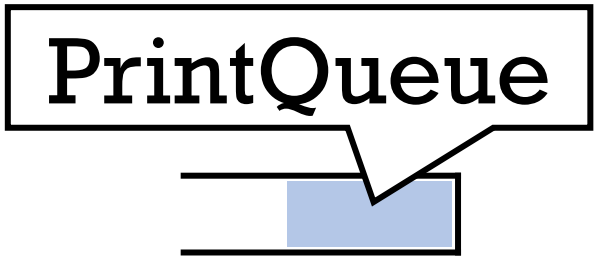
SIGCOMM 2022
Amsterdam

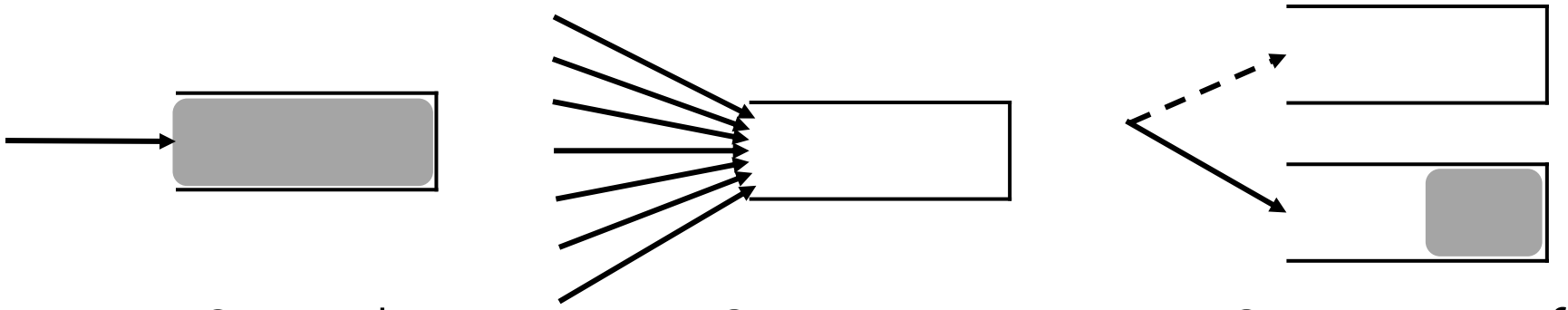


Performance Diagnosis via Queue Measurement in the Data Plane

Yiran Lei, Liangcheng Yu, Vincent Liu, Mingwei Xu

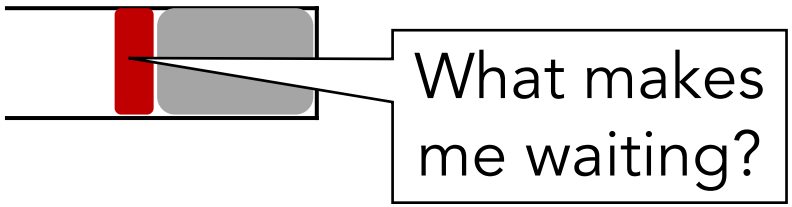
Why performance issue?

Performance issue debugging: hard



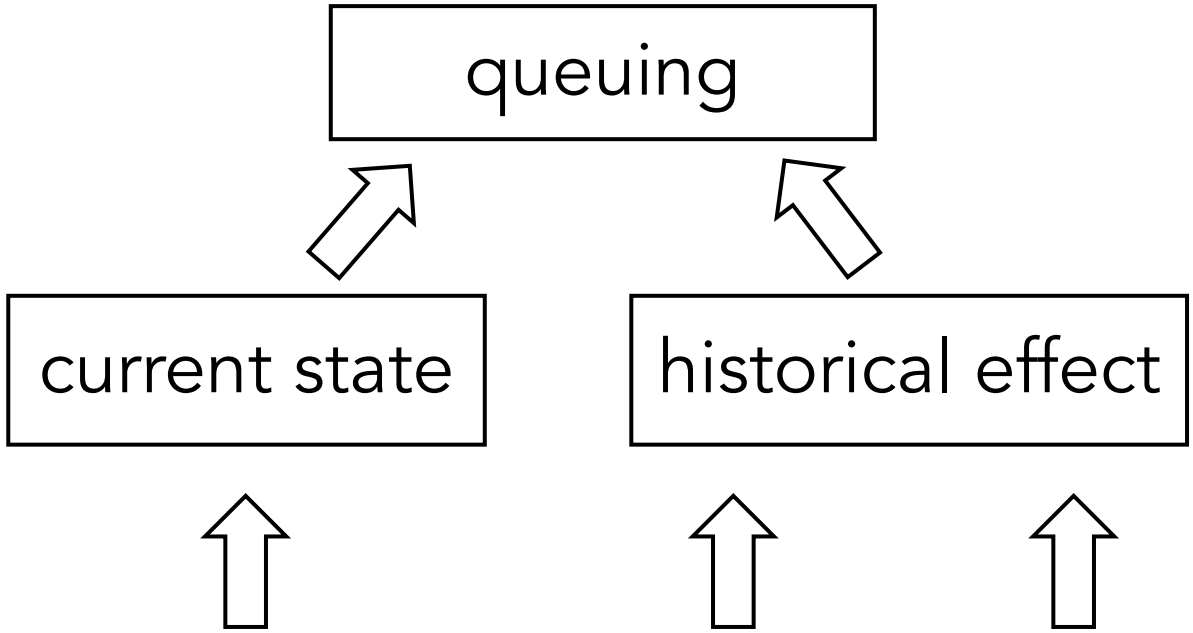
Packets reaching destinations: late

Packet-level causes of queuing delay: critical



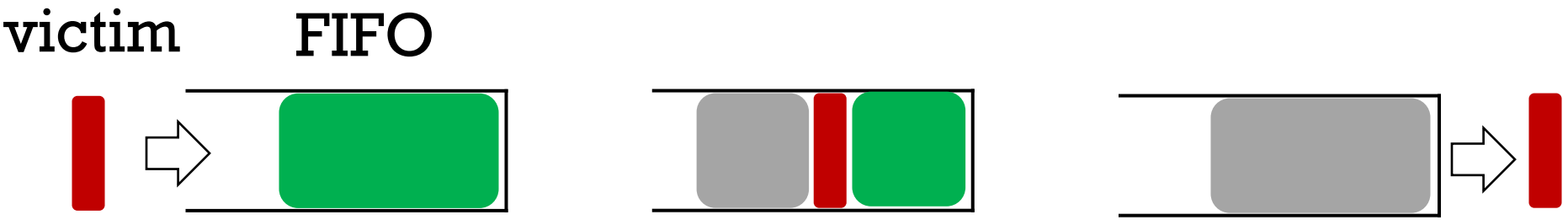
Provenance of queuing delay

Observation:



A congestion regime: direct indirect original

Direct culprits

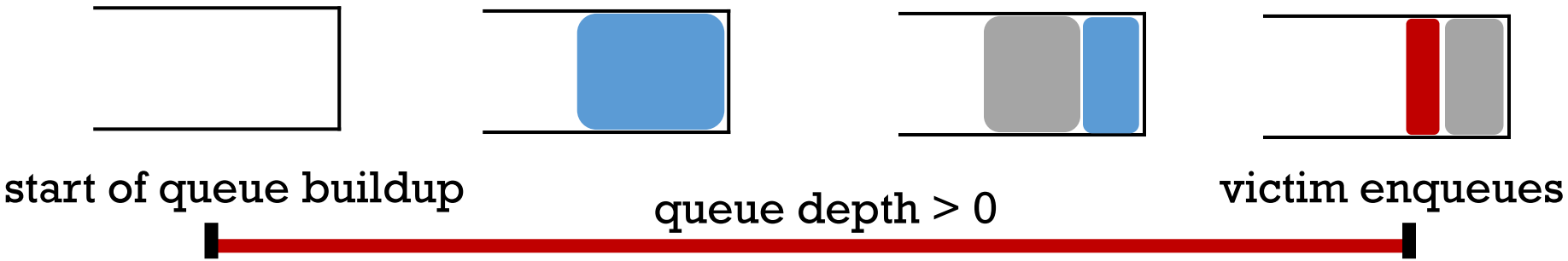


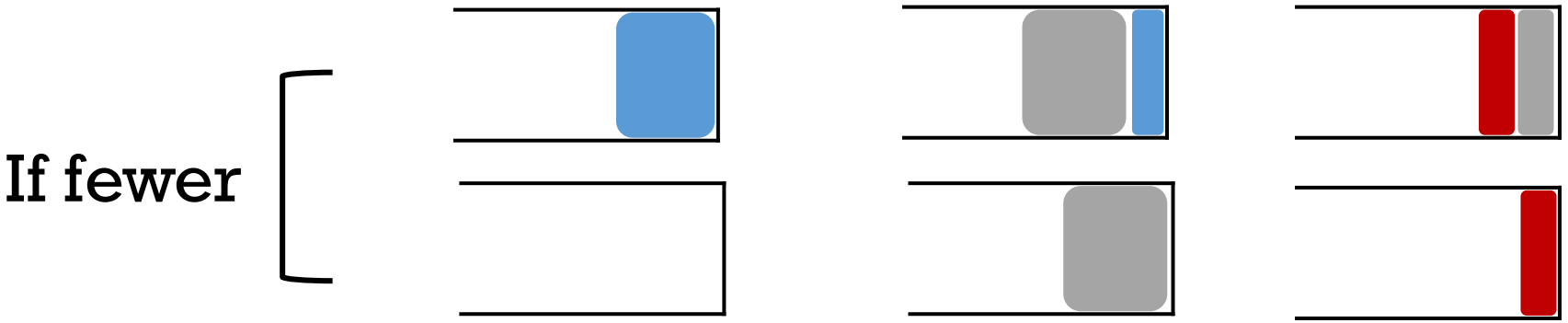
enqueue queuing dequeue

Why victim waits?

Indirect culprits

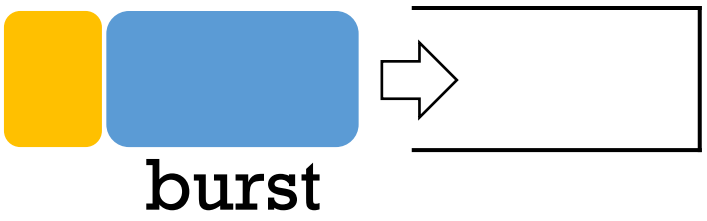
Why direct culprits are delayed?

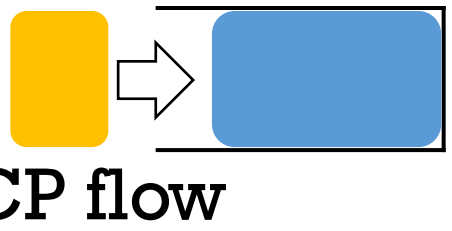


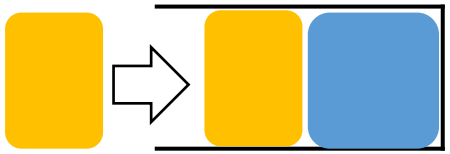


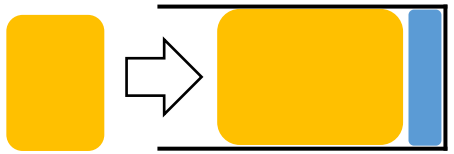
Original culprits

A subset of indirect culprits: more blame

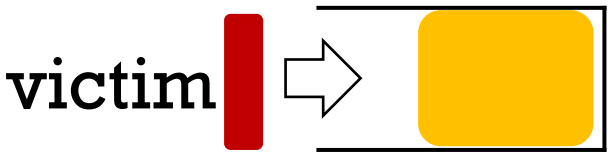








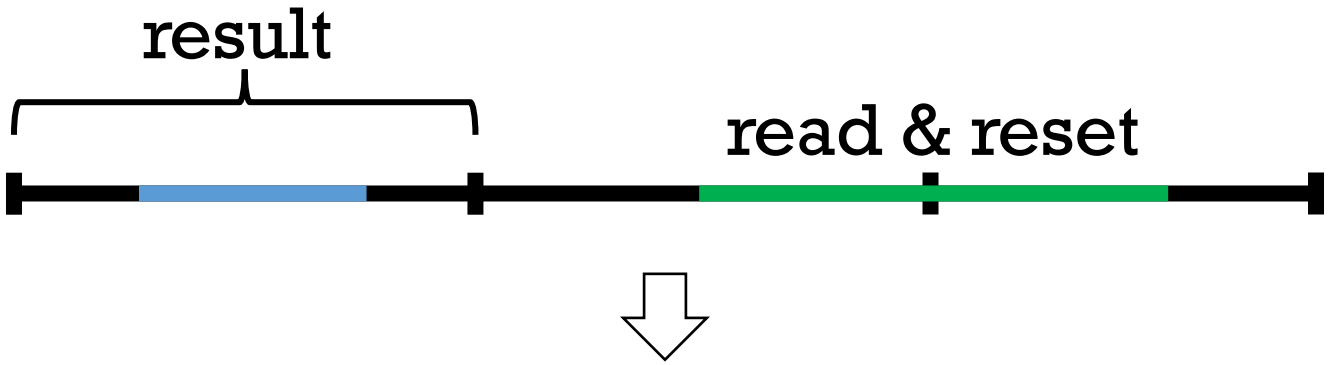
. . .



Why queue depth > 0 ?

Gap of existing works

- Heavy hitter detection: fixed time intervals



arbitrary timespans: over/under-estimate

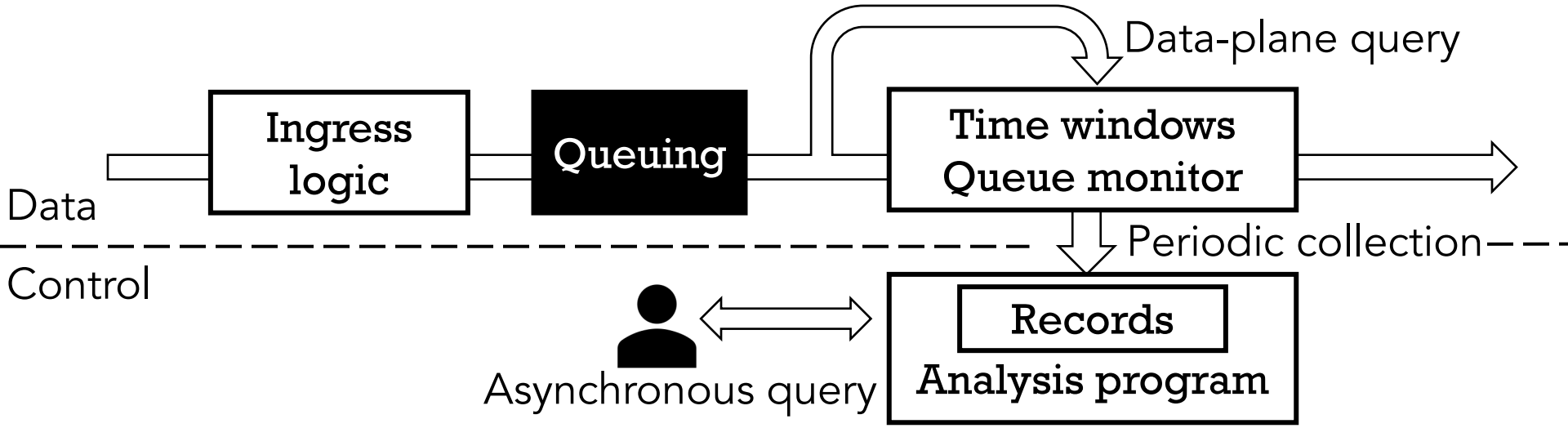
- Packet mirroring: large overhead

Overview

Passive monitoring framework, new data structures:

- time windows – direct, indirect culprits
- queue monitor – original culprits

Hardware prototype: 3x accuracy↑, 20x overhead↓



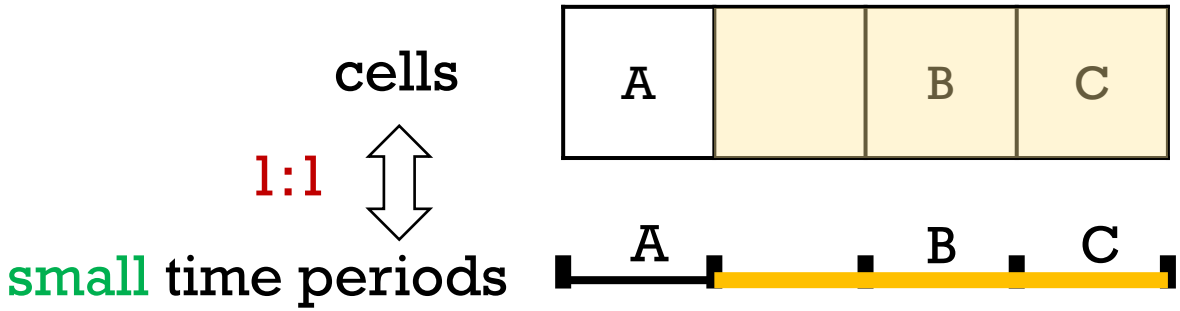
→ Time windows

Queue monitor

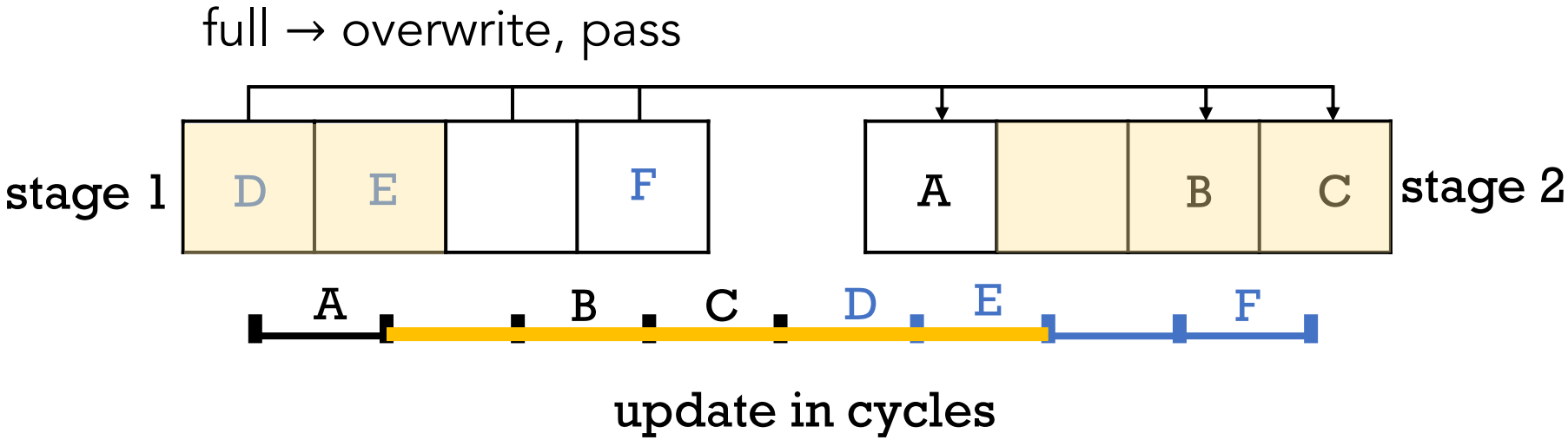
Query execution

Time windows

- Strawman

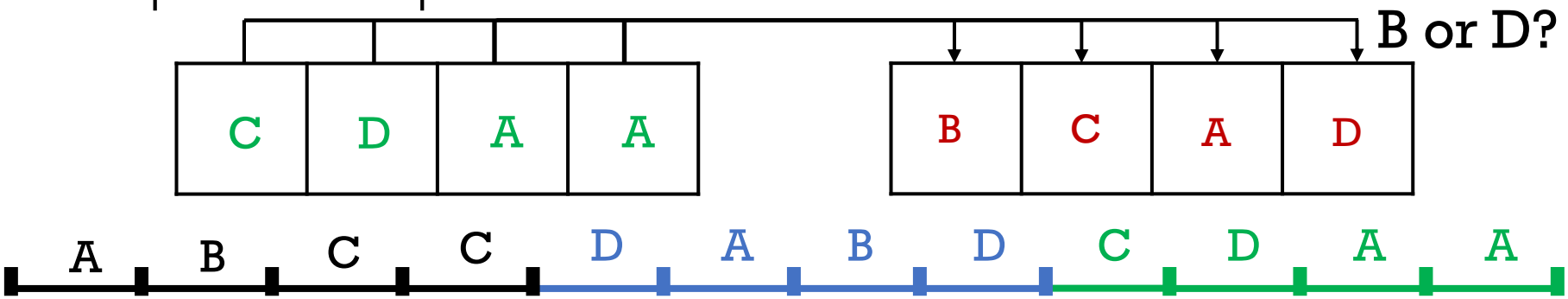


- Hierarchical variant: fit into data-plane stages



Time windows

Save space by slightly sacrificing accuracy:
compress multiple cells into one



How to compress:
latest record ∈ cells that stored packets in the previous cycle

Proportional property:
compressed packet number : original number = constant

Example:

A	B	C	D
2	0	1	1

+

A	B	C	D
1	1	1	1

× 2 =

A	B	C	D
4	2	3	3

Time windows

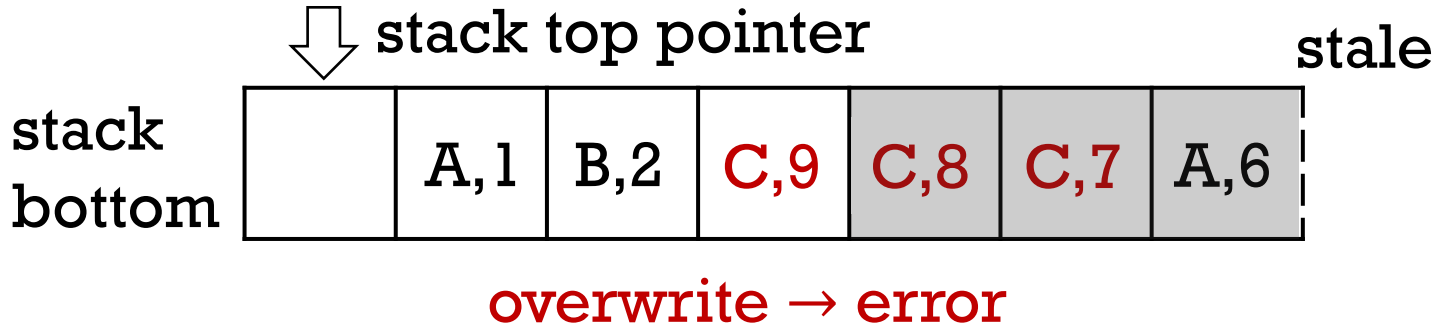
→ Queue monitor

Query execution

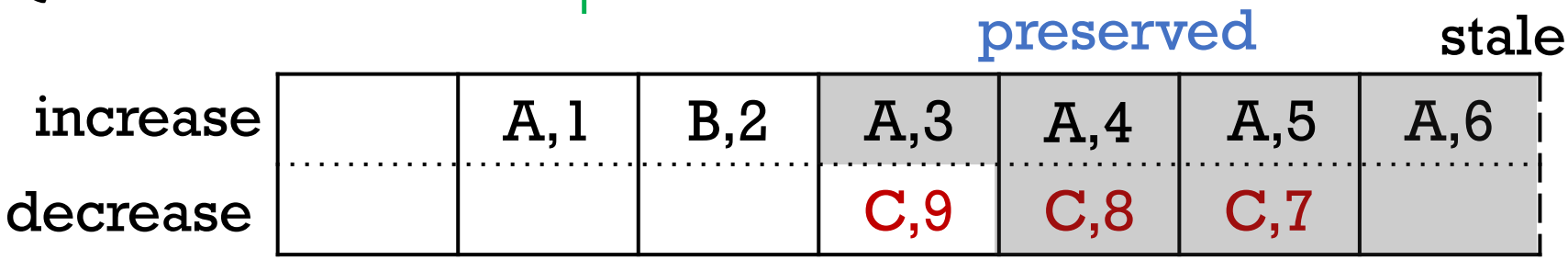
Queue monitor

Stack

Strawman: sequence number



Queue monitor: separate in/de-crease



Time windows

Queue monitor

⟶ Query execution

Query execution

Asynchronous query

- triggered by users
- concurrently read, write with *Mantis;* periodic collect, store

Data-plane query

- triggered by packets
- freeze, switch registers
- higher accuracy: frozen registers → initial time windows → less compression

Evaluation

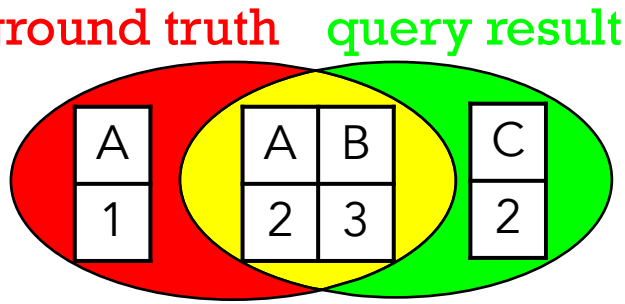
github.com/A-Dying-Pig/PrintQueue ~5000 loc

Testbed:

- Star topology: 1 Tofino switch, 4 servers
- tcpreplay, netmap, DPDK

Workloads:

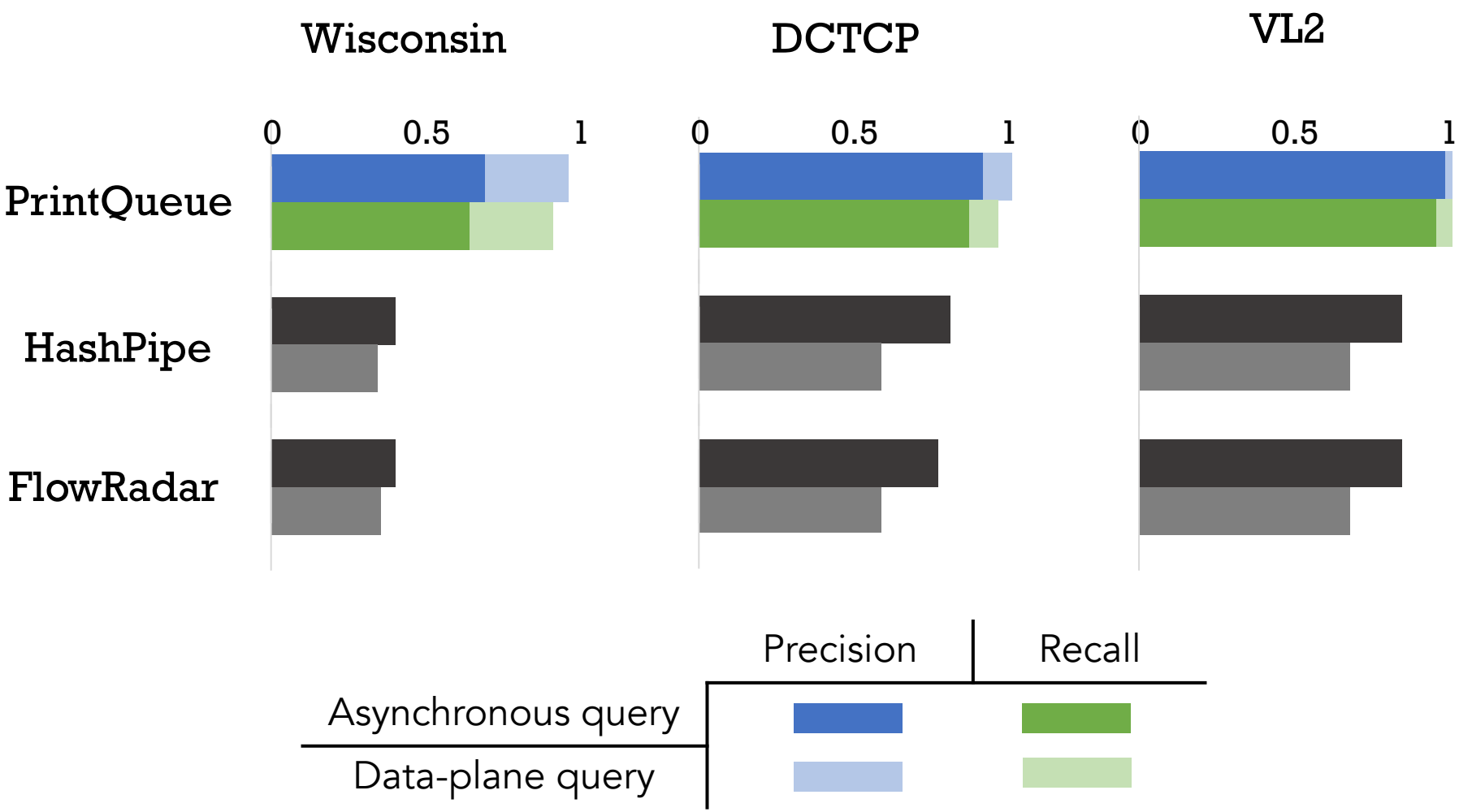
- Wisconsin trace
- DCTCP, VL2 synthetic trace



Methodology:

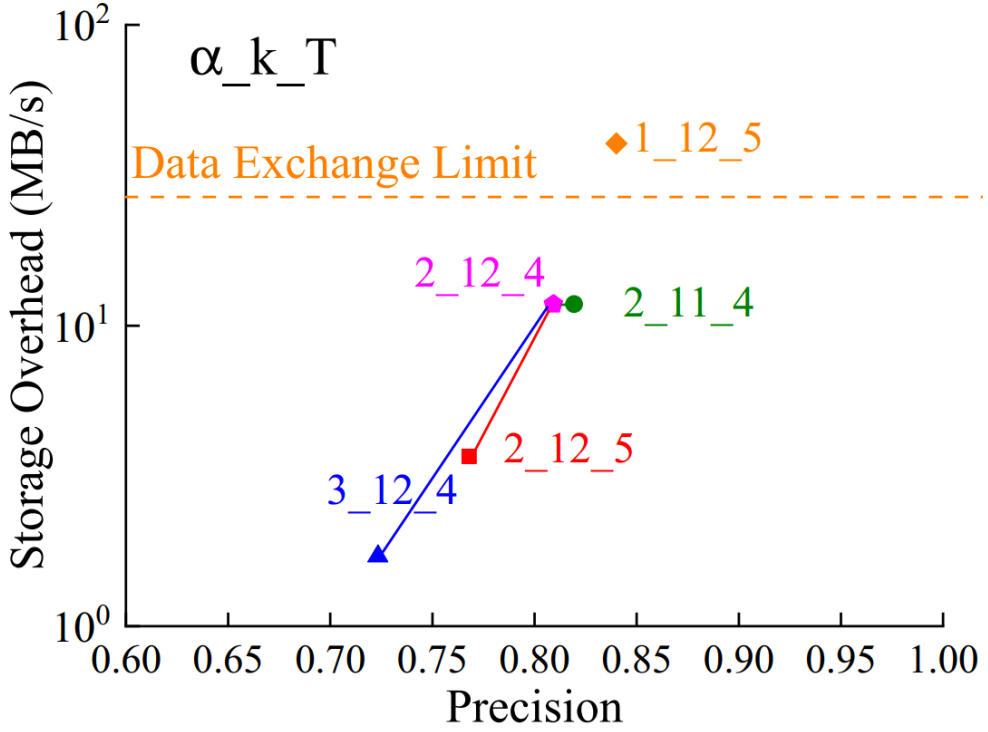
- Ground truth: packet carry queuing period → receivers
- precision = $\frac{5}{7}$, recall = $\frac{5}{6}$

PrintQueue achieves high accuracy



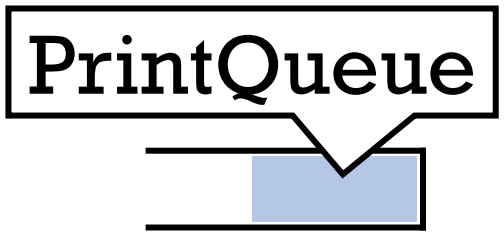
PrintQueue incurs low overhead

- parameters - α, k, T
- bottleneck



Simultaneous activation in 10 ports.

Summary



- **Practical data-plane monitoring system:** the provenance of queuing delay
- **A congestion regime:** direct, indirect, original culprits
- **Two data structures:** time windows, queue monitor
- **Hardware prototype:** 3x accuracy↑, 20x overhead↓

github.com/A-Dying-Pig/PrintQueue

yiranlei.yiranlei@gmail.com