

Aérgia: Exploiting Packet Latency Slack in On-Chip Networks

ISCA 2010

Reetuparna Das § Onur Mutlu† Thomas Moscibroda‡ Chita R. Das §

§ Pennsylvania State University †Carnegie Mellon University ‡Microsoft Research

Presented by Olivier Becker

-
- Reetuparna Das, Onur Mutlu, Thomas Moscibroda, Chita R. Das,
“Aérgia: Exploiting Packet Latency Slack in On-Chip Networks”
presented at the 37th Annual ACM IEEE International Symposium on Computer Architecture 2010 (ISCA 2010), Saint-Malo, France, June 2010. [[slides.pptx](#)]

Aérgia: Exploiting Packet Latency Slack in On-Chip Networks

Reetuparna Das
Pennsylvania State University
rdas@cse.psu.edu

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Thomas Moscibroda
Microsoft Research
moscitho@microsoft.com

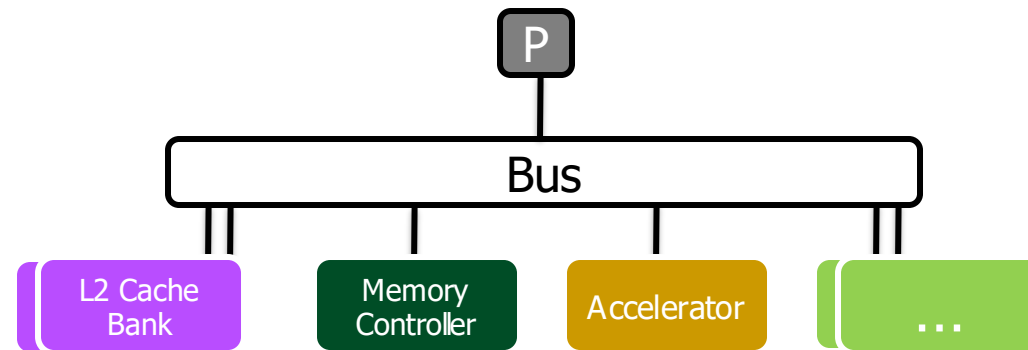
Chita R. Das
Pennsylvania State University
das@cse.psu.edu

Executive Summary

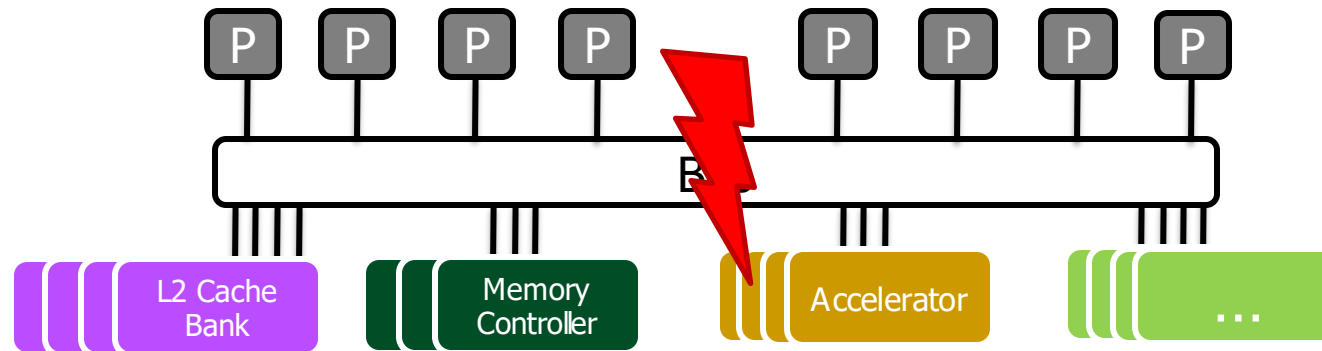
- Problem: Treating all packets equally during scheduling will lead to performance loss
- Goal: Improving overall system performance by accelerating performance-critical packets
- Key Idea: Utilize packet slack to prioritize critical packets
- Key Mechanism: "Predicting" packet latency & prioritizing packets with low slack
- Results:
 - Overall system throughput improved by 10.3%
 - Network fairness improved by 30.8%

Background, Problem & Goal

System-on-Chip

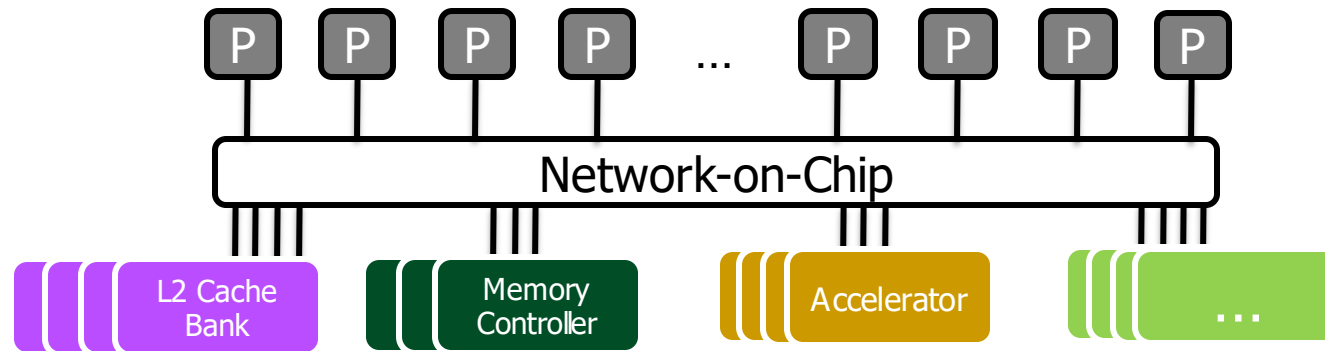


Scalability

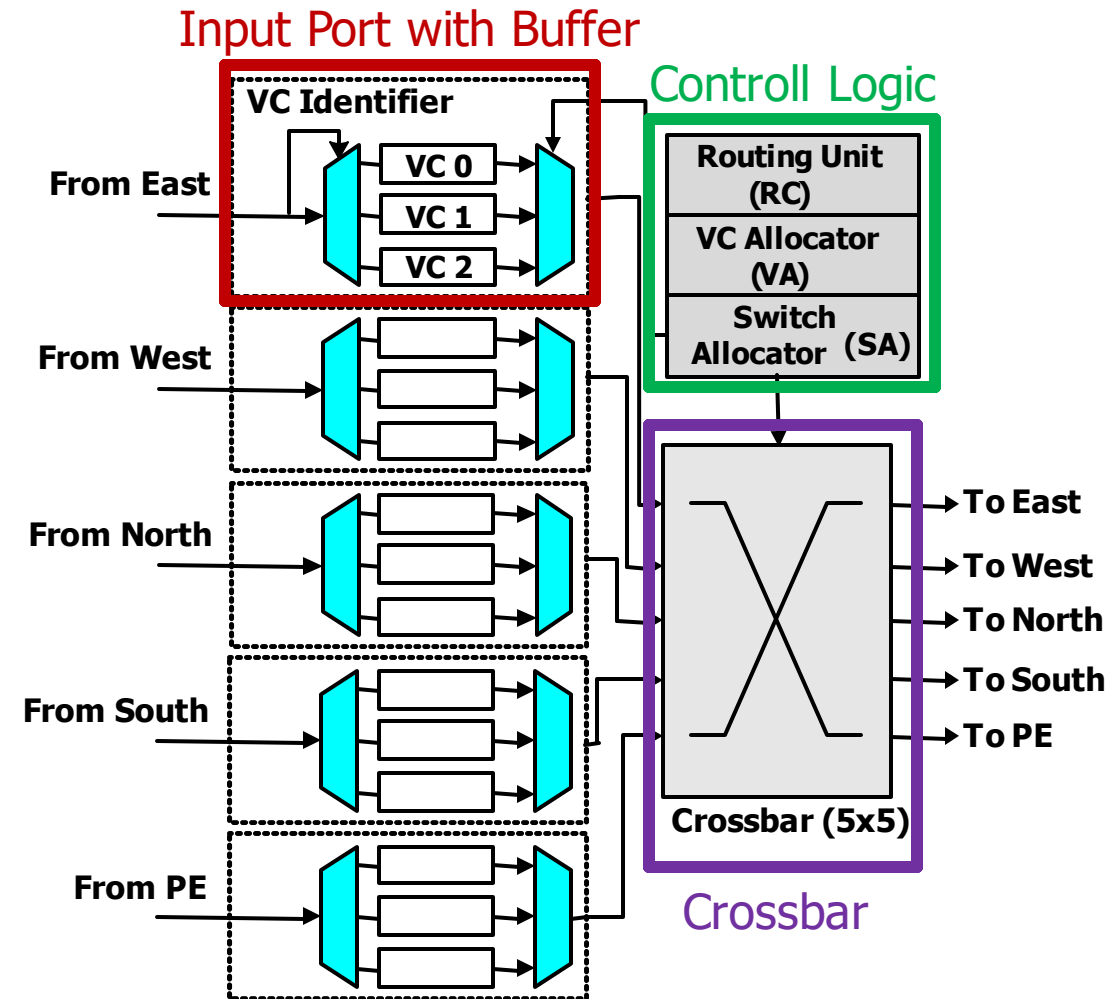
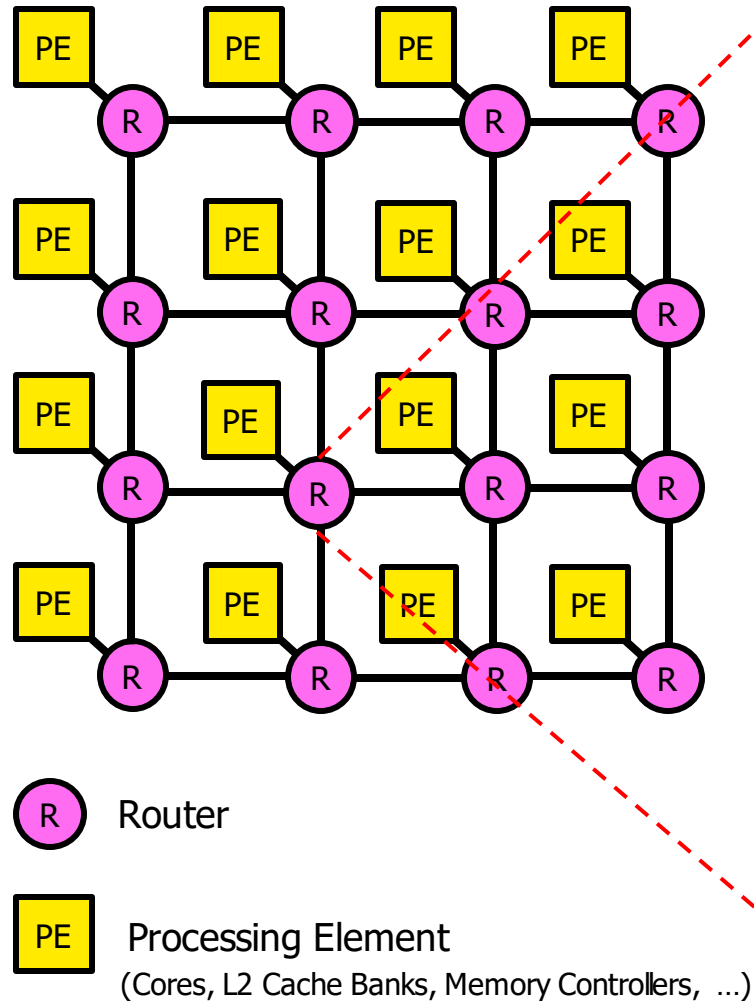


Busses do not scale well!

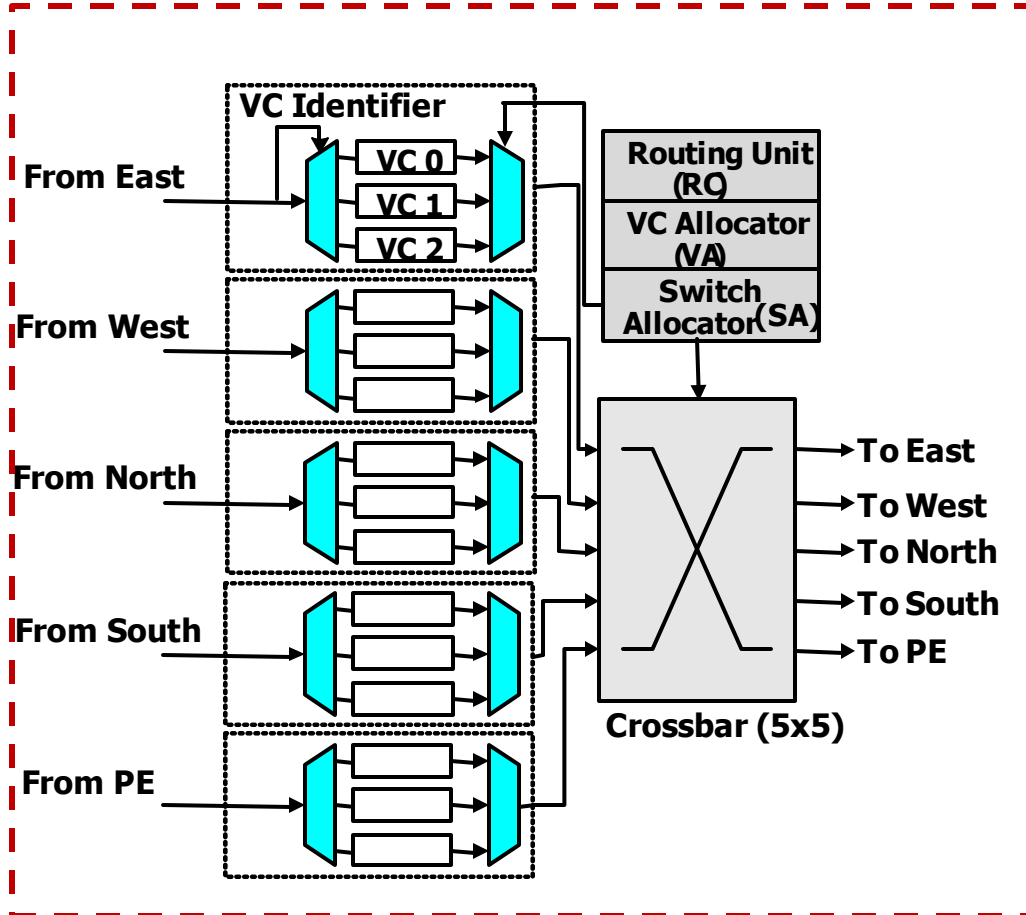
Network-on-Chip



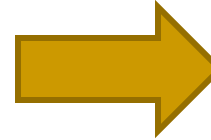
NoC Routers



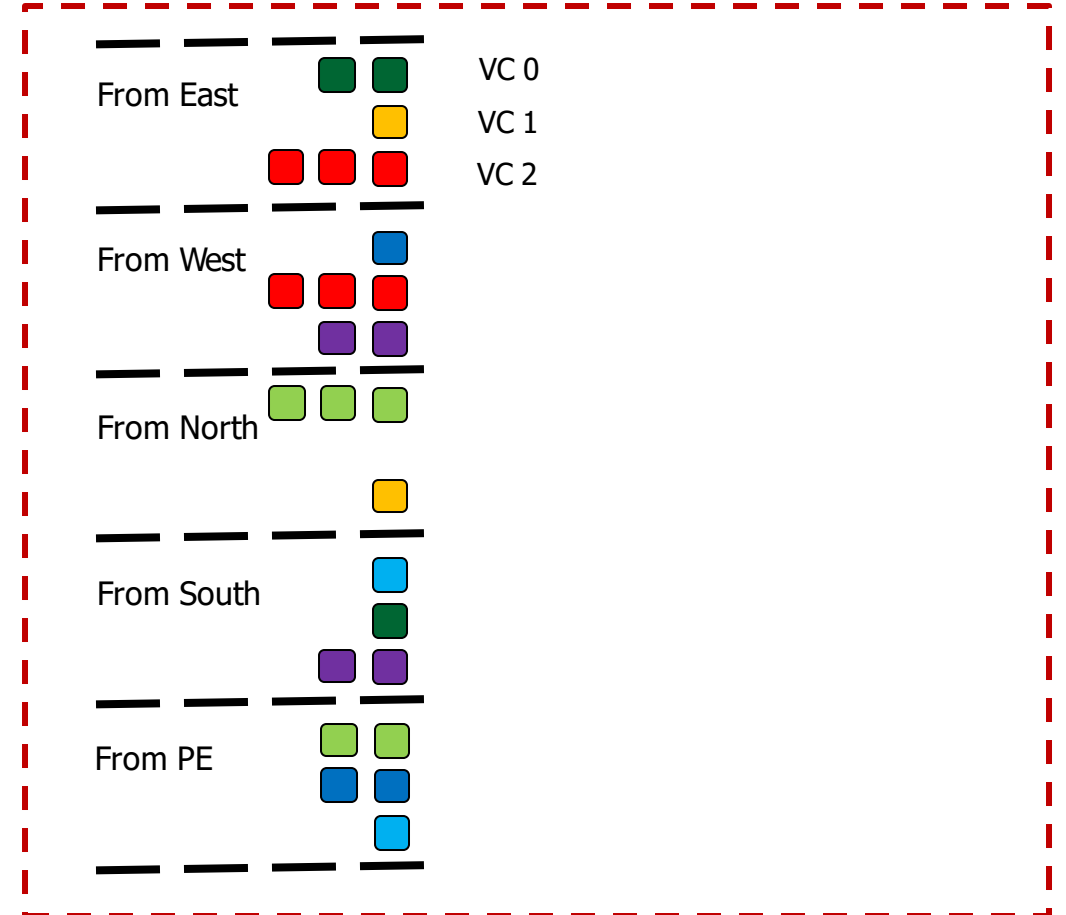
NoC Packet Scheduling



Simplified



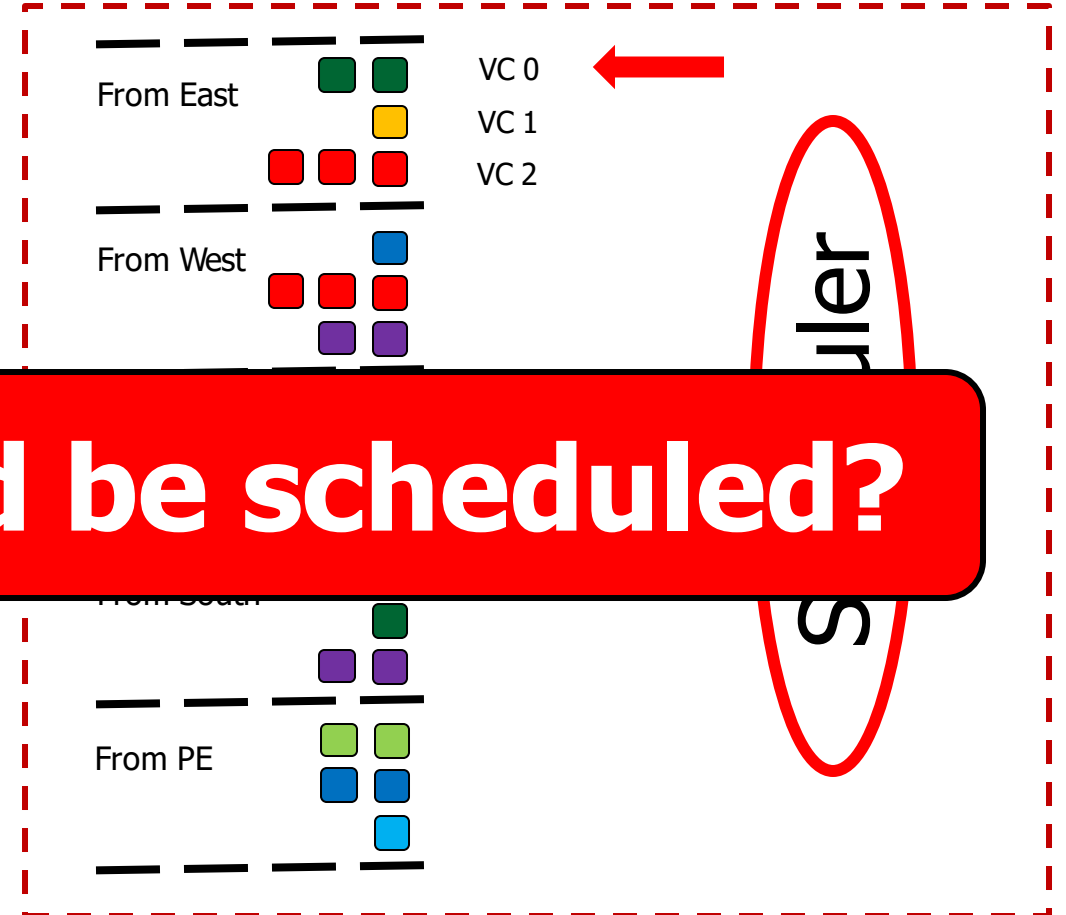
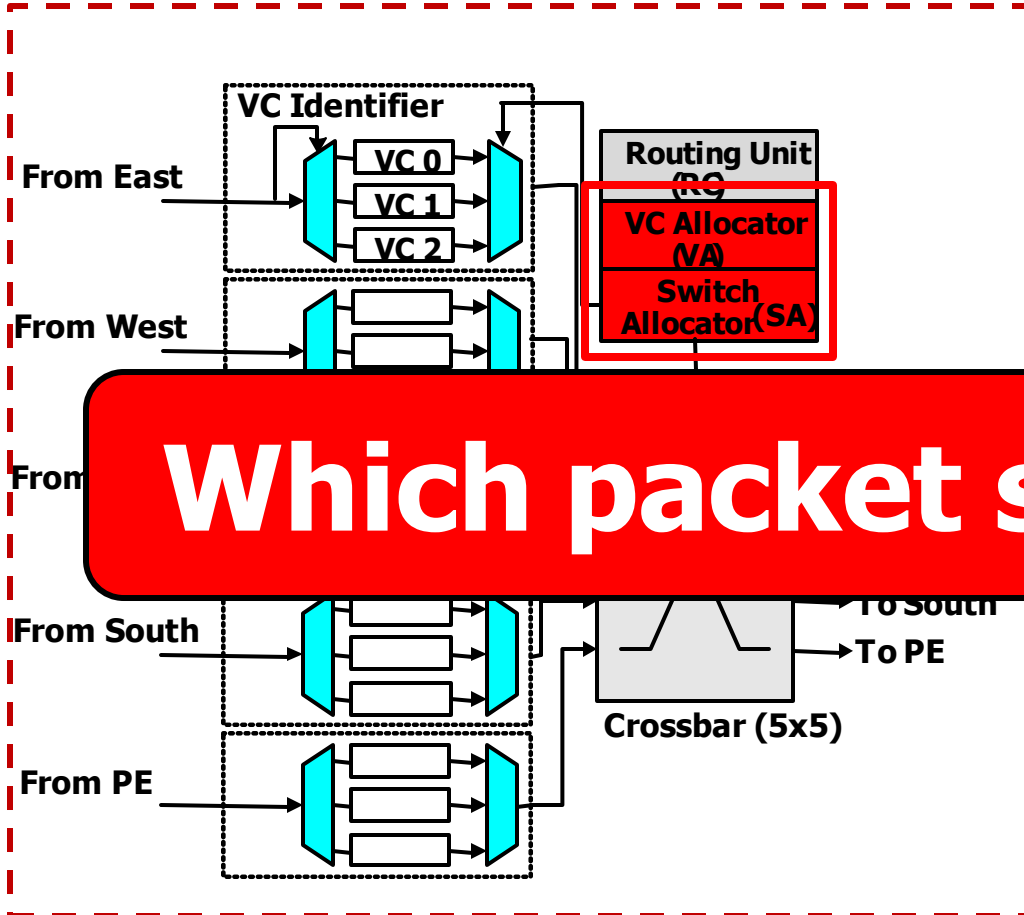
View



Packets from different applications:



NoC Packet Scheduling



Which packet should be scheduled?

Packets from different applications:

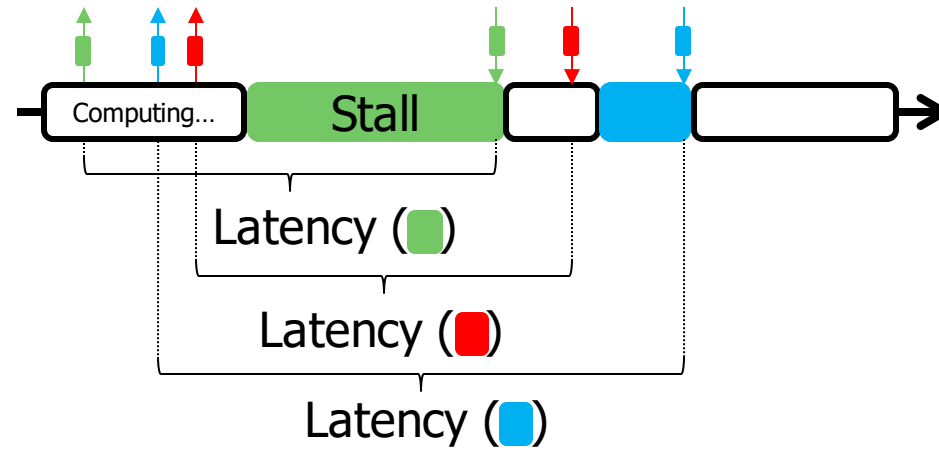


NoC Packet Scheduling

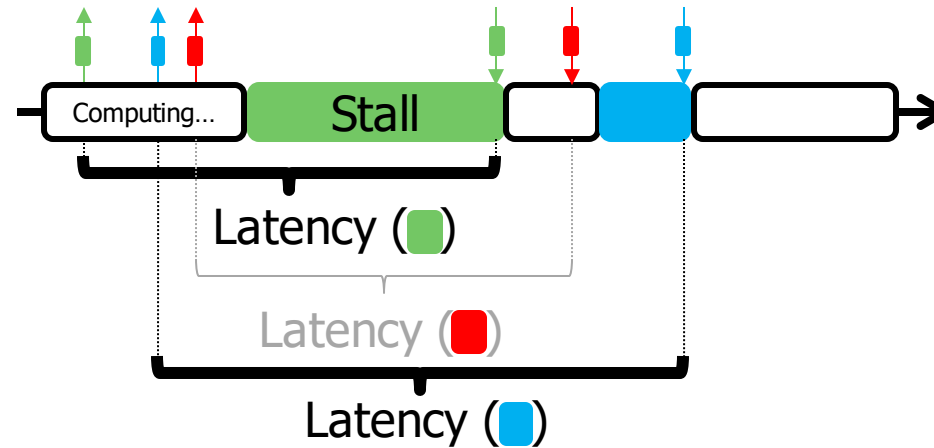
- Current policies:
 - Round robin scheduling
 - Age-based scheduling
- Treat packets from different programs equally
- Treat packets from the same program equally

Different packets can have different effects on system performance!

Memory Level Parallelism (MLP)



Memory Level Parallelism (MLP)



- Due to MLP:
 - Packet Latency \neq Network Stall Time
 - Different packets can have different criticality

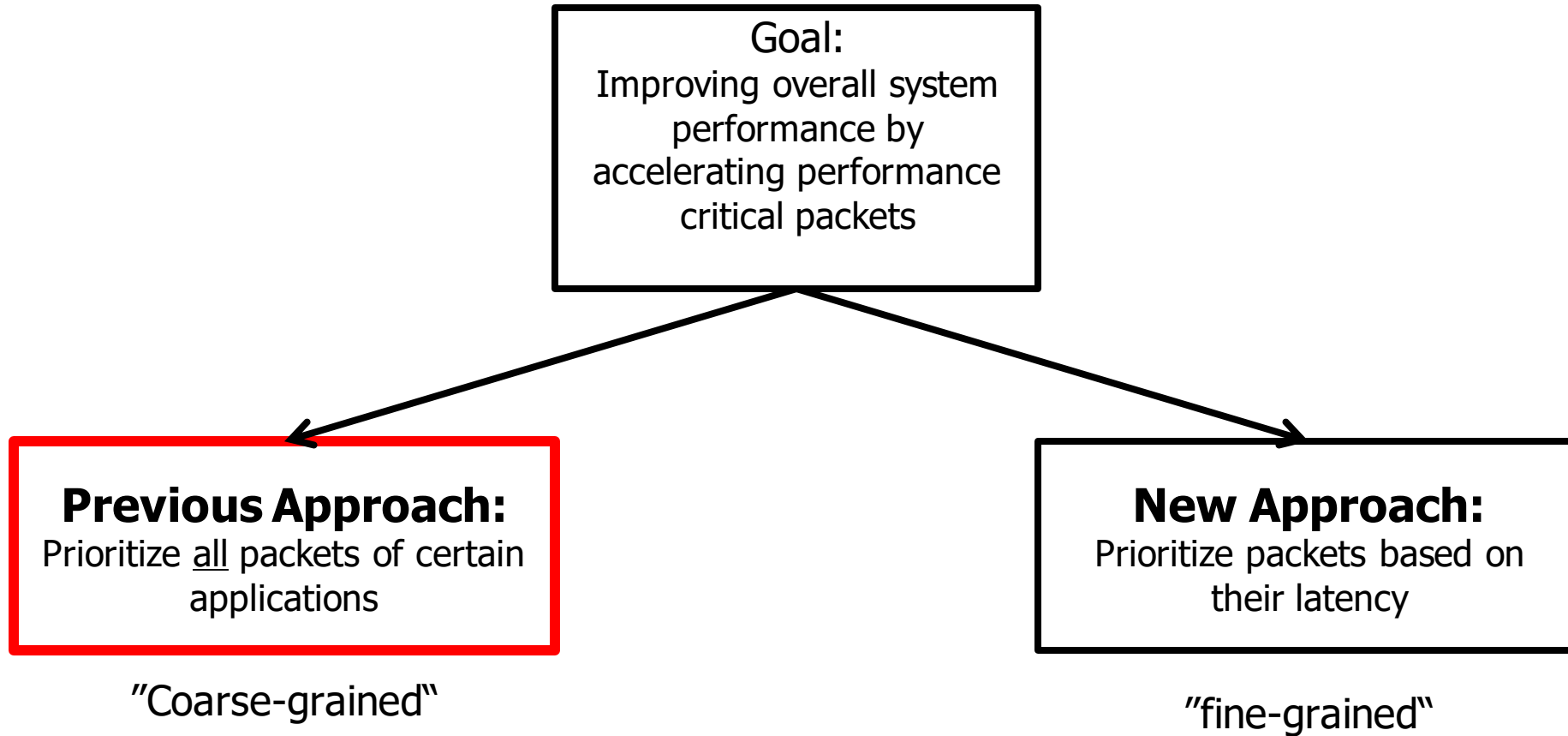
In our Example: Criticality (■) > Criticality (■) > Criticality (■)

Goal

Improving overall system performance by accelerating performance critical packets

Novelty

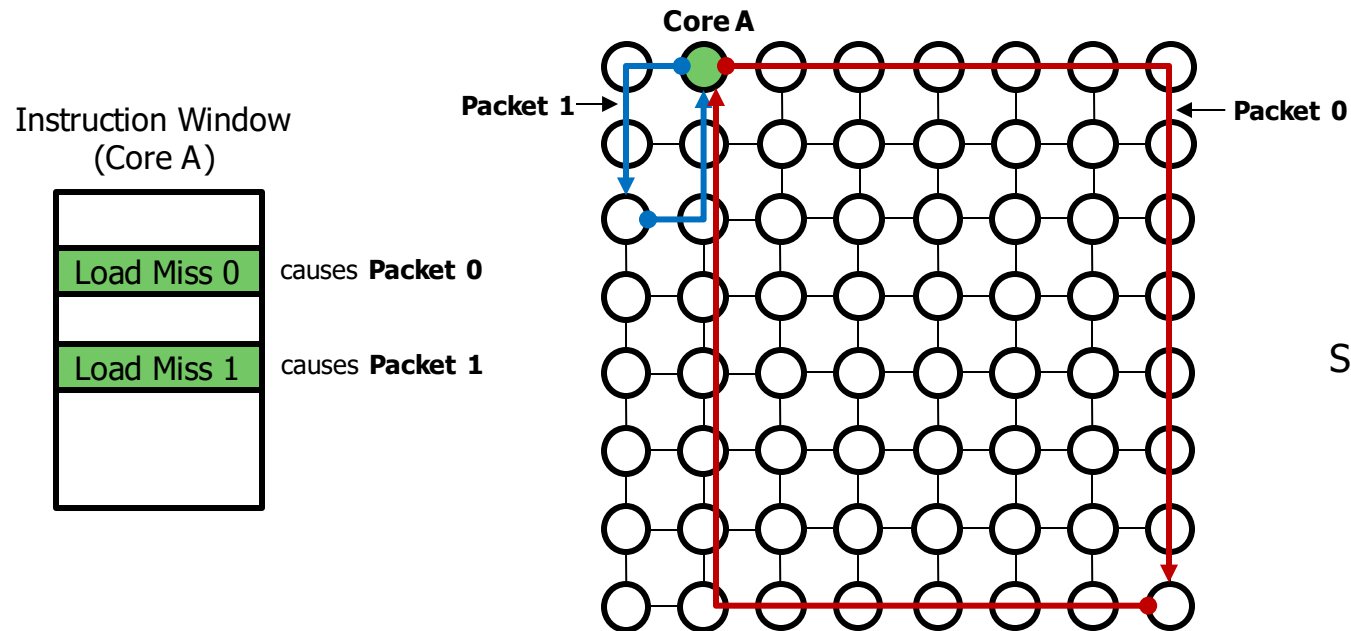
Previous Approaches



Key Approach and Ideas

Slack Intuitively

- The number of cycles a packet can be delayed without affecting application performance is called "Slack"



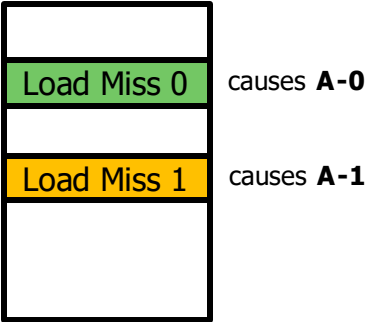
Latency of Packet 0: 26 hops
Latency of Packet 1: 6 hops



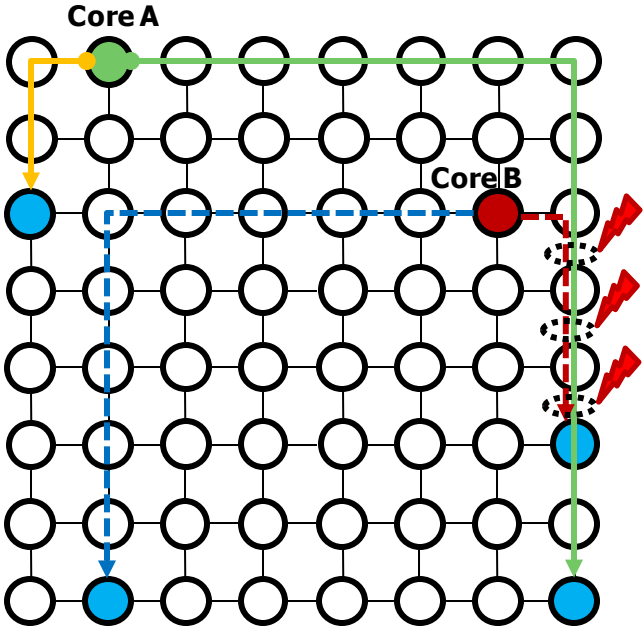
$$\begin{aligned}\text{Slack}(\text{Packet 1}) &= \text{Latency}(\text{Packet 0}) - \text{Latency}(\text{Packet 1}) \\ &= 26 - 6 = 20 \text{ hops}\end{aligned}$$



Key Insight: Accelerating critical Packets

Instruction Window
(Core A)



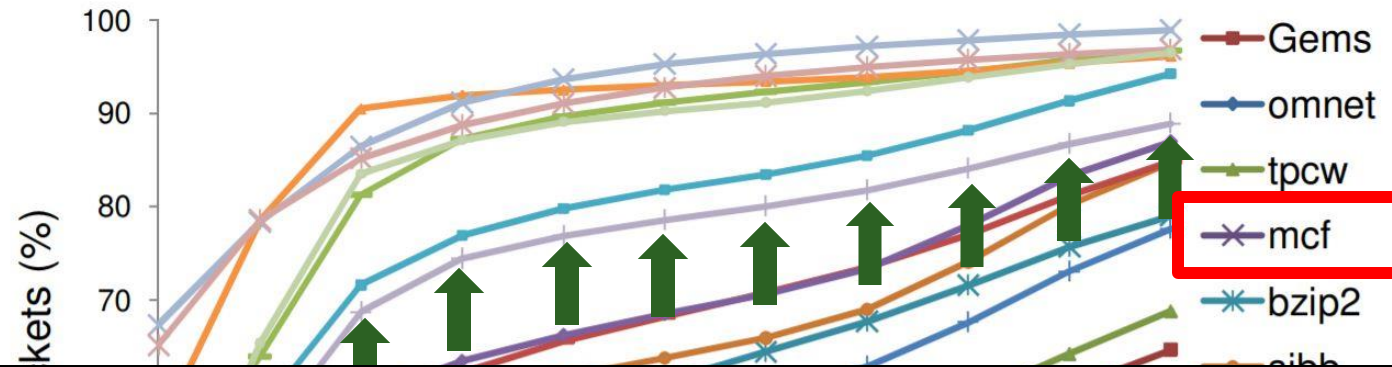
Instruction Window
(Core B)



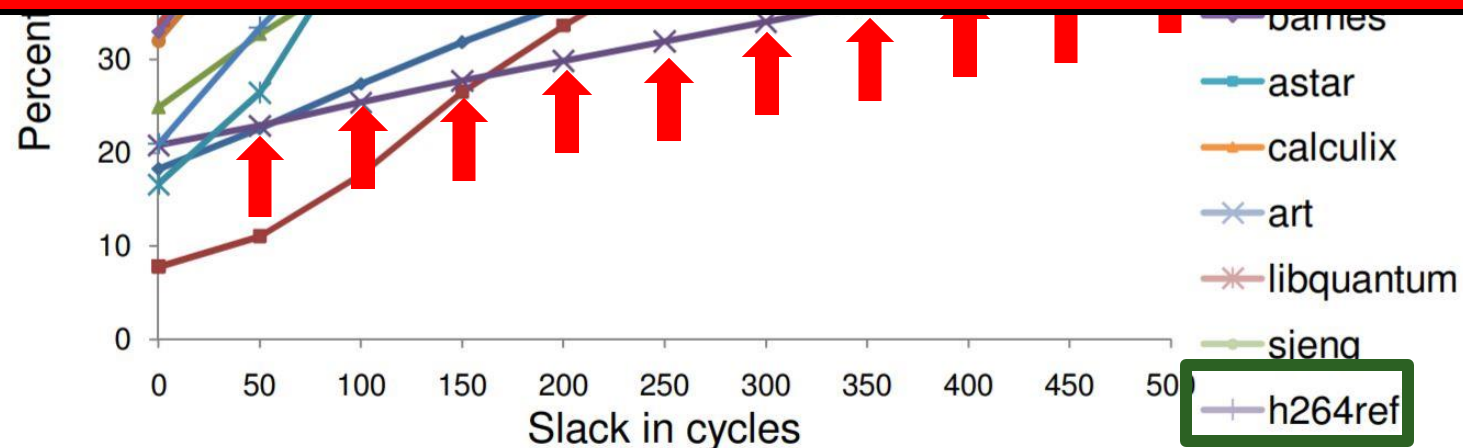
Packet	Latency	Slack
A-0 	13 hops	0 hops
B-1 	3 hops	10 hops

Prioritizing B-1 leads to performance loss!

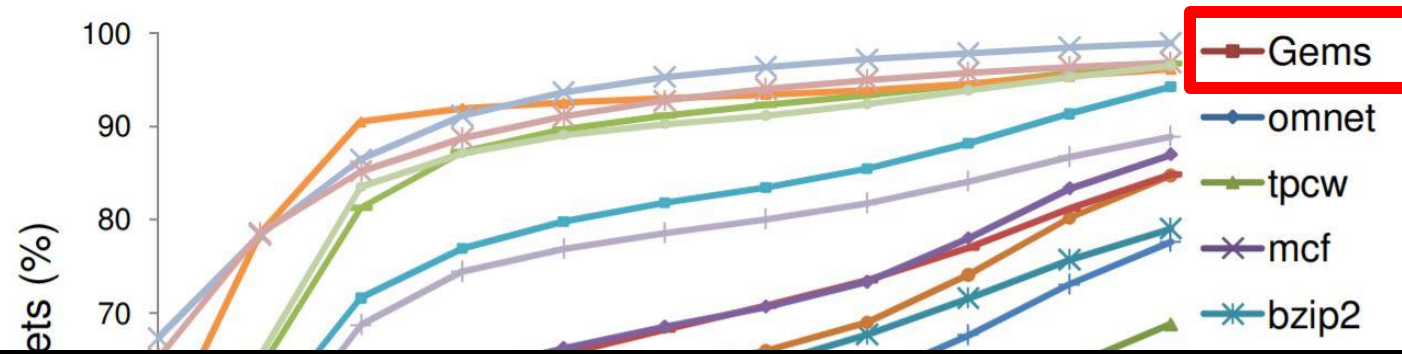
Diversity of Slack



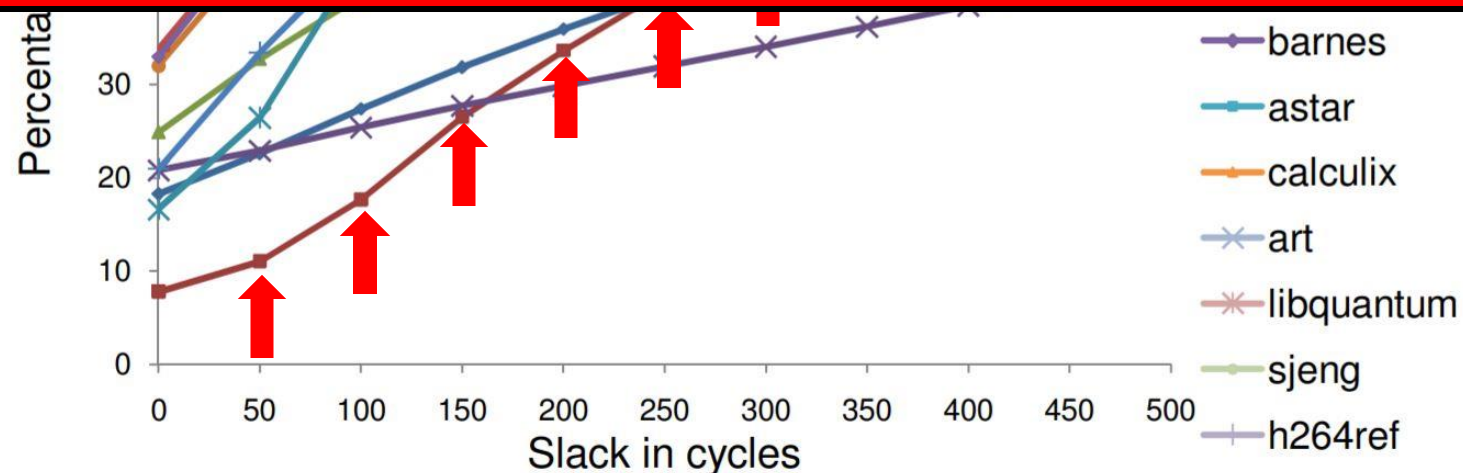
Packet slack varies between different applications!



Diversity of Slack



Packet slack varies within the same application!



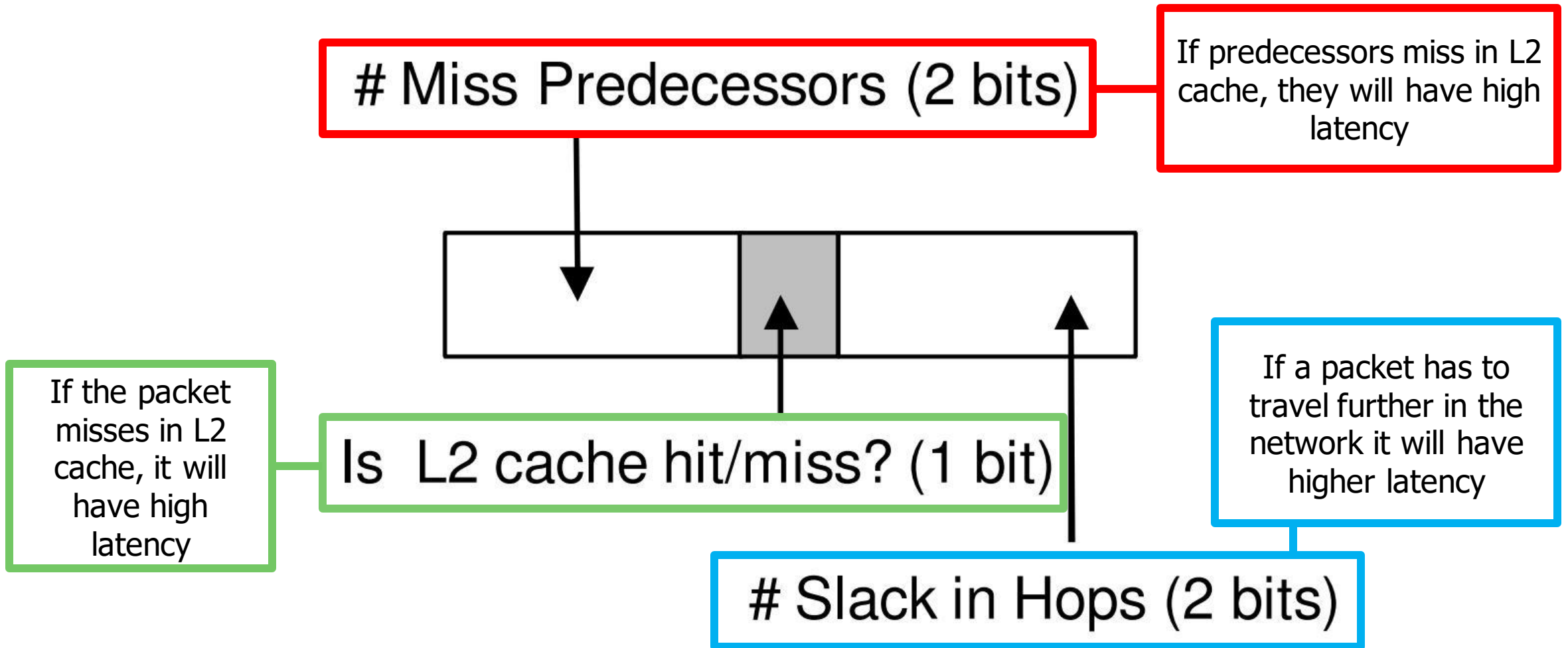
Mechanisms & Implementation

Estimating Slack

$$\text{Slack}(\text{Packet}_i) = \max_k \text{Latency}(\text{Packet}_k \forall k=0 \text{ to } \text{NumberofPredecessors}) - \text{Latency}(\text{Packet}_i)$$

Predicting packet latency is very difficult!

3 Causes of Latency

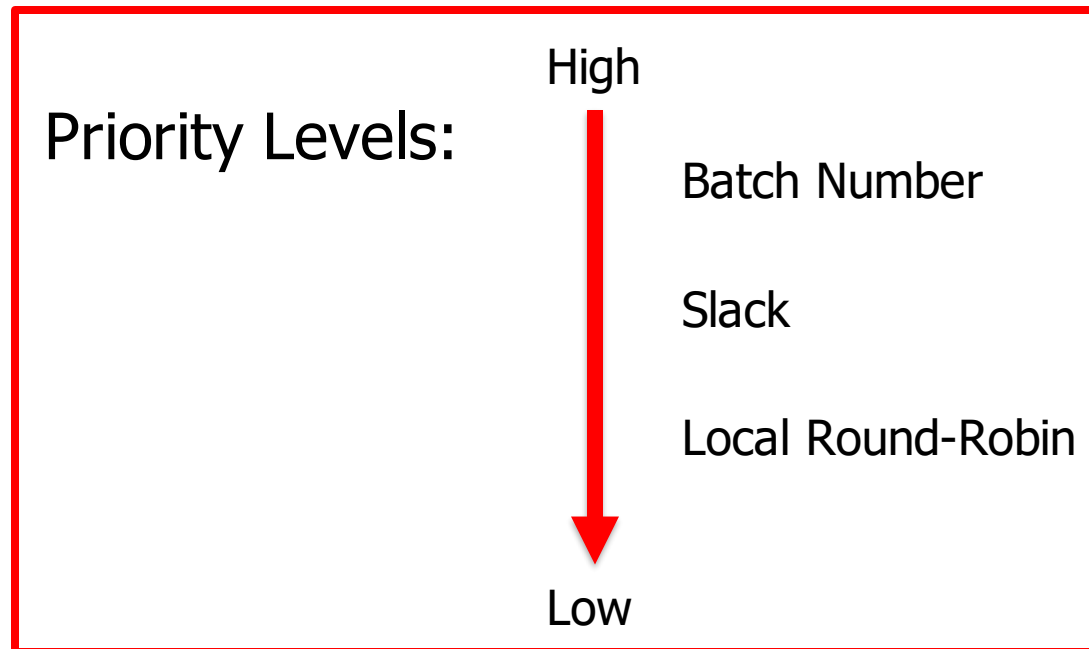
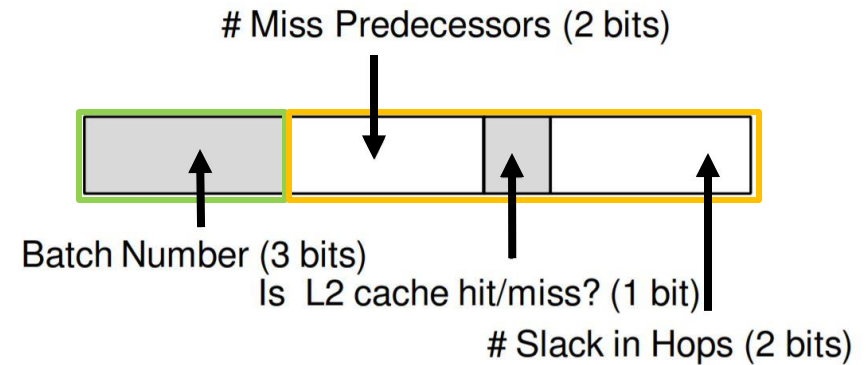


How to avoid Starvation? Batching!

- Problem: Slack-based prioritization might lead to starvation!
- We divide time in intervals of N cycles
- Packets inserted during the same interval are part of the same batch
- Packets of older batches are prioritized over packets from newer batches

Connecting the dots

Packet header with Aérgia priority structure:



Methodology & Major Results

Experimental Setup

- 64-core system
 - 2 GHz processor
 - 128-entry instruction window
 - 32KB private L1 and 1MB per core shared L2 caches
 - 4 GB DRAM, 4 on-chip DRAM controllers
- Network-on-Chip model
 - 8x8 mesh
 - Each node has a router, processor, private L1 cache and shared L2 cache bank
- 35 different applications, 56 different combinations

Experimental Setup

- Age (Baseline) "blind"
 - Treats all packets equally

- Application-Aware Prioritization Mechanism (STC) [Lee et al., ISCA 2008] "coarse-grained"
 - Prioritizes all packets of non-intensive application

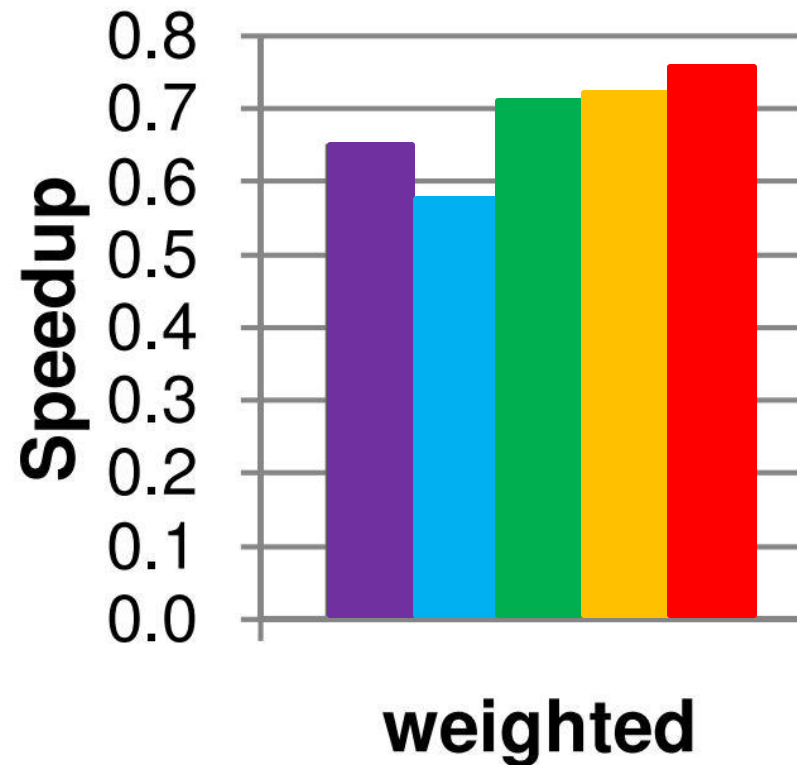
- Global Synchronized Frames (GSF) [Das et al., MICRO 2009] "coarse-grained"
 - Guarantees minimum bandwidth and network delay to all applications

Results: System Speedup

- STC: 8.9% improvement
- Aérgia: 10.3% improvement
- Aérgia+STC: 16.1% improvement

$$W. Speedup = \sum_i \frac{IPC_i^{shared}}{IPC_i^{alone}}$$

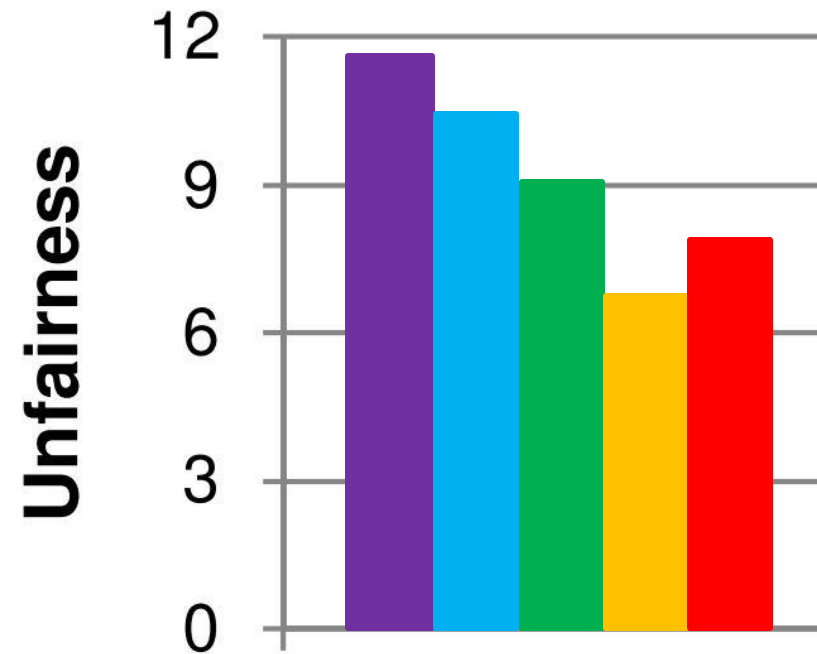
■ Base ■ GSF ■ STC ■ Aergia ■ STC+Aergia



Results: Network Unfairness

■ Base ■ GSF ■ STC ■ Aergia ■ STC+Aergia

- Aergia: ~1.5X improvement
- Aergia+STC: ~1.3X improvement



$$NetSlowdown_i = \frac{NST_i^{shared}}{NST_i^{alone}}, \quad Unfairness = \max_i NetSlowdown_i$$

Summary

Summary

- Problem: Treating all packets equally during scheduling will lead to performance loss
- Goal: Improving overall system performance by accelerating performance-critical packets
- Key Idea: Utilize packet slack to prioritize critical packets
- Key Mechanism: "Predicting" packet latency & prioritizing packets with low slack
- Results:
 - Overall system throughput improved by 10.3%
 - Network fairness improved by 30.8%

Strengths

Strenghts

- New approach to a problem that will likely become more significant over time
- Only slightly increases the header size
- Potential for further research
 - MemScale: Active Low-Power Modes for Main Memory, ASPLOS 2010
- Intuitive idea
- Well-written, well-structured and easy-to-understand paper

Weaknesses

Weaknesses

- Mechanism may not work effectively if workload utilizes small/no MLP
- (Does not consider the affect of different batch sizes)
- (Evaluation is done solely in simulation)
- (Are these the best workloads to evaluate?)

Takeaways

Key Takeaways

- Novel method to schedule packets on NoCs
- Simple idea
- Potential for further research
- Well written and easy-to-understand paper

Thoughts and Ideas

Thoughts & Ideas

- Can we improve Aéria?
 - "Network-on-Chip Packet Prioritisation based on Instantaneous Slack Awareness", INDIN 2015
- How can we utilize Aéria to protect us from malicious attacks such as Denial-of-Service attacks?
 - "Real-time Detection and Localization of DoS Attacks in NoC based SoCs", DATE 2019
- Can we use slack-based routing on bufferless On-Chip Networks?
 - "CHIPPER: A Low-complexity Bufferless Deflection Router", HPCA 2011

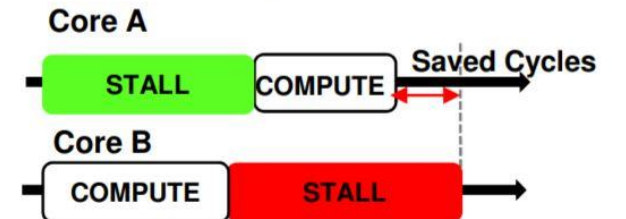
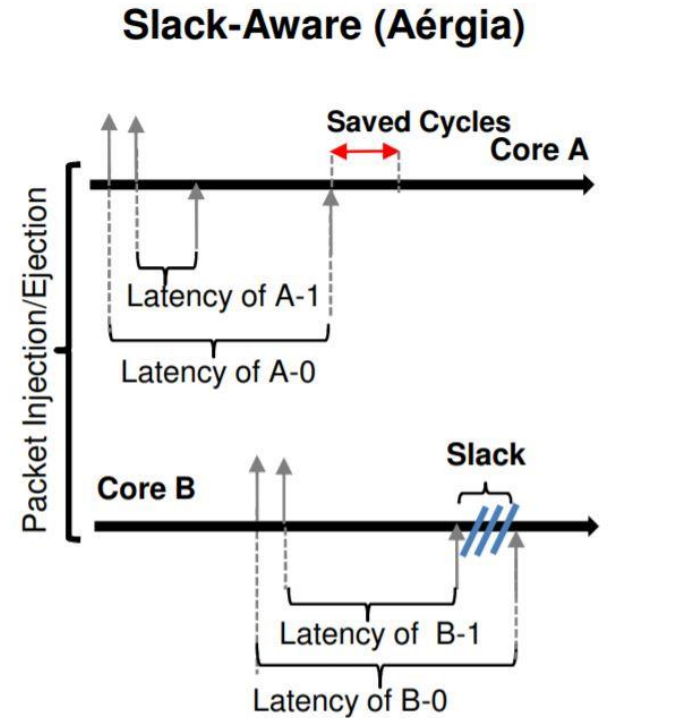
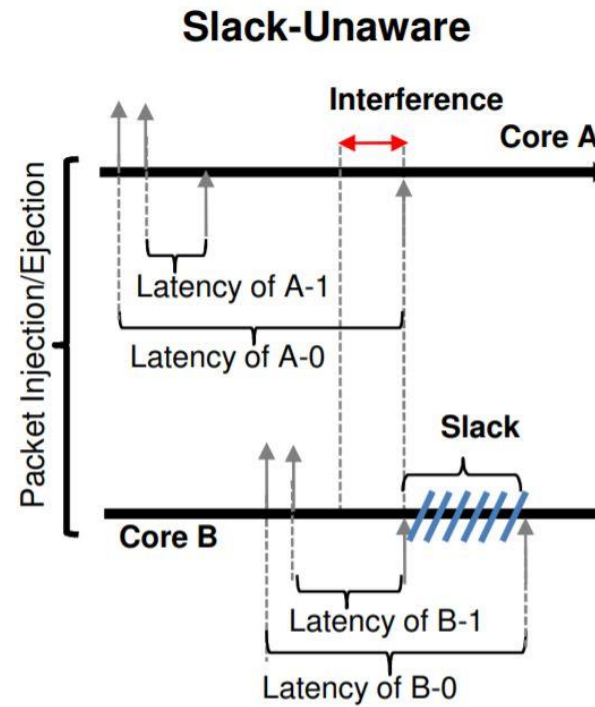
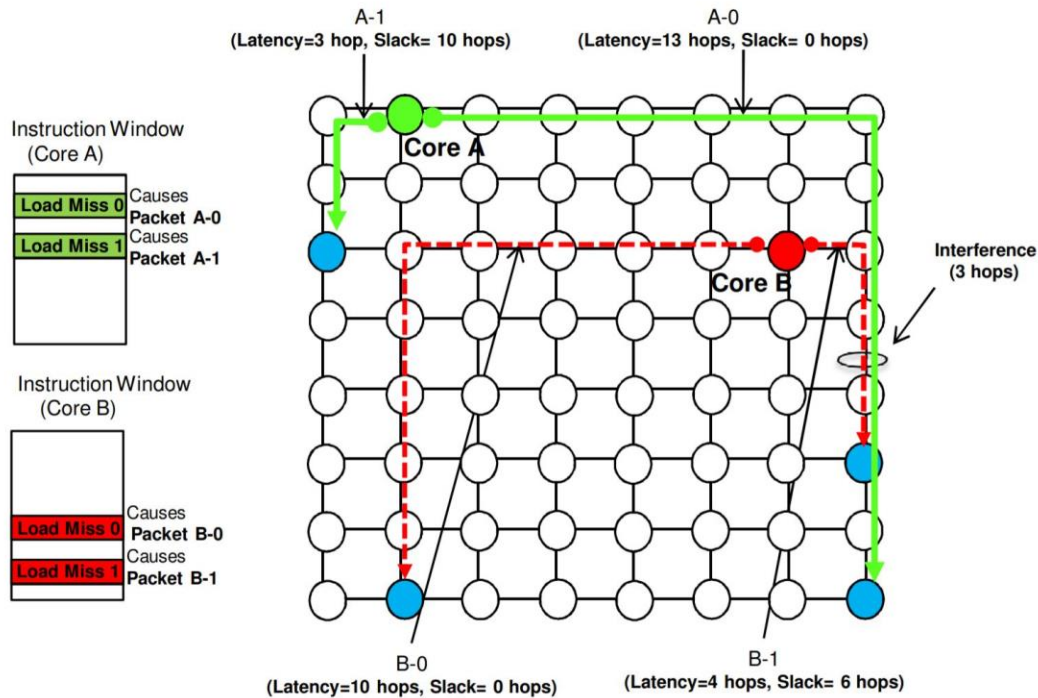
Discussion

Discussion Starters

- Thoughts on Aérgia?
- Thoughts on the previous ideas?
- Will the problem become more important in the future?
- Will Aérgia become more important in the future?

Backup Slides

Slack-Unaware vs Slack-Aware



Analysis of Miss Predictors

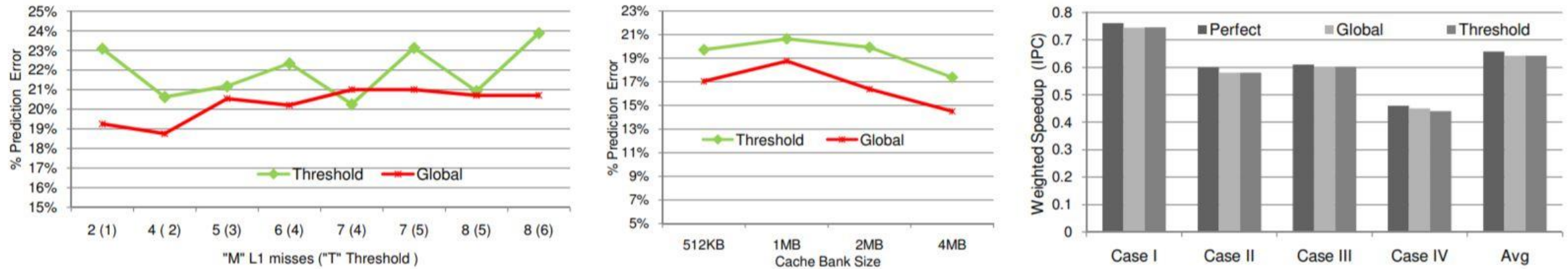


Figure 12: Analysis of L2 miss predictors: (a) error with parameters, (b) error with cache size, (c) effect of perfect prediction

L2 Hit/Miss Predictors

- How to predict L2 hit or miss at core?
 - *Global Branch Predictor* based L2 Miss Predictor
 - Use Pattern History Table and 2-bit saturating counters
 - *Threshold* based L2 Miss Predictor
 - If #L2 misses in “M” misses \geq “T” threshold then next load is a L2 miss.