Justitia

Software Multi-Tenancy in Hardware Kernel-Bypass Networks

Yiwen Zhang[1], *Yue Tan*[2], *Brent Stephens*[3], *Mosharaf Chowdhury*[1]

University of Michigan[1], Princeton University[2],

University of Illinois at Chicago[3]

SymbioticLab

Why use Kernel-Bypass Networks?

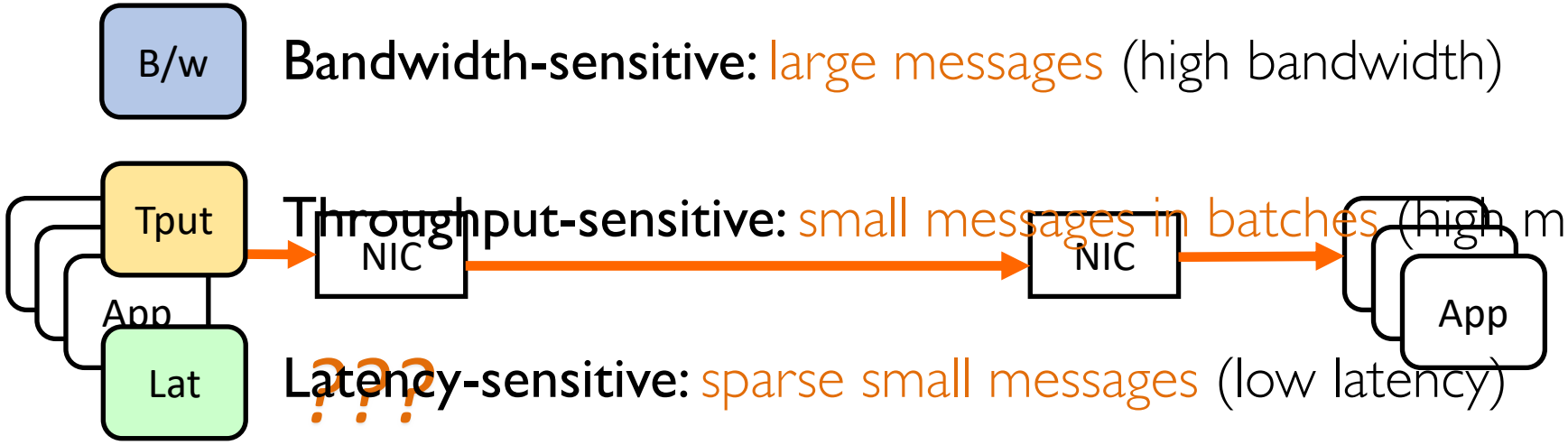
KBN provides low latency & high throughput **by removing kernel from the data path**.

Hardware-based KBN (e.g., RDMA): offloads data transfer to specialized NIC

- Lower latency & better CPU efficiency
- Widely used in cloud environments

What about Multi-Tenancy Support?

What if there are multiple applications sharing the same NIC?



Bandwidth-sensitive: large messages (high bandwidth)

Throughput-sensitive: small messages in batches (high message rate)

Latency-sensitive: sparse small messages (low latency)

In hardware KBN, the OS has no control but to **rely on the** NIC to isolate among multiple applications.

Multiple on-NIC Resources

2 on-NIC resources:

- Link Bandwidth → B/w
- Execution Unit Throughput → Tput

➢ **Resource contention**

Lat does not saturate either resources, but can be blocked by large messages due to **head-of-line (HOL) blocking**

How Well Does Hardware KBN Provides Performance Isolation?

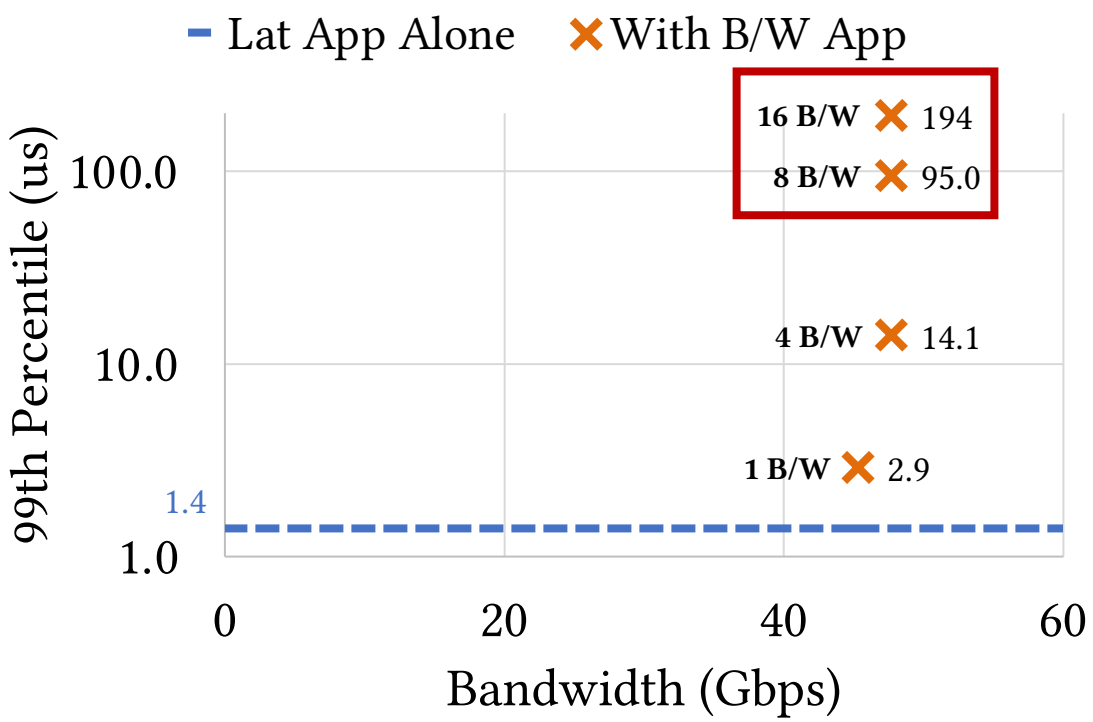
Does RDMA Provide Isolation?

Provides Isolation?[1]	Latency	Throughput	Bandwidth
Bandwidth	NO	NO	NO
Throughput	YES	YES	
Latency	YES		

- Performance isolation is not guaranteed when bandwidth-sensitive apps are present.

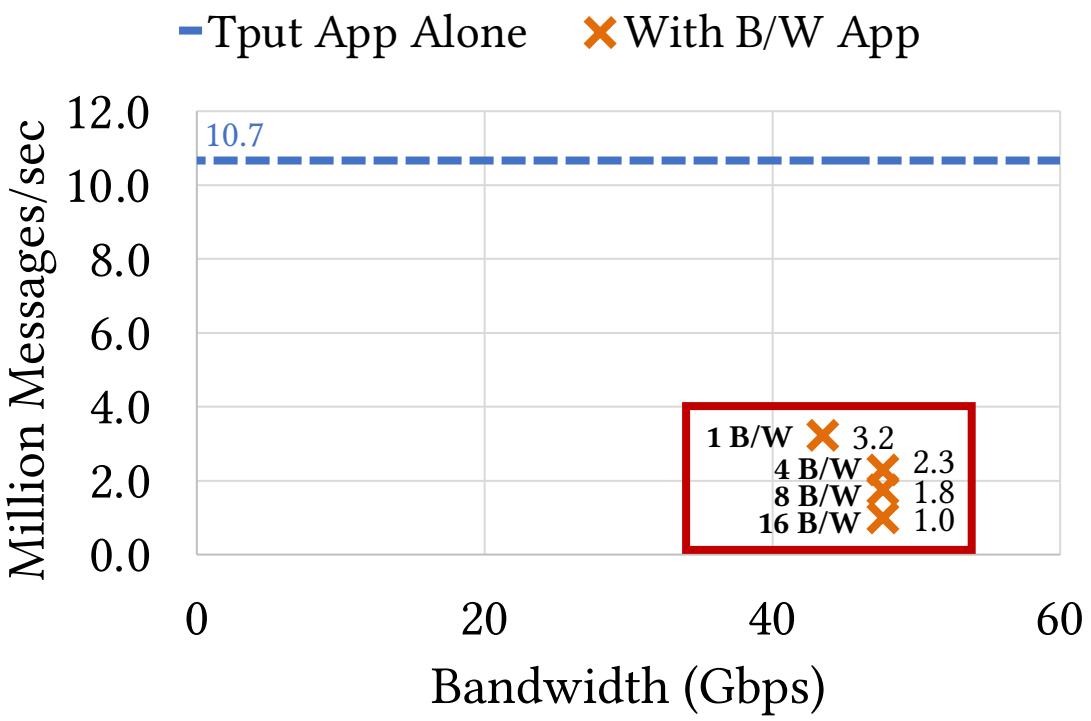
[1]: For each experiment we choose 2 types of applications to compete under InfiniBand/RoCE/iWarp. Applications include Mellanox Perftest and other real RDMA applications.

Anomalies in Latency-Sensitive Apps



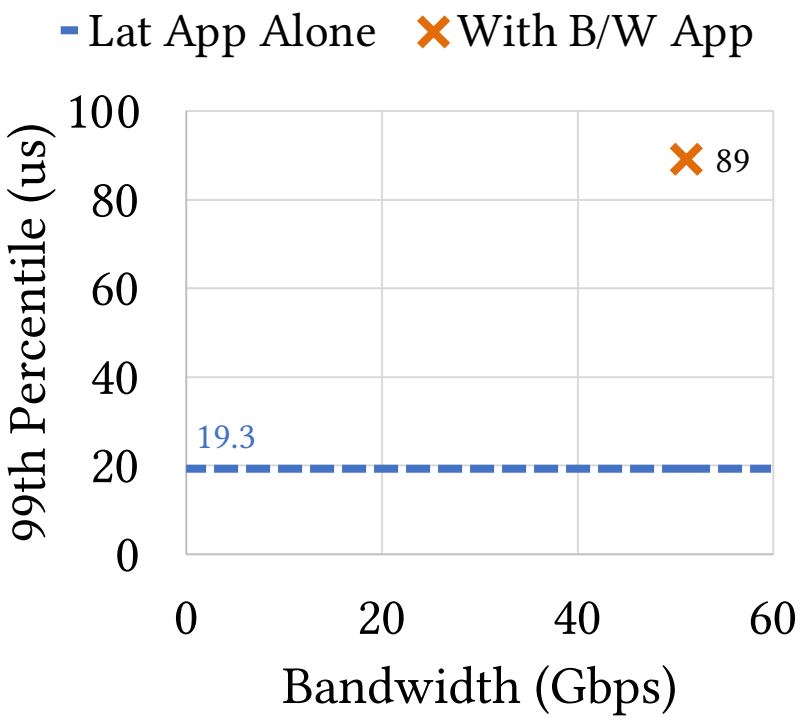
- **Latency-sensitive** apps need isolation from **bandwidth-sensitive** apps

Anomalies in Throughput-Sensitive Apps

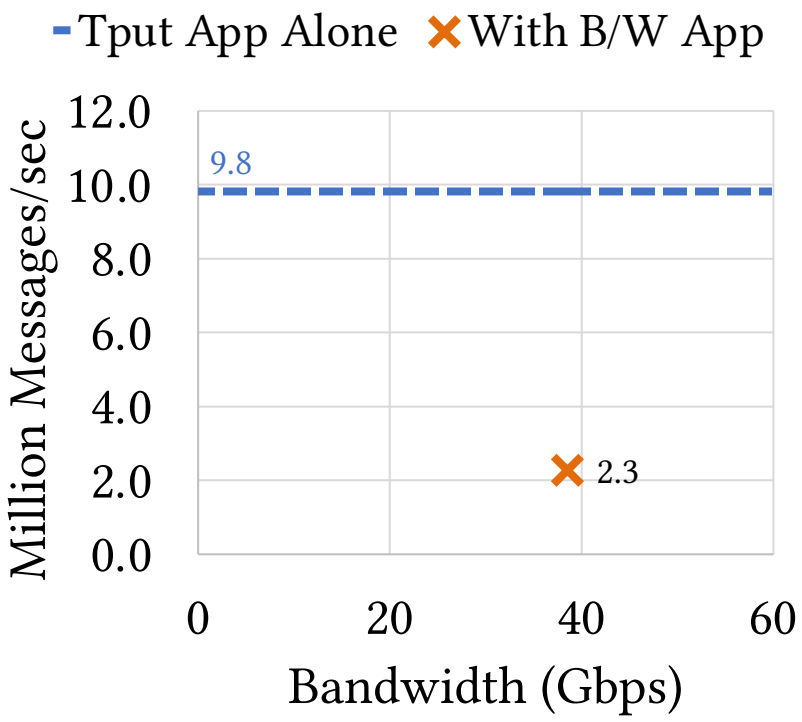


- **Throughput-sensitive** apps also need isolation from **bandwidth-sensitive** apps

What About Real Applications?



DARE (Latency-sensitive)



FaSST (Throughput-sensitive)

Justitia

Provides multi-tenancy support for hardware-based KBN

Provides multi-tenancy support in an

- efficient,
- scalable, and
- transparent manner

without modifying

- applications,
- operating systems, or
- hardware

Justitia Key Idea

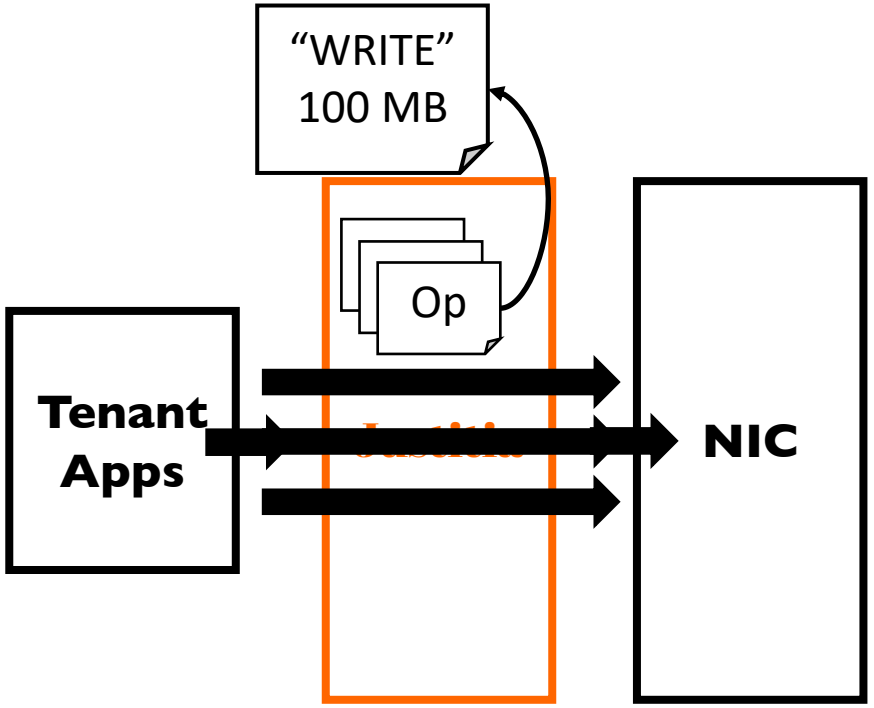
Tenant directly talks to NIC

- arbitrary large messages
- many connections

Justitia:

Add a software layer between tenants and NIC

- creates a point of control
- prevent tenants from hogging NIC resources



Justitia

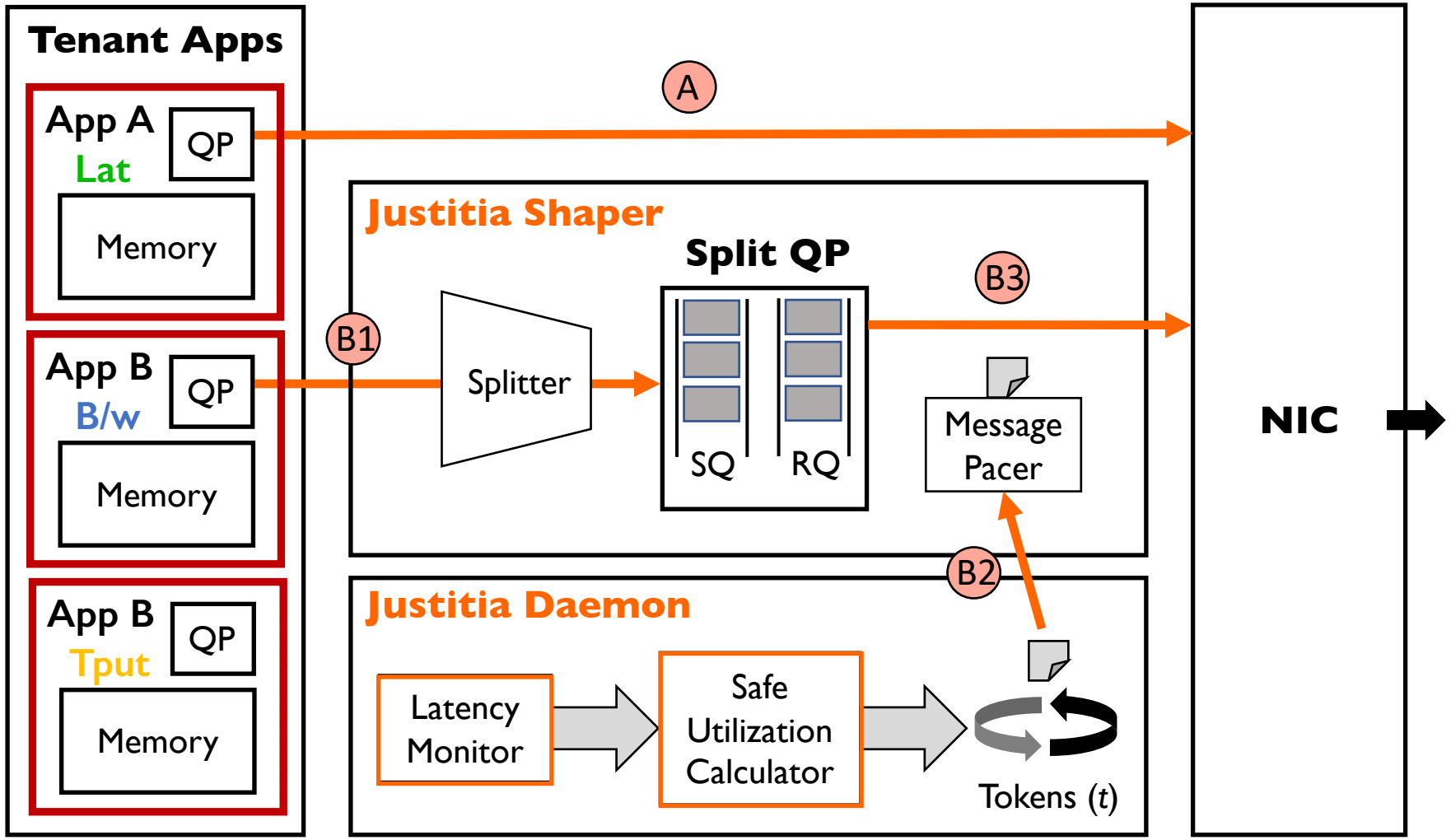
Provides multi-tenancy support for hardware-based KBN

1. **Justitia** Daemon
2. **Justitia** Shaper

Performs latency monitoring and proactive rate management

Enforces resource utilization provided by the Daemon

Justitia in One Slide

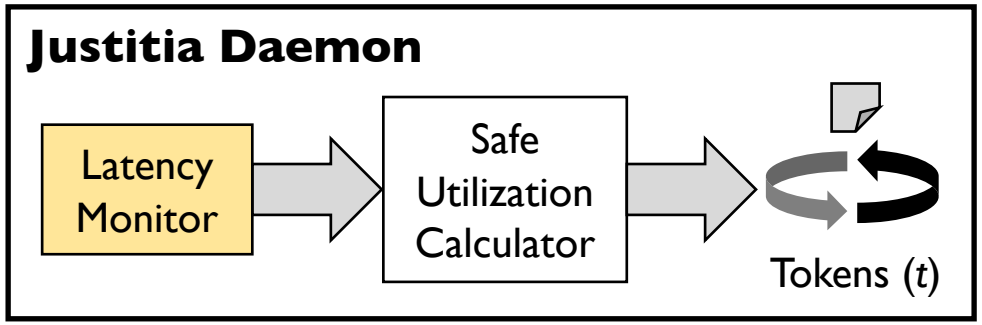


Justitia Daemon

Passive Latency Monitoring

- Probing latency-sensitive apps is expensive
- Maintains a reference flow

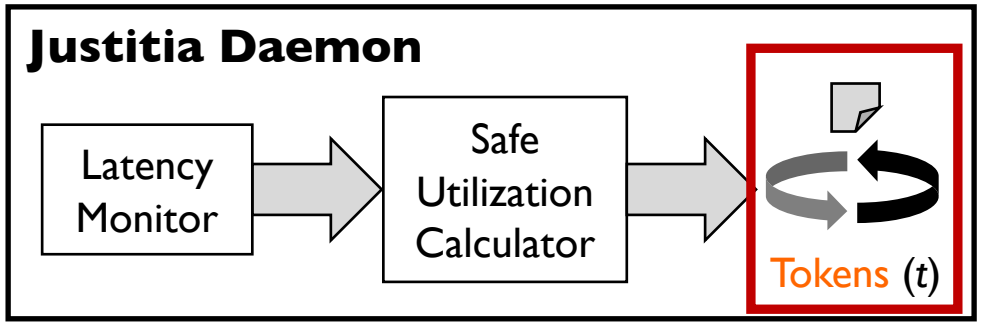
Provides isolation?	Latency	Throughput	Bandwidth
Bandwidth	NO	NO	NO
Throughput	YES	YES	
Latency	YES		



Justitia Daemon

Multi-Resource Token

- Used to rate-limit both bandwidth- and throughput-sensitive apps
- Represent either resource based on the application type

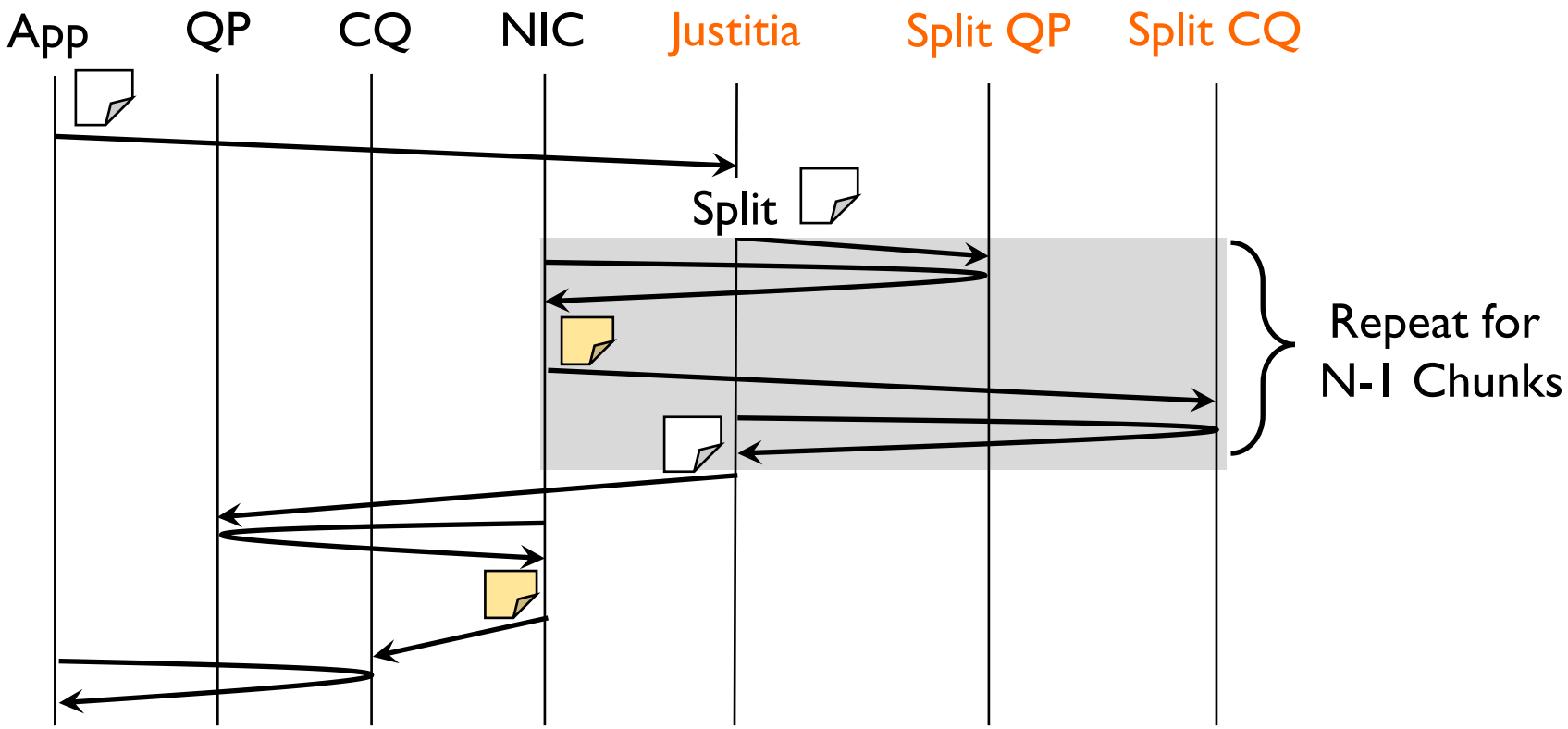


One Token → Bandwidth-sensitive app: One spilt chunk

One Token → Throughput-sensitive app: Equivalent number of messages

Justitia Shaper

Transparently split messages with *Split QP*



More in Our Paper

- How does the shaper works
  - Dynamic receiver-side updates
  - Batch pacing for throughput-sensitive apps
- Calculating Safe Utilization
- Determining token sizes

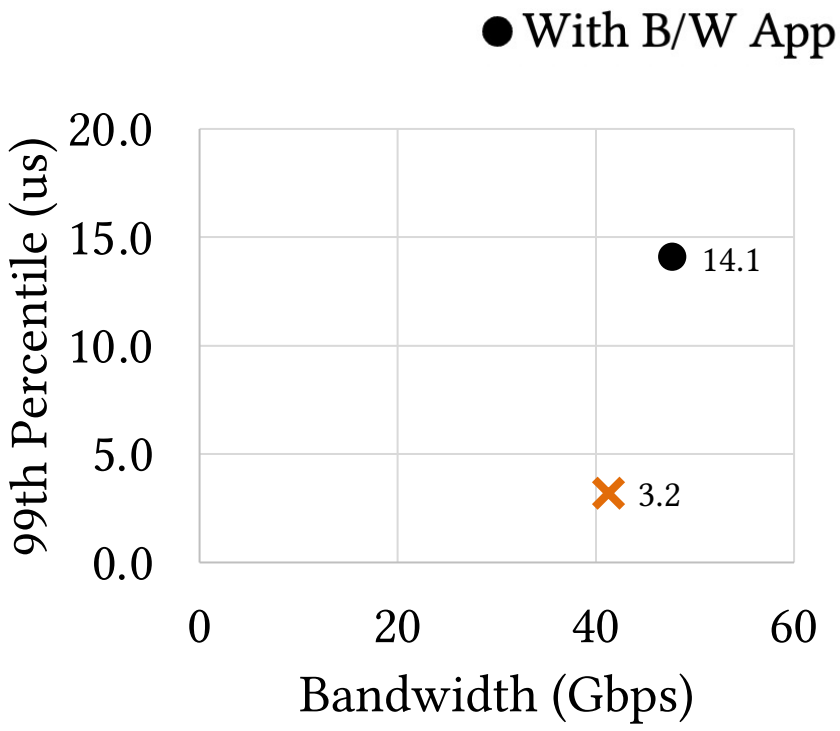
Evaluation

Evaluated Justitia on both InfiniBand and RoCEv2

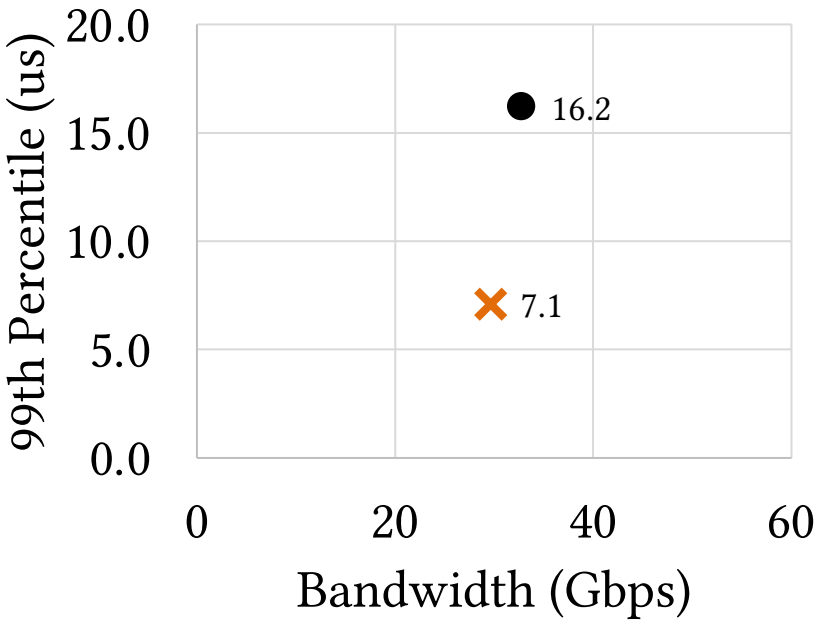
1. Provides isolation while maximizing utilization.
2. Works well for real RDMA-based applications.
3. Complements congestion control in incast scenarios.
4. Handles unexpected network congestion.
5. …

Better Isolation and Utilization

Latency Target = 10us

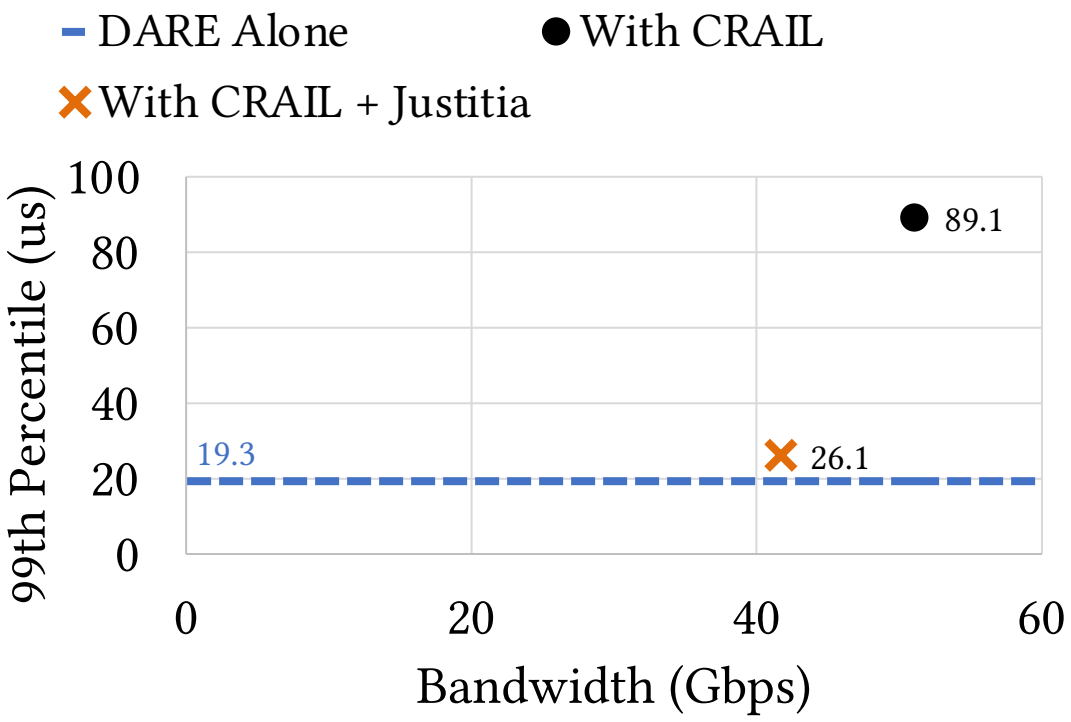


InfiniBand



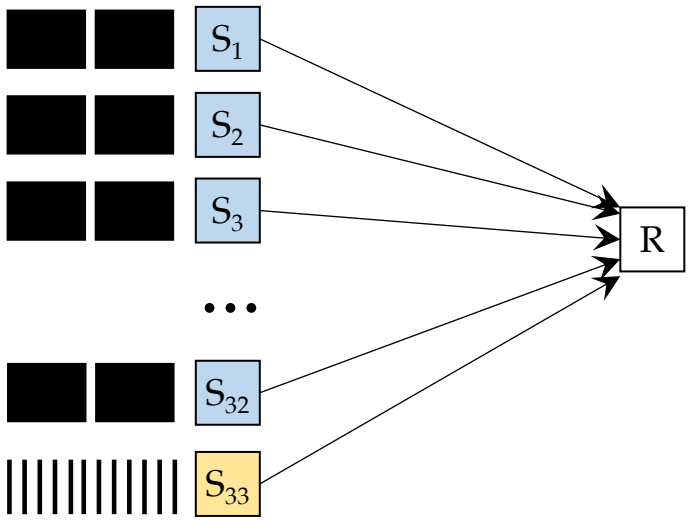
RoCE

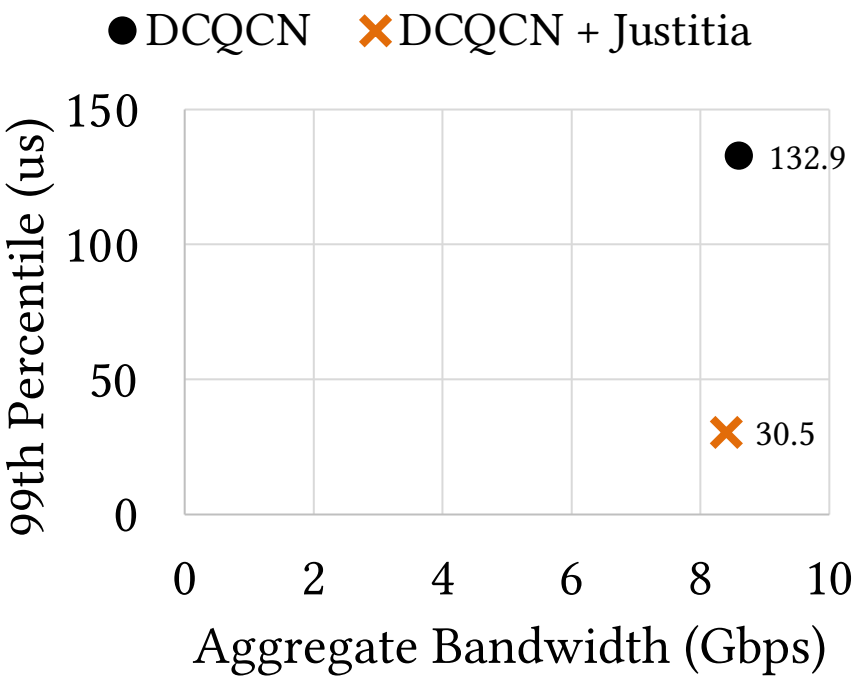
Isolate Real Applications: DARE vs Crail



InfiniBand

Complements with Congestion Control





RoCE

- More details on how we handle incast can be found in the paper

Justitia

https://github.com/SymbioticLab/Justitia

Provides multi-tenancy support for hardware-based KBN

Provides multi-tenancy support in a

- efficient,
- scalable, and
- transparent manner

without modifying

- applications,
- operating systems, or
- hardware

Thank you!

University of Michigan