A Model for Application Slowdown Estimation in On-Chip Networks and Its Use for Improving System Fairness and Performance

Xiyue Xiang*, Saugata Ghose†, Onur Mutlu†§, Nian-Feng Tzeng*

*University of Louisiana at Lafayette

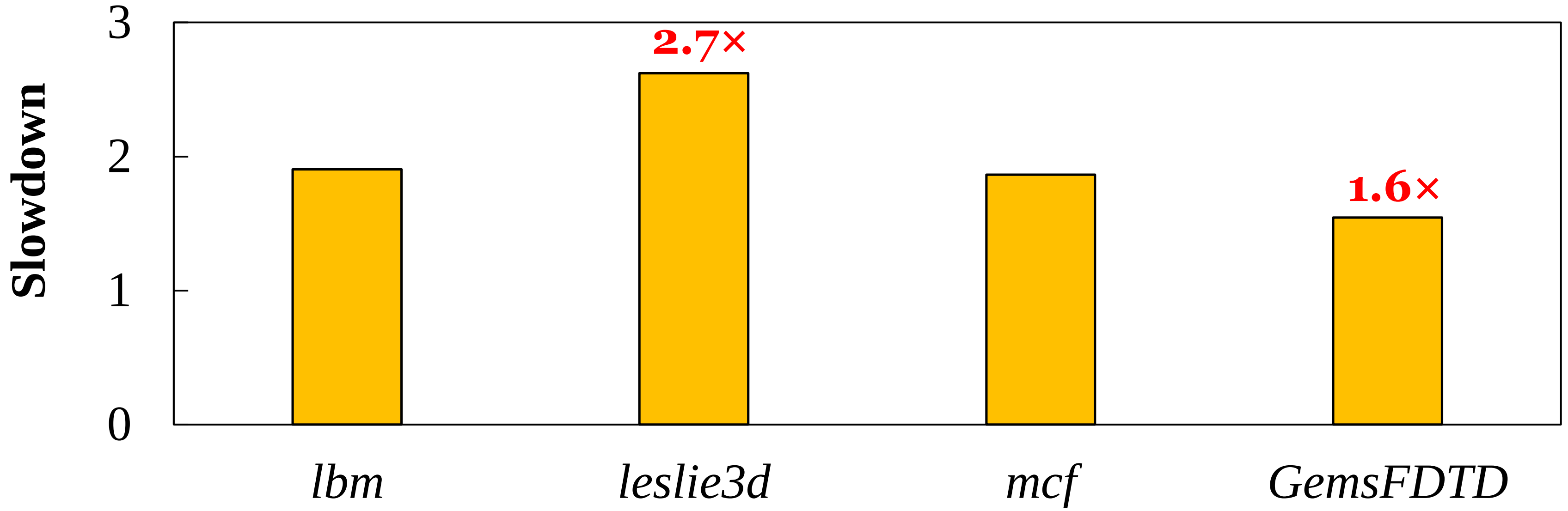
†Carnegie Mellon University

§ETH Zürich

Executive Summary

- **Problem**: **inter-application interference** in on-chip networks (NoCs)
  - In a multicore processor, interference can occur due to NoC contention
  - Interference causes applications to **slow down *unfairly***
- **Goal**: estimate NoC-level slowdown at runtime, and use slowdown information to improve system fairness and performance
- **Our Approach**
  - **NoC Application Slowdown Model (NAS)**: first **online model** to quantify inter-application interference in NoCs
  - **Fairness-Aware Source Throttling (FAST)**: **throttle** network injection rate of processor cores **based on slowdown estimate** from NAS
- **Results**
  - **NAS** is ***very accurate and scalable***: 4.2% error rate on average (8×8 mesh)
  - **FAST** ***improves system fairness*** by 9.5%, ***and performance*** by 5.2% (compared to a baseline without source throttling on a 8×8 mesh)

Motivation: Interference in NoCs



16 copies of each application run concurrently on a 64-core processor

$$slowdown = \frac{t_{interference}}{t_{no\_interference}} = \frac{t_{shared}}{t_{alone}}$$

Root cause:

NoC bandwidth is shared

Interference slows down applications and increases system unfairness

NAS: NoC Application Slowdown Model

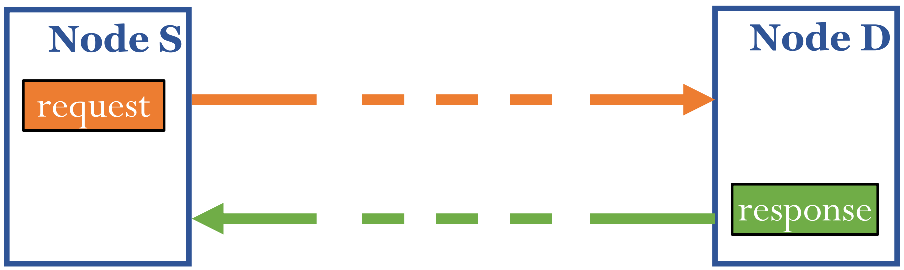
t_{shared}: measured directly t_{alone}: unknown at runtime

$$slowdown = \frac{t_{shared}}{t_{alone}} = \frac{t_{shared}}{t_{shared} - \Delta t_{stall}}$$

Online estimation of Δt_{stall}: *application stall time due to interference*

Challenges:

- Flit-level delay ≠ slowdown



Each request involves multiple packets

NAS: NoC Application Slowdown Model

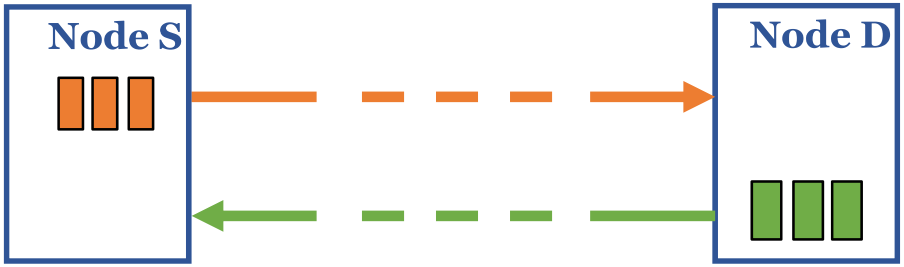
t_{shared}: measured directly t_{alone}: unknown at runtime

$$slowdown = \frac{t_{shared}}{t_{alone}} = \frac{t_{shared}}{t_{shared} - \Delta t_{stall}}$$

Online estimation of Δt_{stall}: application stall time due to interference

Challenges:

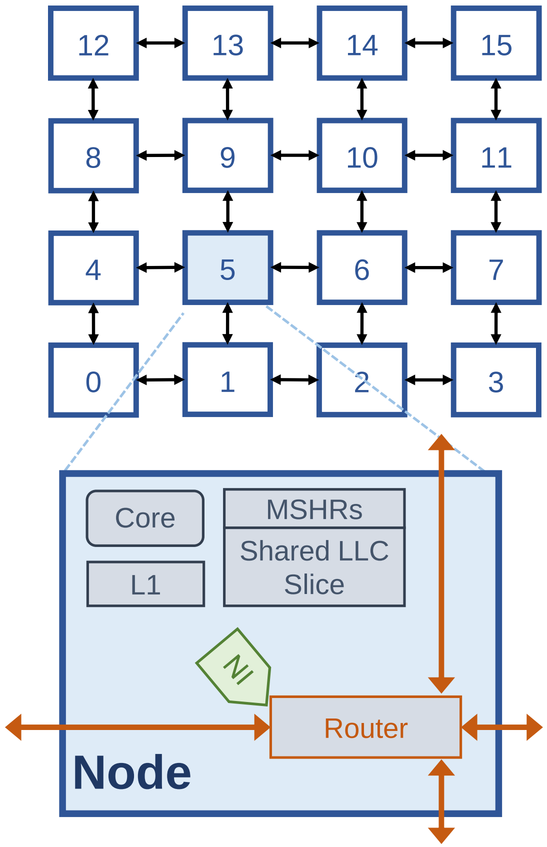
- Flit-level delay ≠ slowdown
- Random and distributive
- Overlapped delay



A packet is formed by multiple flits

Basic idea: track delay and calculate Δt_{stall}

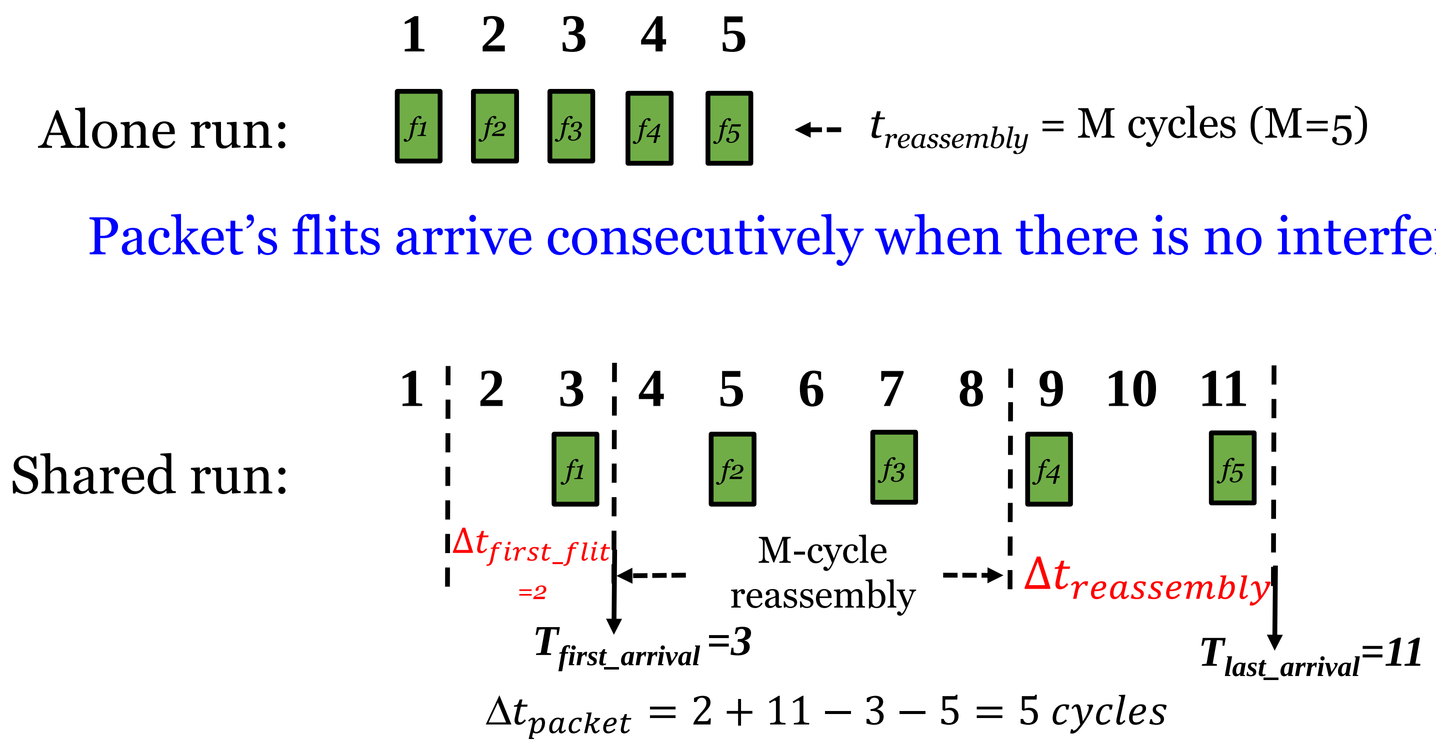
Flit-Level Interference



- **Three interference events**
  - Injection
  - Virtual channel arbitration
  - Switch arbitration
- Each flit carries an additional field Δt_{flit}
  - If arbitration loses, $\Delta t_{flit} = \Delta t_{flit} + 1$

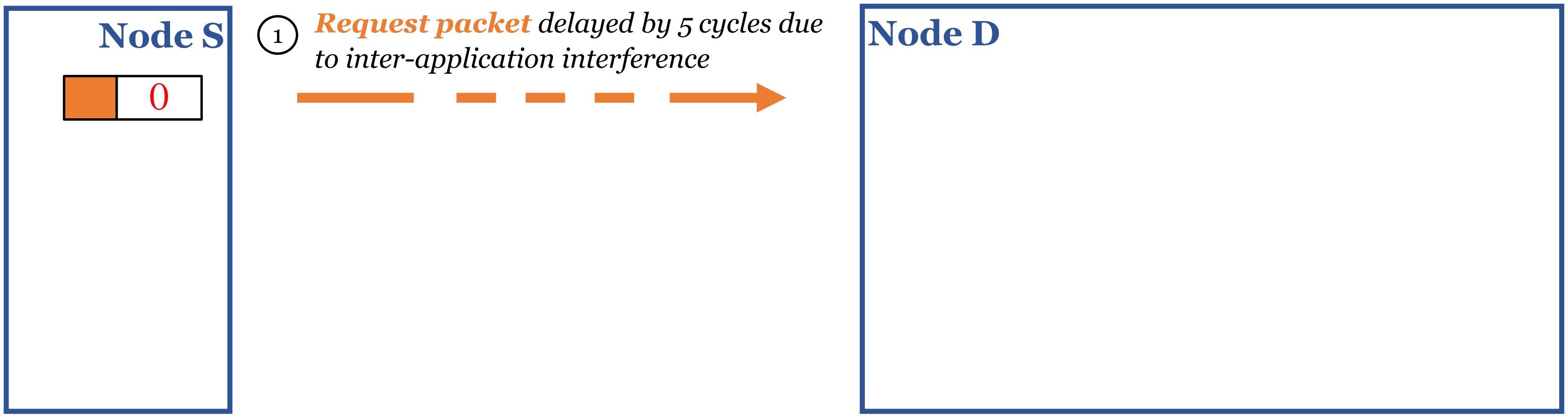
Sum up arbitration delays due to interference

Packet-Level Interference



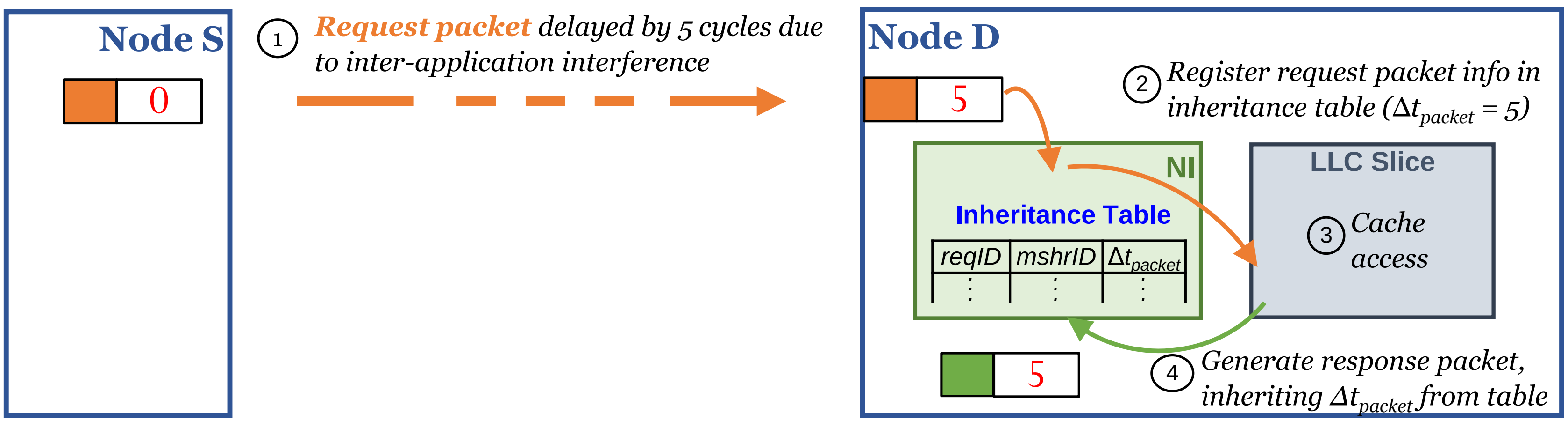
Track increase in packet reassembly time

Request-Level Interference



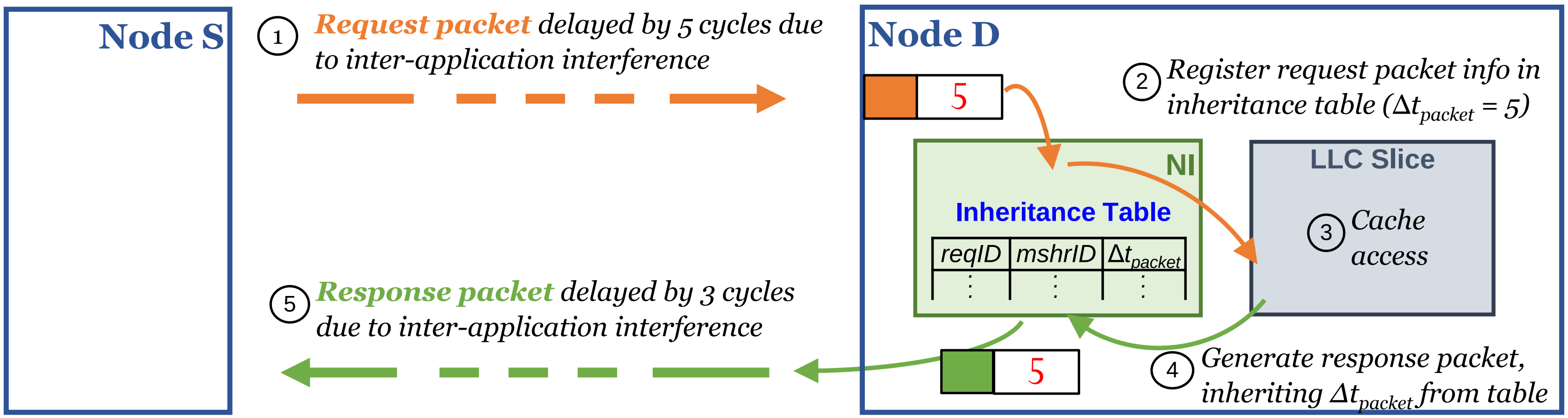
- Leverage **closed-loop** packet behavior to accumulate Δt_{packet}
- Inheritance Table: **lump sum of Δt_{packet}** for associated packets

Request-Level Interference



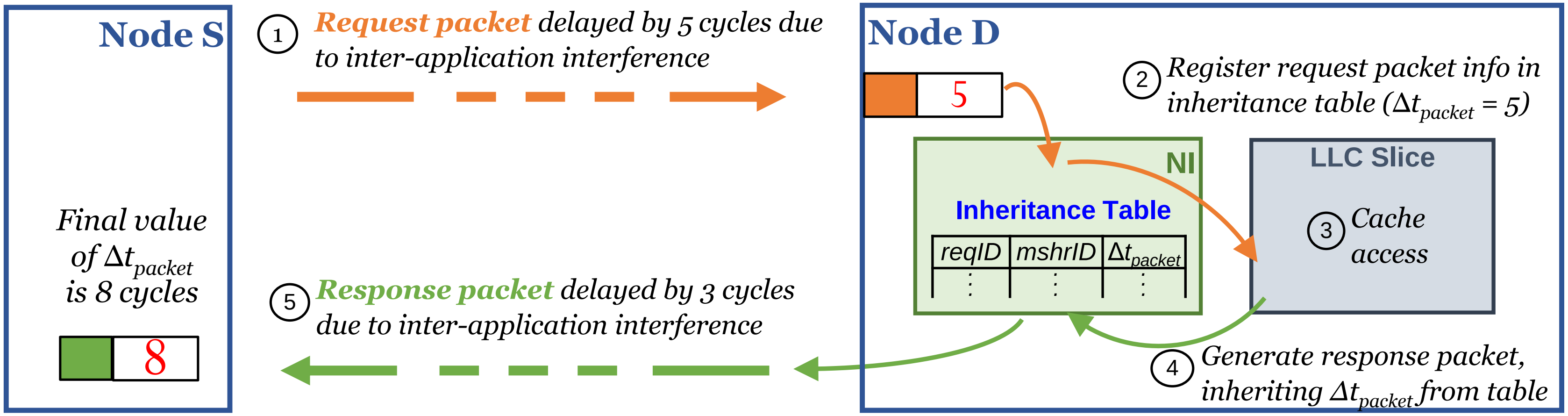
- Leverage **closed-loop** packet behavior to accumulate Δt_{packet}
- Inheritance Table: **lump sum of Δt_{packet}** for associated packets

Request-Level Interference



- Leverage **closed-loop** packet behavior to accumulate Δt_{packet}
- Inheritance Table: **lump sum of Δt_{packet}** for associated packets

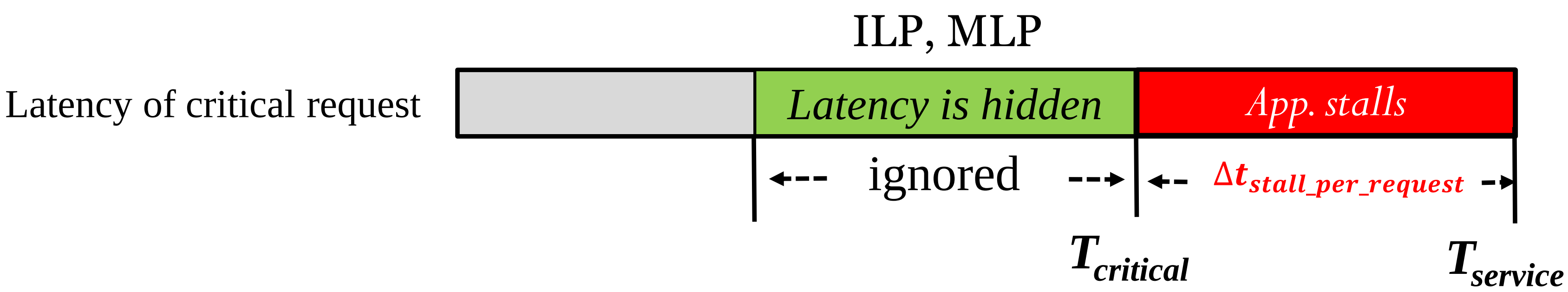
Request-Level Interference



- Leverage **closed-loop** packet behavior to accumulate Δt_{packet}
- Inheritance Table: **lump sum of Δt_{packet}** for associated packets

Sum up delays of all associated packets

Application Stall Time



A memory request becomes critical if

1) It is the oldest instruction at ROB and ROB is full, and/or
2) It is the oldest instruction at LSQ and LSQ is full when the next is a memory instruction

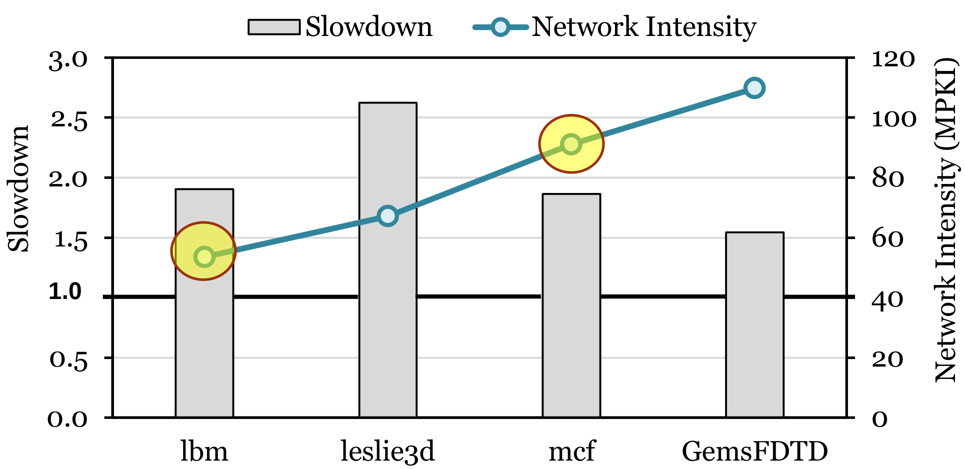
For all critical requests

Count only request delays on critical path of execution time

Using NAS to Improve Fairness

- NAS provides **online** estimation of slowdown
  - Sum up flit-level arbitration delays due to interference
  - Track increase in packet reassembly time
  - Sum up delays of all associated packets
  - Determine which request delays causes application stall
- **Goal**
  - Use NAS to **improve system fairness and performance**
- **FAST: Fairness-Aware Source Throttling**

A New Metric: NoC Stall-Time Criticality



Interference in NoCs has uneven impact

NoC Stall-Time Criticality

$$STC_{noc} = \frac{slowdown}{L1\ miss}$$

Lower STC_{noc} <==> Less sensitive to NoC-level interference

Good candidate to be throttled down

FAST utilizes STC_{noc} to <u>proactively</u> estimate the expected impact of each L1 miss

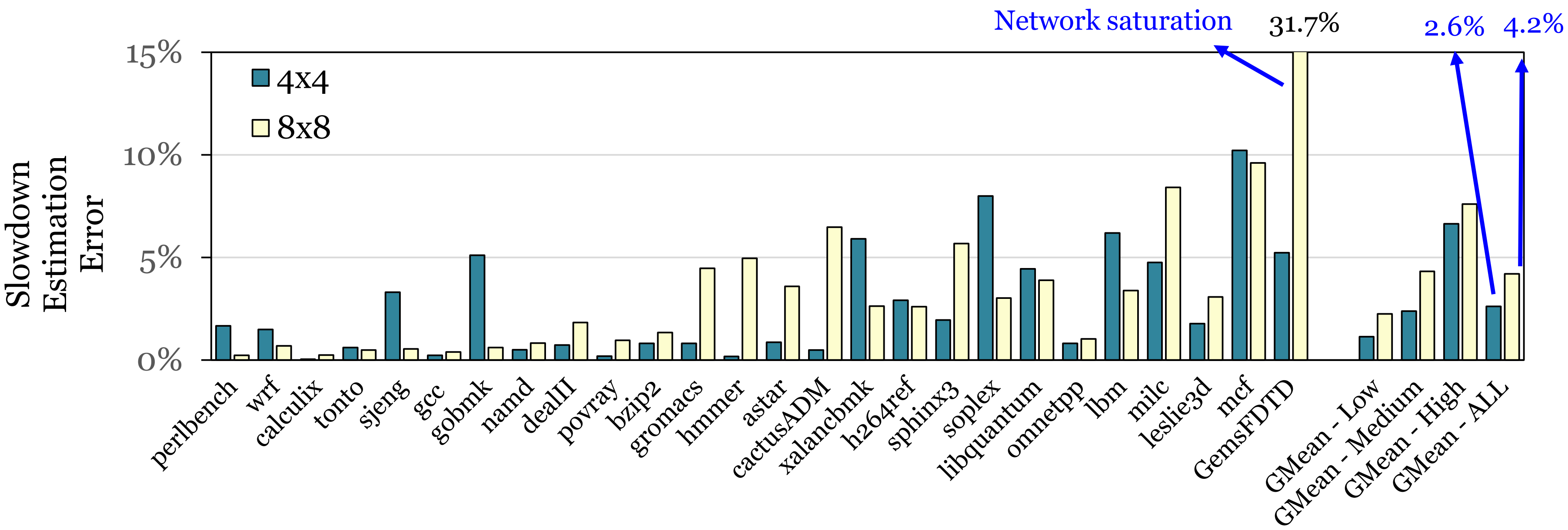
Key Knobs of FAST

- **Rank** **based on** ***slowdown***
- **Classification** **based on** ***network intensity***
  - Latency-sensitive: spends more time in the core
  - Throughput-sensitive: network intensive
- **Throttle Up**
  - Latency-sensitive applications: improve system performance
  - Slower applications: optimize system fairness
- **Throttle Down**
  - Throughput sensitive application with lower STC_{noc}: reduce interference with lower negative impact on performance
  - Avoid throttling down the slowest application

Methodology

- **Processor**
  - Out-of-order, ROB / instruction window = 128
- **Caches**
  - L1: 64KB, 16 MSHRs
  - L2: perfect shared
- **NoCs**
  - Topology: 4×4 and 8×8 mesh
  - Router: conventional VC router with 8 VCs, 4 flits/VC
- **Workloads: multiprogrammed SPEC CPU2006**
  - 90 randomly-chosen workloads
  - Categorized by network intensity (i.e., MPKI)

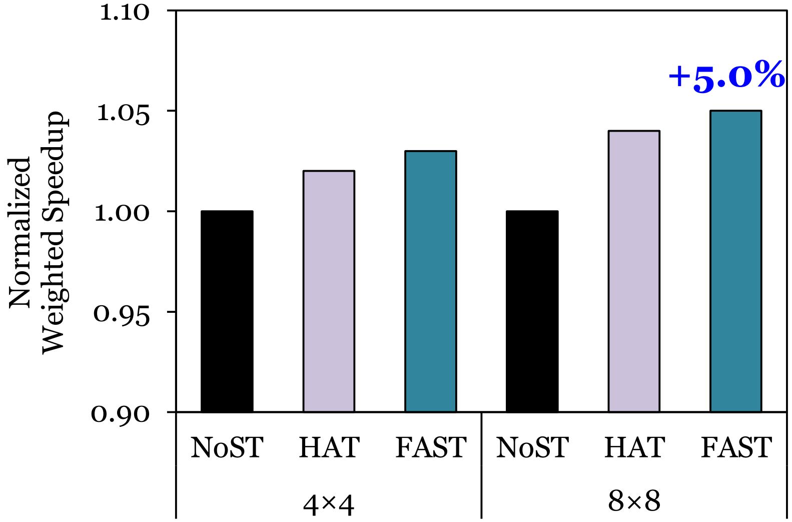
NAS is Accurate



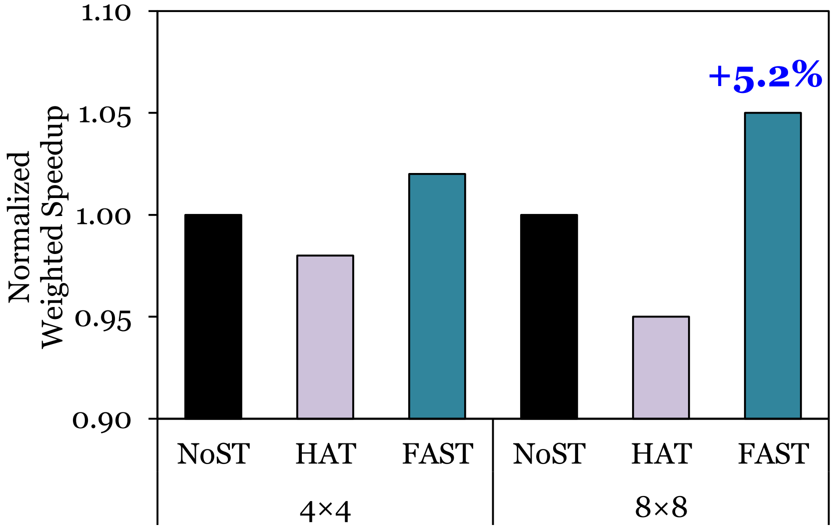
- Slowdown estimation error: 4.2% (2.6%) for 8×8 (4×4)

NAS is highly accurate and scalable

FAST Improves Performance



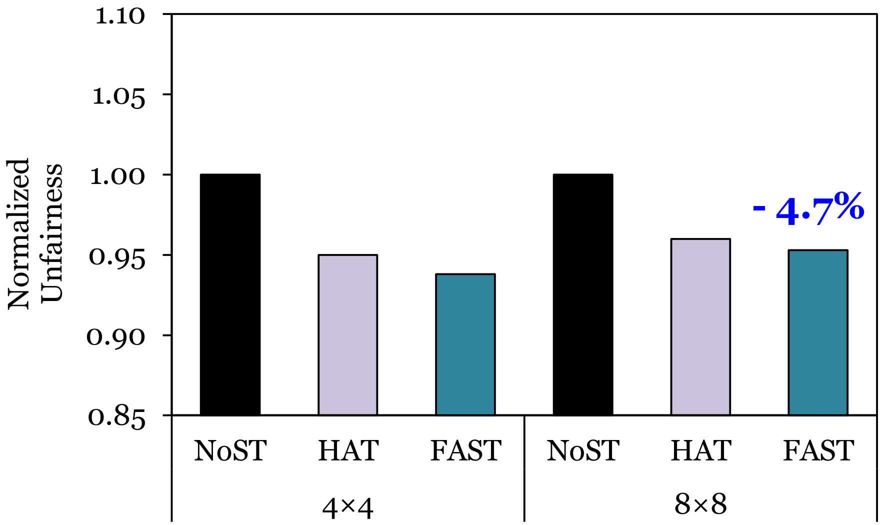
(a) Mixed workloads



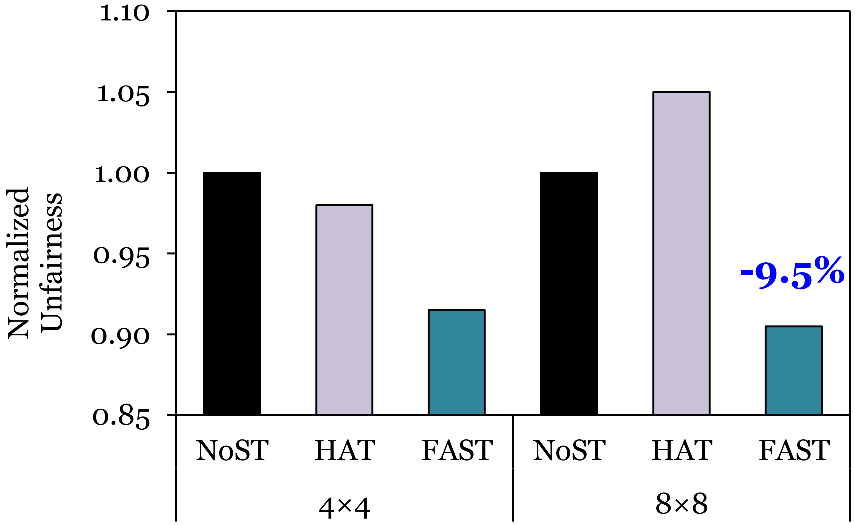
(b) Heavy workloads

- FAST has better performance than both HAT and NoST
  - Inter-application interference is reduced
  - Only throttles applications with low negative impact (i.e., lower STC_{noc})

FAST Reduces Unfairness



(a) Mixed workloads



(b) Heavy workloads

- FAST can improve fairness
  - Source throttling allows slower applications to catch up
  - Uses runtime slowdown to identify and avoid throttling the slowest application

Conclusion

- **Problem**: **inter-application interference** in on-chip networks (NoCs)
  - In a multicore processor, interference can occur due to NoC contention
  - Interference causes applications to **slow down *unfairly***
- **Goal**: estimate NoC-level slowdown at runtime, and use slowdown information to improve system fairness and performance
- **Our Approach**
  - **NoC Application Slowdown Model (NAS)**: first **online model** to quantify inter-application interference in NoCs
  - **Fairness-Aware Source Throttling (FAST)**: **throttle** network injection rate of processor cores **based on slowdown estimate** from NAS
- **Results**
  - **NAS** is ***very accurate and scalable***: 4.2% error rate on average (8×8 mesh)
  - **FAST** ***improves system fairness*** by 9.5%, ***and performance*** by 5.2% (compared to a baseline without source throttling on a 8×8 mesh)

A Model for Application Slowdown Estimation in On-Chip Networks and Its Use for Improving System Fairness and Performance

Xiyue Xiang*, Saugata Ghose†, Onur Mutlu†§, Nian-Feng Tzeng*
*University of Louisiana at Lafayette
†Carnegie Mellon University
§ETH Zürich

Backup Slides

Xiyue Xiang*, Saugata Ghose†, Onur Mutlu†§, Nian-Feng Tzeng*
*University of Louisiana at Lafayette
†Carnegie Mellon University
§ETH Zürich

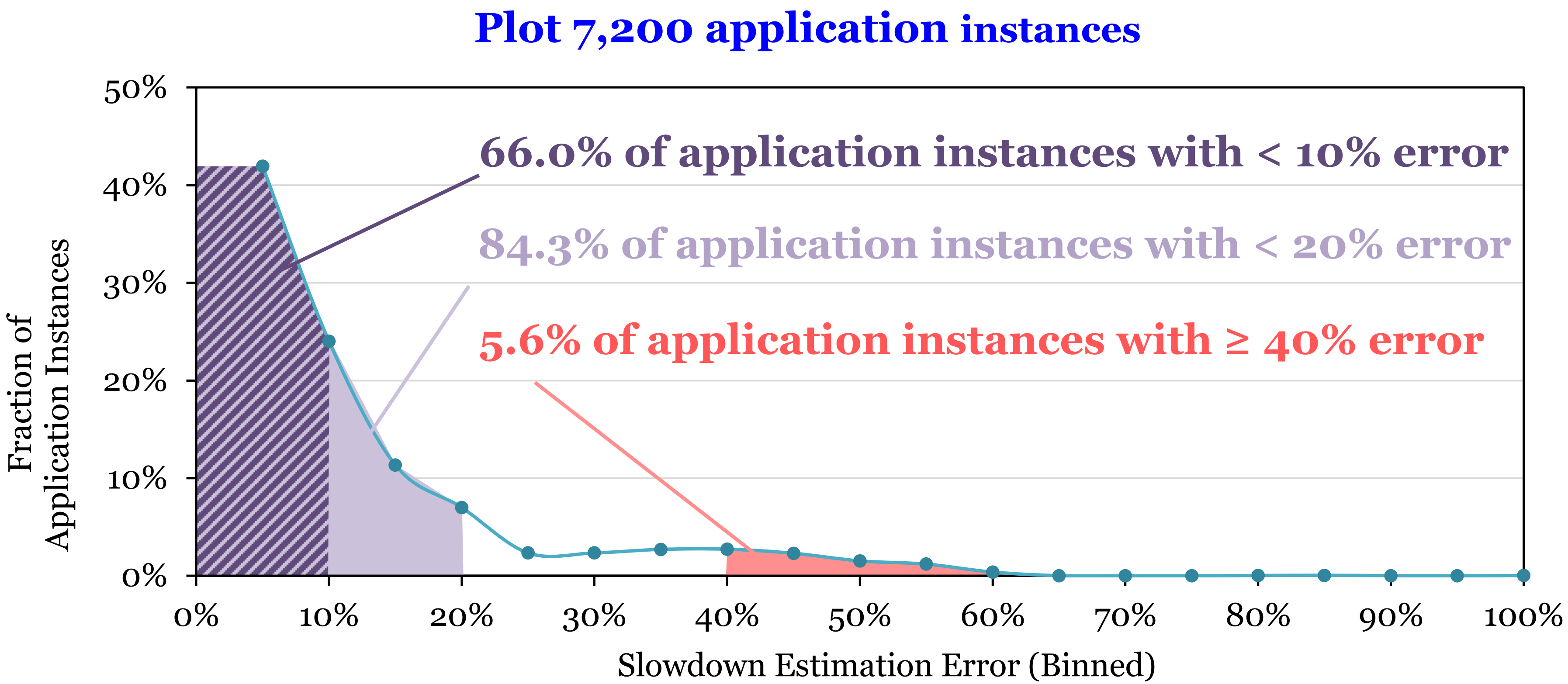
Related Works

- Slowdown modeling
  - Fine grained: [Mutlu+ MICRO '07], [Ebrahimi+ ASPLOS '10], [Bois+ TACO '13]
  - Coarse grained: [Subramanian+ HPCA '13], [Subramanian MICRO '15]
- Source throttling
  - [Chang+ SBAC-PAD '12], [Nychis+ SIGCOMM '12], [Nychis+ HotNet '10]
- Application mapping
  - [Chou+ ICCD '08], [Das+ HPCA '13]
- Prioritization
  - [Das+ MICRO '09], [Das ISCA '10]
- Scheduling
  - [Kim+ MICRO'10]
- QoS
  - [Grot+ MICRO '09], [Grot+ ISCA '11], [Lee+ ISCA '08]

Hardware Cost of NAS

Location	Components	Costs
Router	Interference delay of each flit	5.3% wider data path
NI	Timestamp of the first and last arrival flit of a packet	(16+16)×16 bits
	Inheritance table	(6+4+8)×20 bits
Core	Interference delay of the request	8 bits
	Timestamp when processor stalls	16 bits
	Estimated application stall time	16 bits
Total cost of NAS per node		**114 Bytes + 5.3% router area**

NAS Error Distribution



- Plot 7,200 application instance
- NAS exhibits high accuracy most of the time