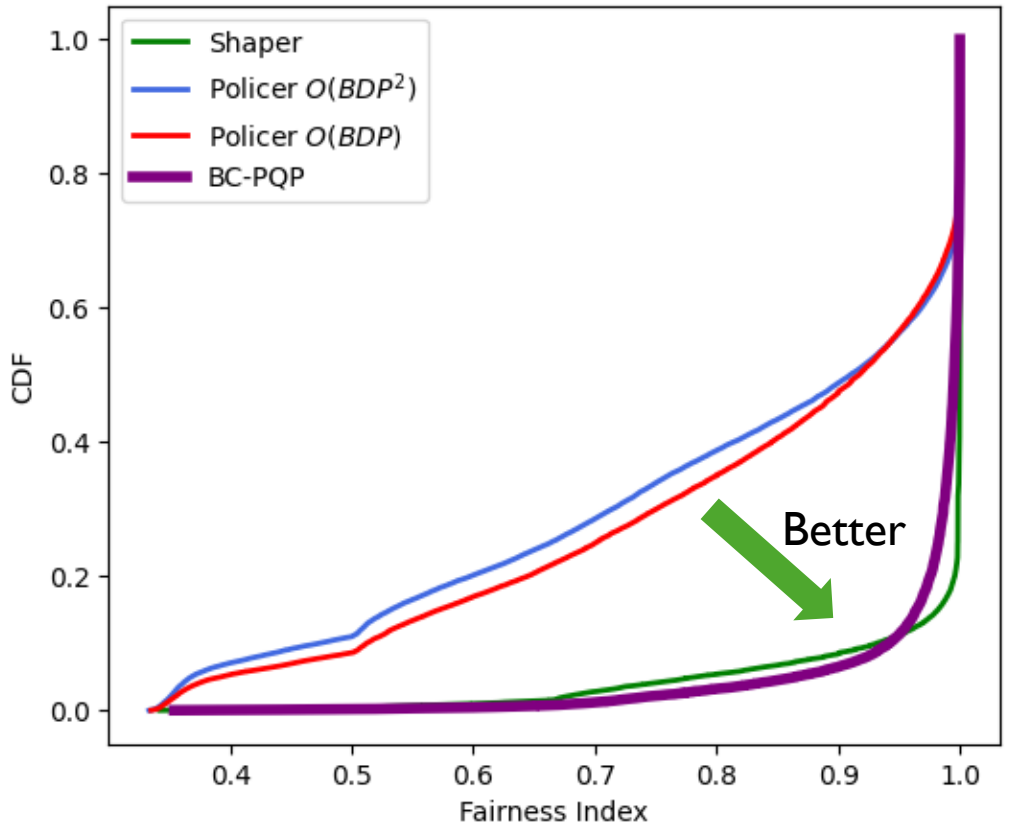


Average Rate Enforcement

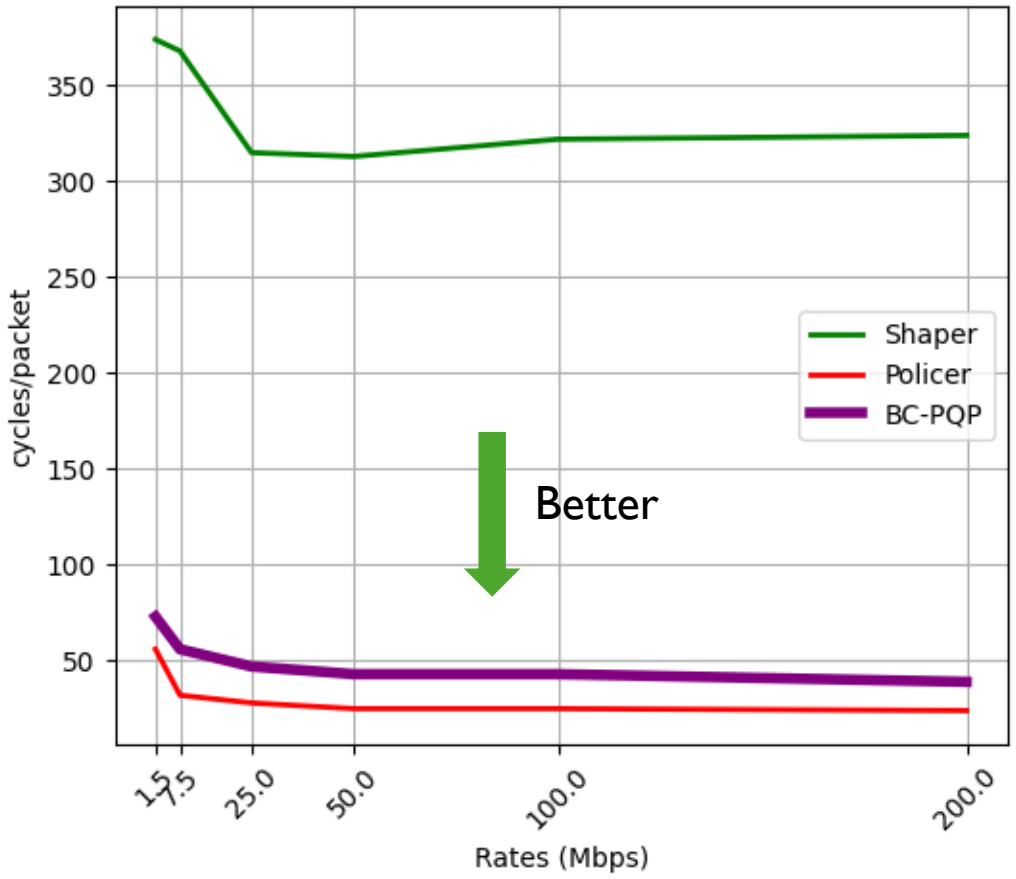


Burst

Evaluation: Policy Enforcement



Evaluation: Computing Efficiency



Evaluation: Other Results

- Support for different policies:
  - weighted fairness, prioritization and hierarchical policies
- Comparison with FairPolicer
- Effect on applications:
  - YouTube, Netflix, Web browsing
- Other metrics:
  - drop-rates

Conclusion

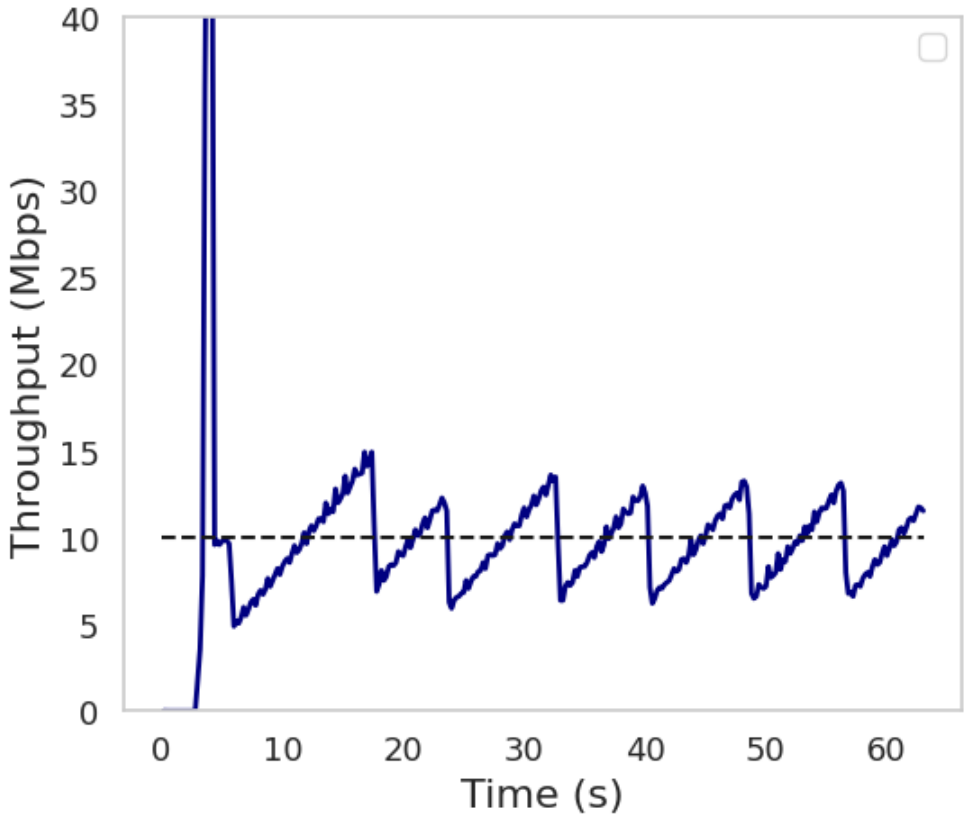
- With BC-PQP, rate enforcement can be *correct*, *policy-rich* and *efficient* simultaneously!
  - ***Low-cost isolation***, at least on artificial bottlenecks?
  - ***ISPs*** can support user-driven rate-sharing policies with minimal additional cost.
  - ***Cellular service providers, Datacenters and WANs*** can do burst-free rate enforcement to manage limited resources at lower cost with fine-grained rate-sharing polices

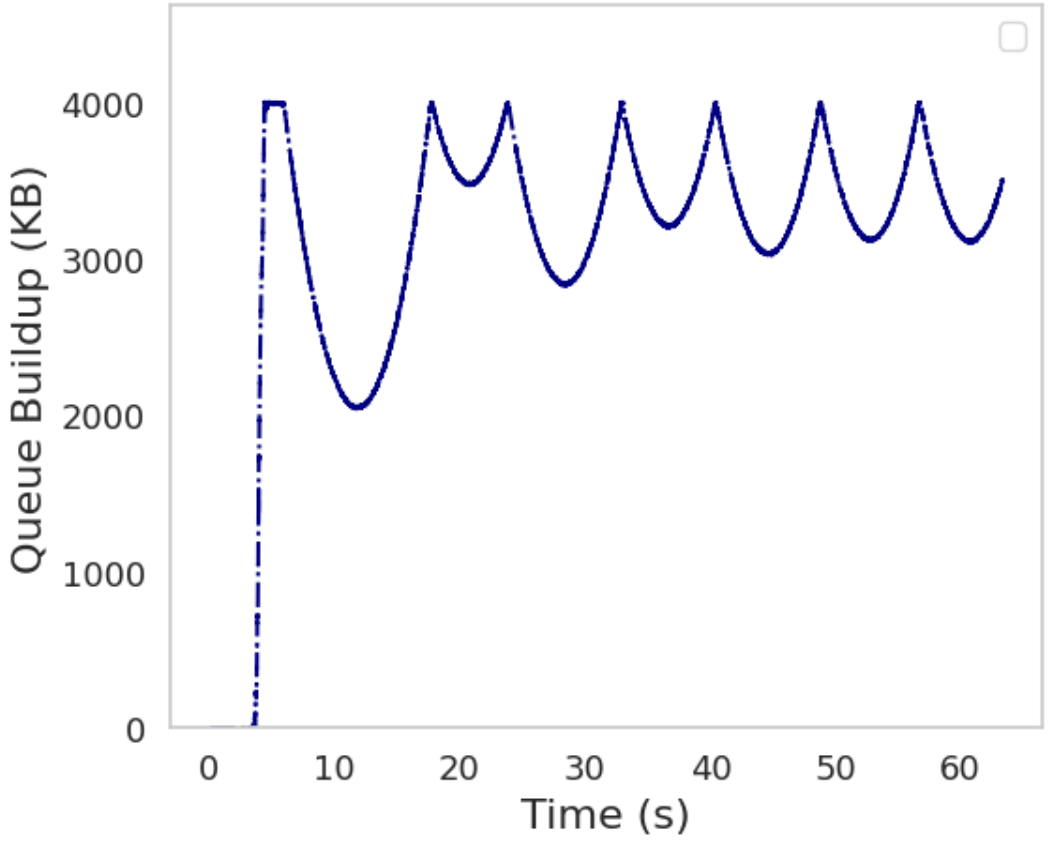
Please reach out if you're interested in using BC-PQP: *ammart2@Illinois.edu*

Thank you for listening!

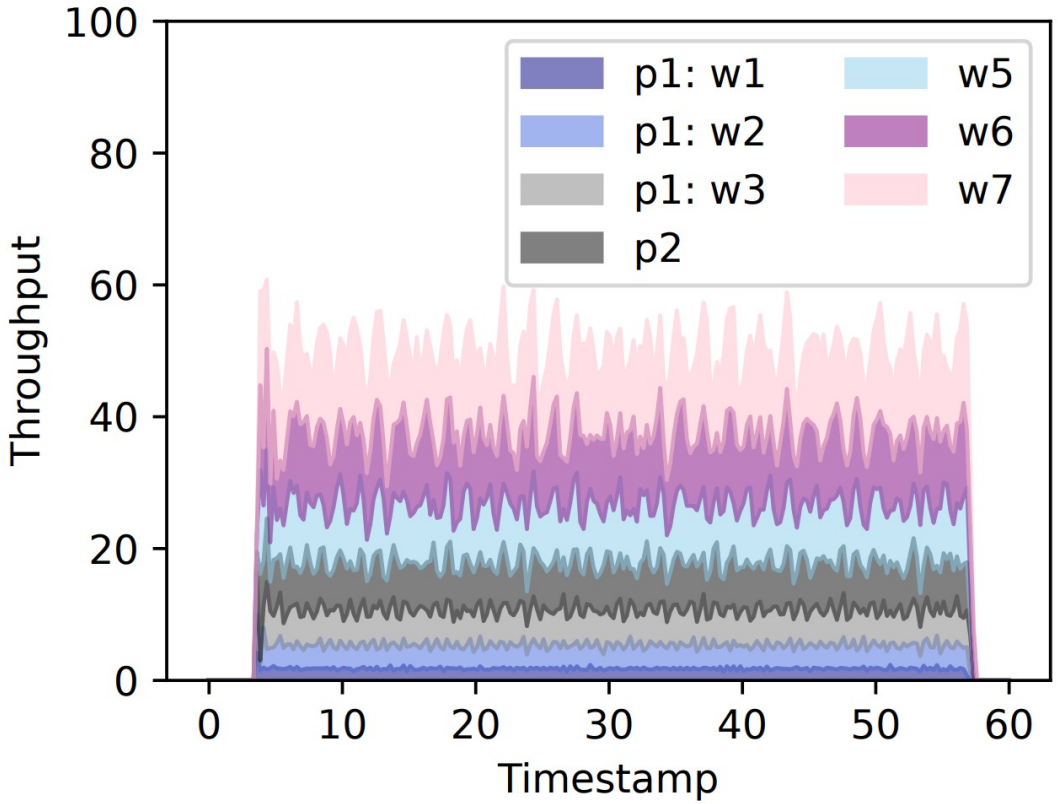
Back-up Slides Ahead!

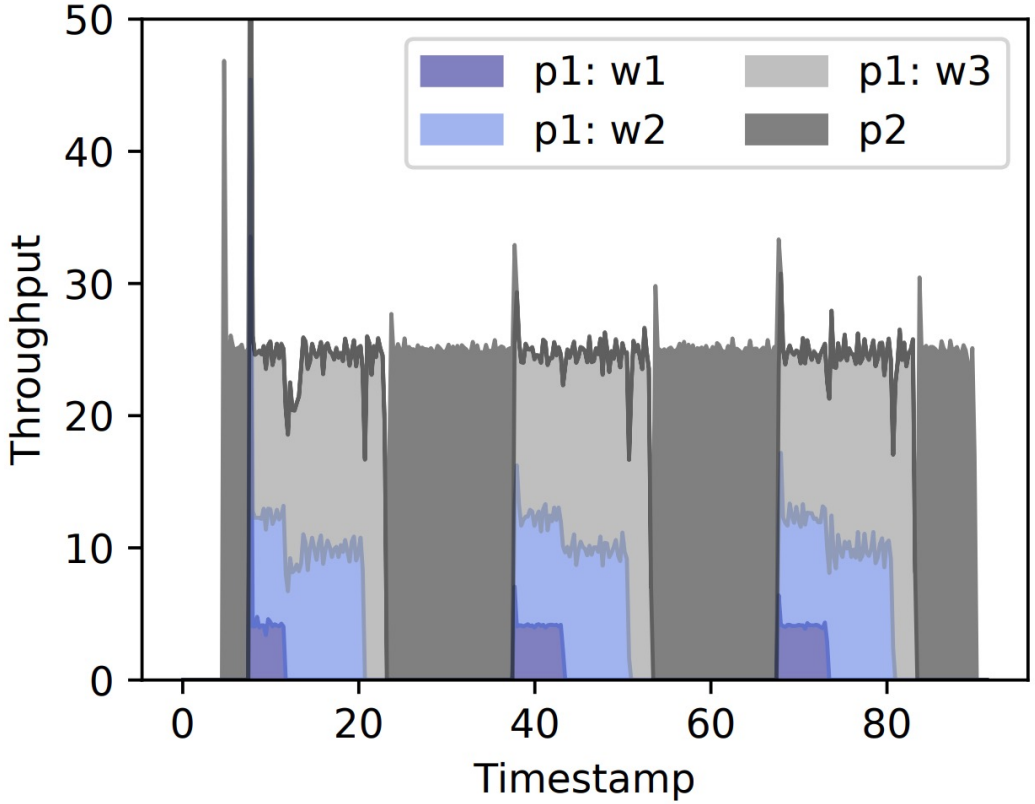
Upper limit on size is not important



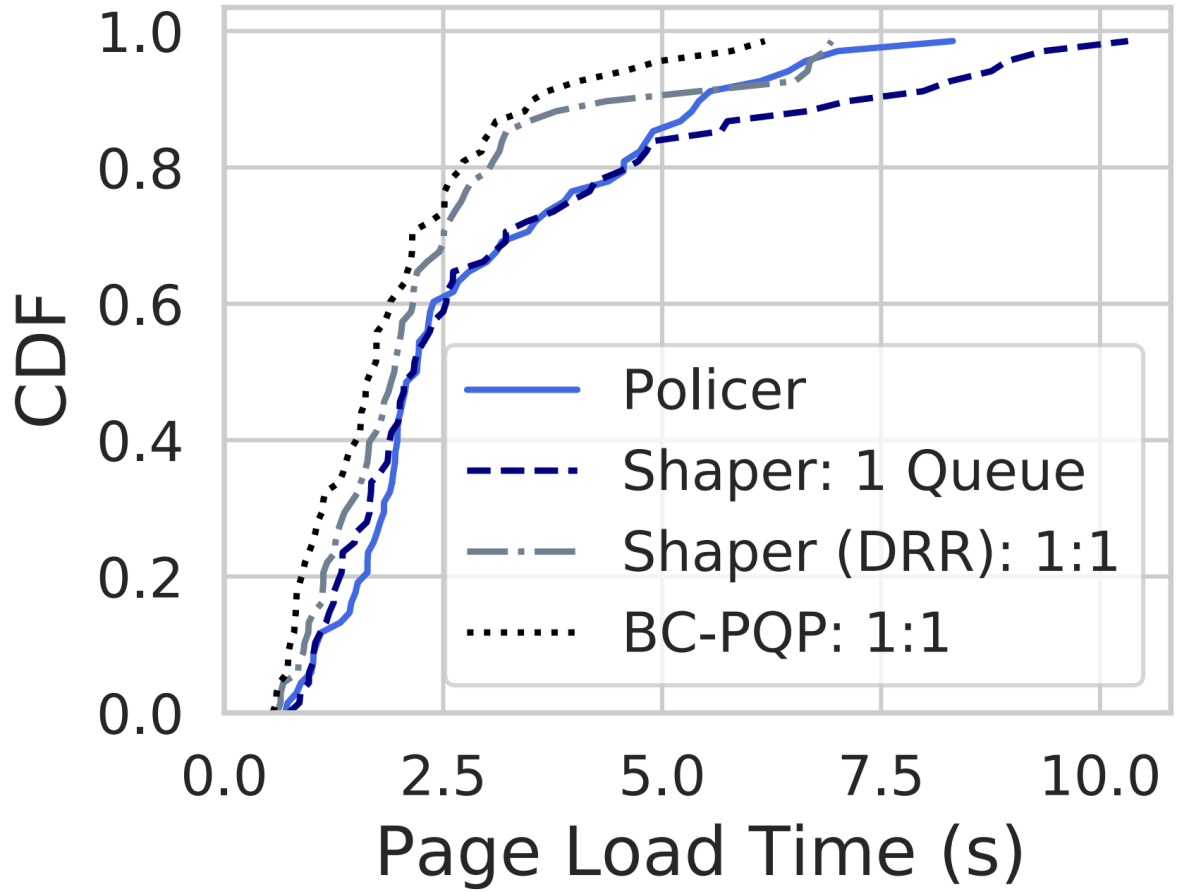


Policy Enforcement with BC-PQP

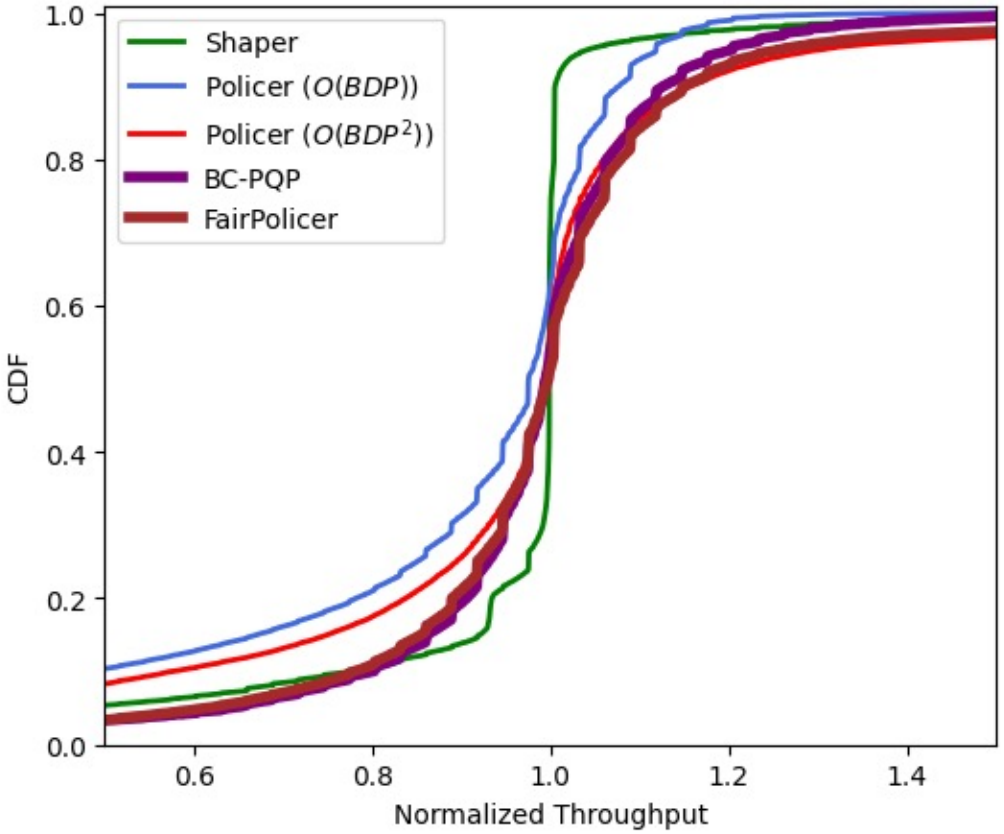




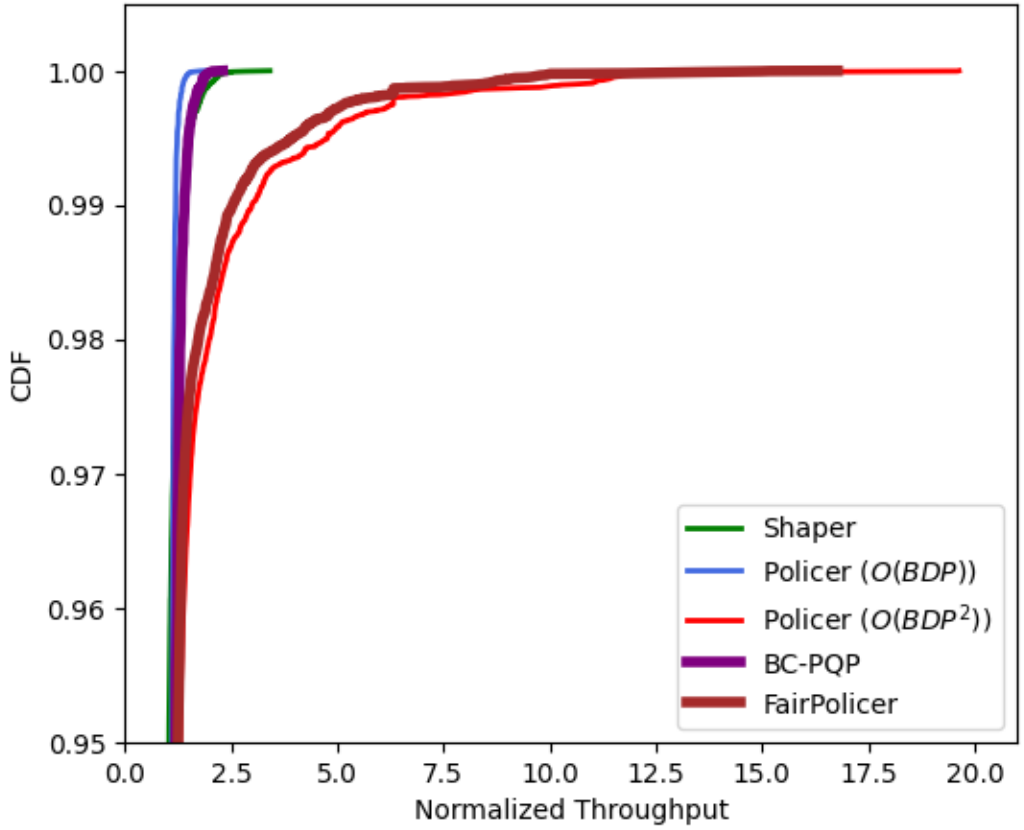
Web browsing with BC-PQP



BC-PQP vs FairPolicer: Rate Enforcement

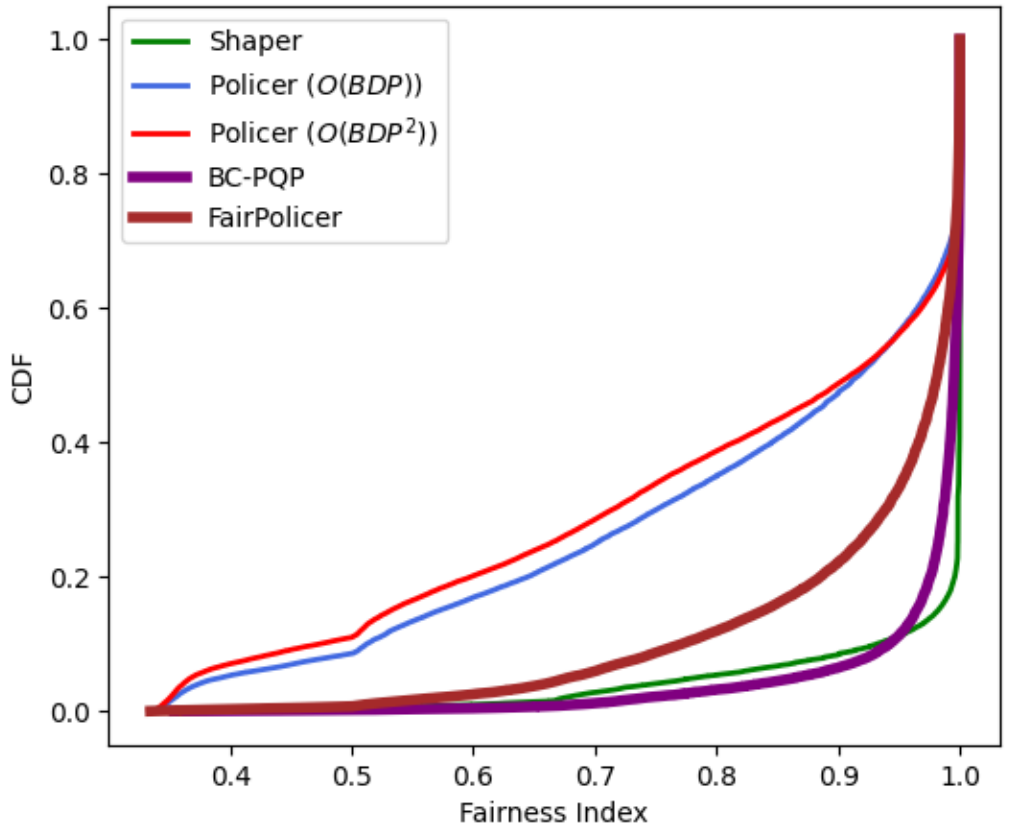


Average Rate Enforcement

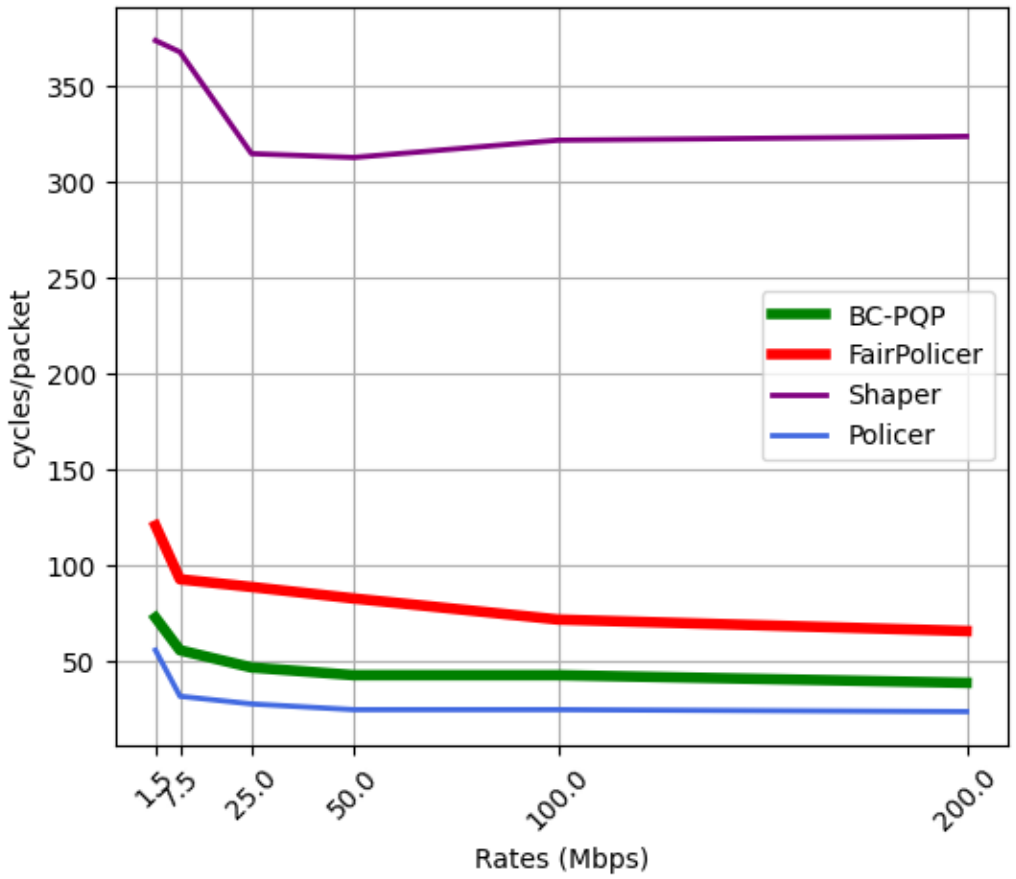


Burst

BC-PQP vs FairPolicer: Fairness



BC-PQP vs FairPolicer: Fairness



BC-PQP vs FairPolicer: CPU Cycles