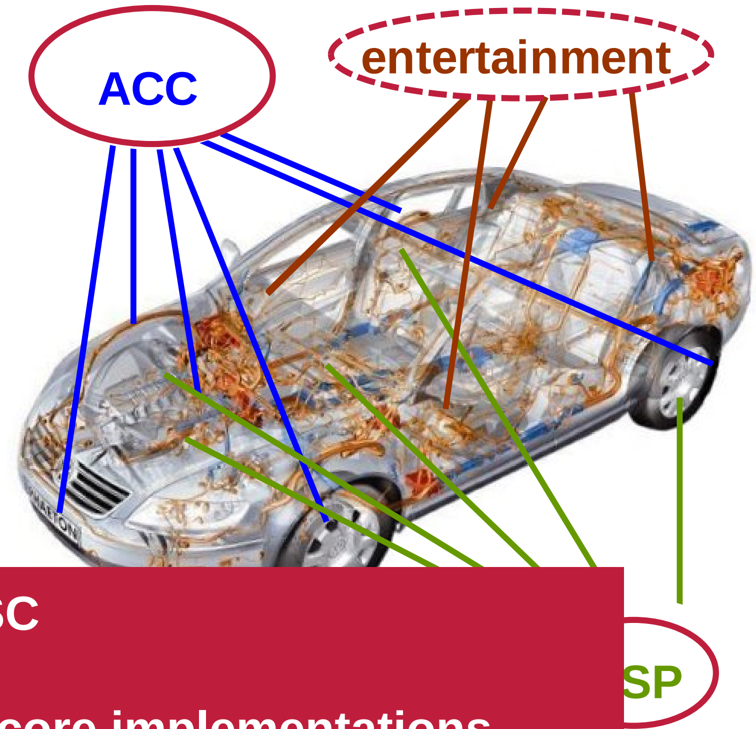
Technische Universität Braunschweig

Adaptive Load Distribution in Mixed-Critical Networks-On-Chip

Adam Kostrzewa, Sebastian Tobuschat, Leonardo Ecco and Rolf Ernst
{**kostrzewa**, tobuschat, ecco, ernst}@ida.ing.tu-bs.de
TU Braunschweig, Germany

Motivation

- today's many-core real-time systems are complex
  - **complexity increases**
  - **integrate** previously distributed functions
  - **implement** new functionality
- different time-related requirements
  - **safety-critical applications (SC)**
    - worst-case dimensioning
    - **deadline oriented**



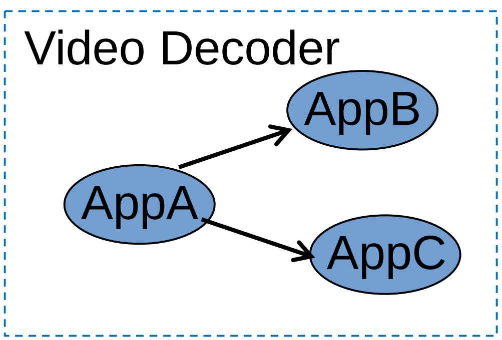
How to meet the simultanously BE & SC requirements in many-cores?

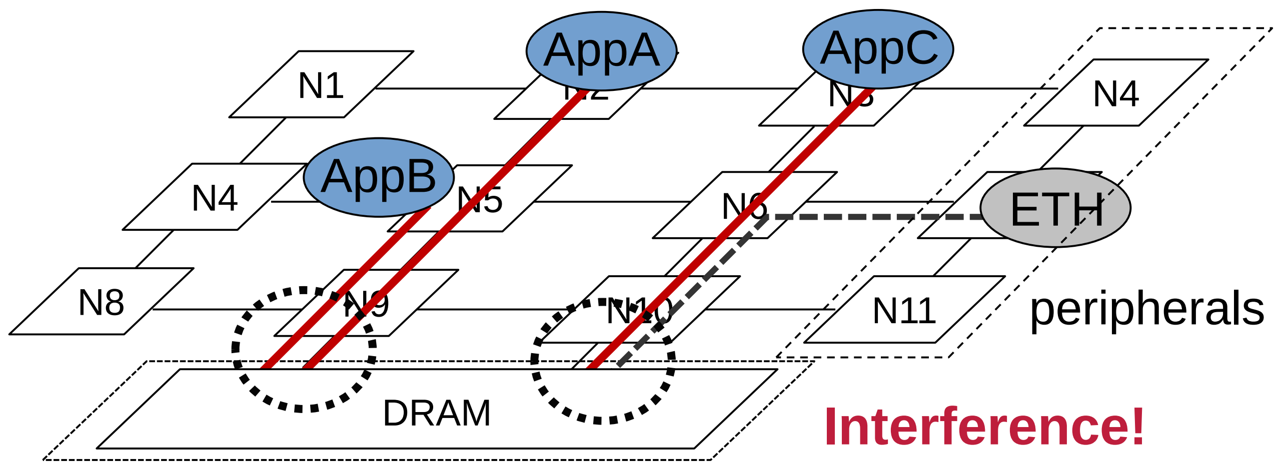
- **already challenging in current multicore implementations**

main difference: communication via Network-on-Chip (NoC)

Networks-on-Chip

- allow **integration** of many components
- **nodes are heterogeneous** e.g. processors, memory controllers, eth. controllers
- BE and SC transmissions **share NoC resources e.g.** links and buffers
- safety **requires separation** in case of shared resources
  - **functional independence -** still allow application communication
  - **timing independence** – still allow efficient scheduling





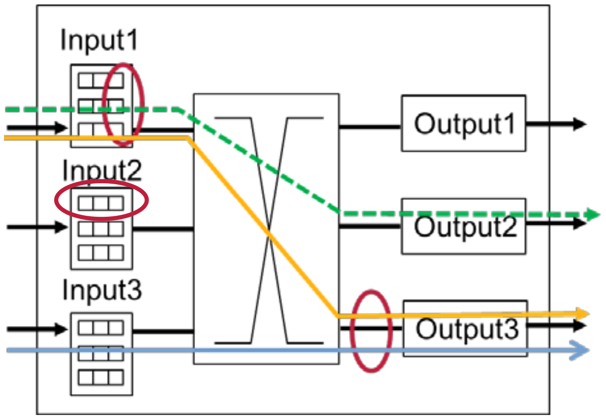
Main Challenge → QoS guarantees + high performance

Overview

- Motivation
- **Existing solutions – Mixed-Critical NoCs**
- **Adaptive Load Distribution**
- **Predictability**
- **Evaluation**
- **Conclusion**

Standard NoCs

- standard NoCs concentrate on best-effort applications
  - main trade-off between total buffer size (buffering strategy)
  - and link utilization (bandwidth allocation)



Advantages

  - high average performance for BE senders
  - relatively simple implementation - low costs

Disadvantages (safety)

- complex **multistage arbitration**
  - FIFO in buffers, two-staged iSLIP for output ports
- distributed and **local arbitration**
  - h
- BE

Problems:
hardly analyzable i.e. high analysis complexity
no or pessimistic guarantees for SC senders

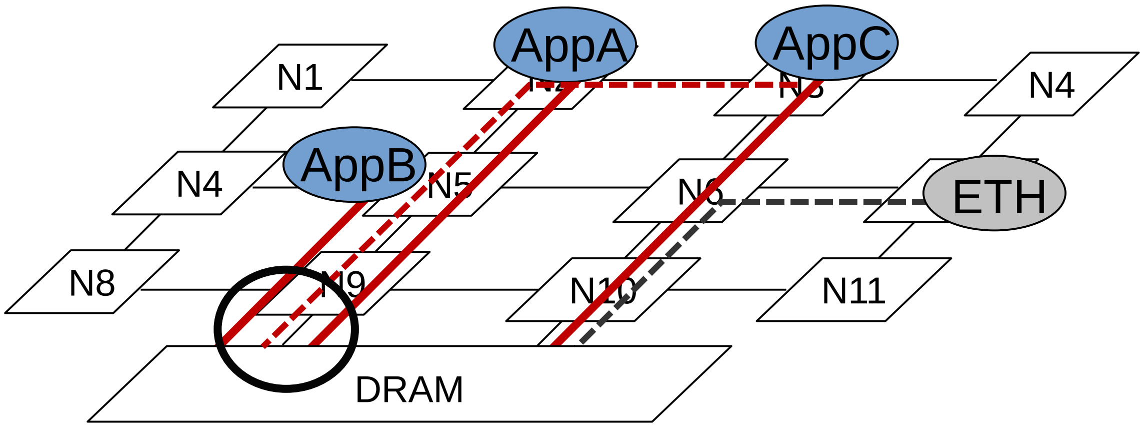
- applies mapping to distribute and isolate mixed-critical senders in space
- every SC stream with a dedicated set of resources i.e. links and buffers

Advantages

- full isolation of BE senders
- guarantees for SC senders

Disadvantages

- hardware overprovisioning
  - large chip area
  - high power consumption
- may be impossible
  - commercially available MPSoCs have limitations
    - e.g. number of ports, ETH interfaces etc.



Interference!

QoS in NoCs – Temporal Isolation

- **paths** for senders are **static during runtime**
- **links and buffers** as **resources shared in time**
  - **static isolation** - service independent from other senders
    - e.g. **TDM** time-division multiple-access
      (AEthereal [Goossens], PhaseNoC[Psarras], SurfNoC[Wassel])
    - transmissions access NoC in a predefined cyclic order

s [Lee])
Es
d

Problems:

Resign from multidimensional nature of the NoC
Lack of a load balancing during runtime !

BE senders blocked for the full duration of SC transmissions!

→ Decreased utilization or no safety!

Overview

- Motivation
- Existing solutions – Mixed-Critical NoCs
- **Adaptive Load Distribution**
- **Predictability**
- **Evaluation**
- **Conclusion**

Our Goal

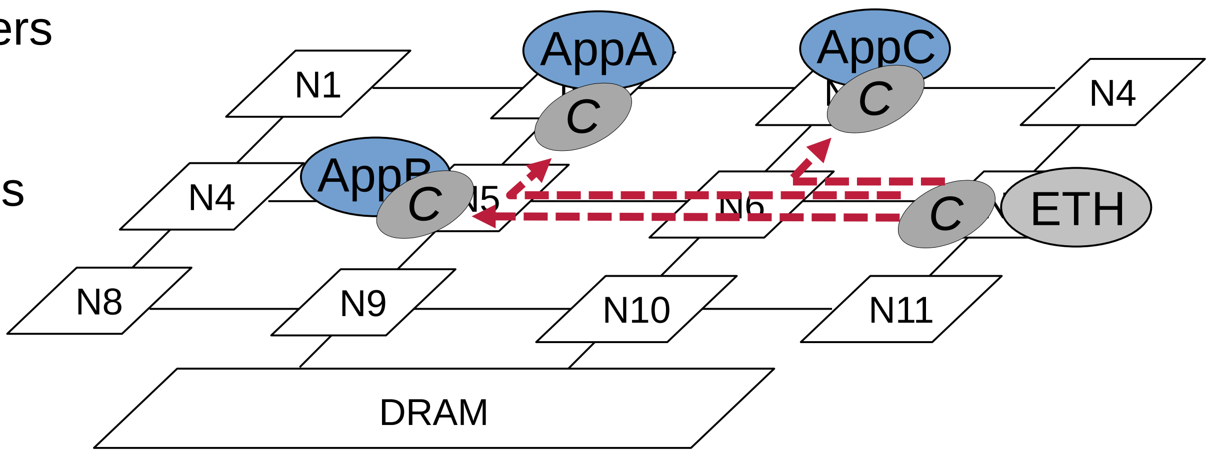
- **different path allocation methods depending on sender's criticality**
  - **SC traffic – static path allocation**
  - **BE traffic – multiple paths from source to destination**
- **dynamically distribute transmissions over the set of paths**
  - **adaptive QoS -** based on the global state of the system **at runtime**
  - **release resources reserved for SC for BE traffic whenever not used**
- **high performance for BE and improved guarantees for SC**
  - increases hardware utilization – re-using of links and buffers
  - permits safe **sharing of VCs** by tasks with different QoS requirements
- **low hardware overhead**
  - no need for router modifications, re-using existing components
  - app

Is it possible?
How to achieve this?

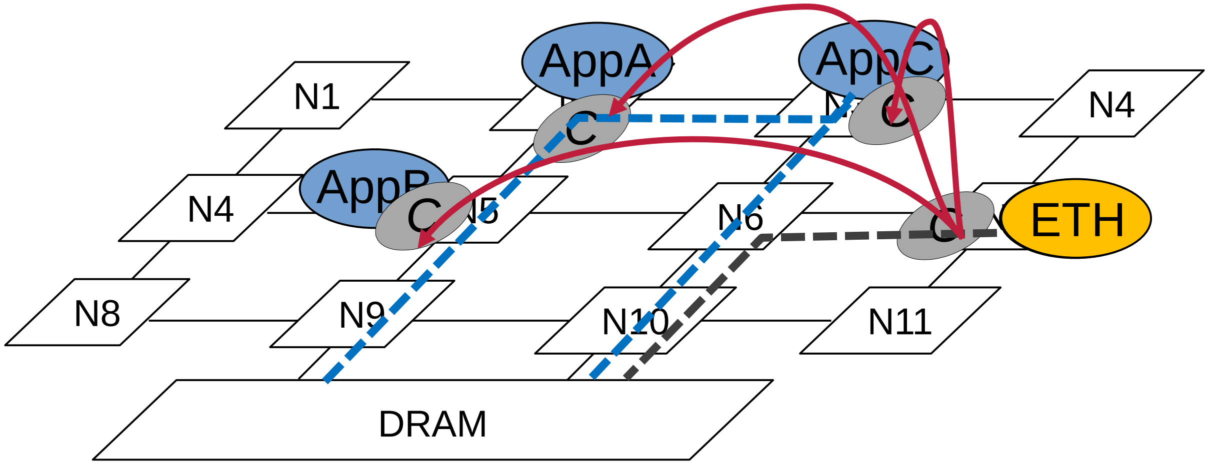
Overlay network

- **overlay network to decouple data flow and admission control**
- **data layer – data transport and data routing and arbitration**
- **control layer** – **global and dynamic arbitration**
  - **clients (C)** - admission control locally in nodes
  - protocol based **synchronization of SC and BEs**

- **Broadcast propagate the global NoC-state**
  - currently active SC senders
  - used by them resources
- adjust (block/unblock) paths available for BE senders



Protocol



- **BE may use the resources for SCs – if SCs are not active**
- BE must release the resources whenver SC is activated
  - safety must be protected by clients
  - switch latency must be formally guaranteed
- **Safety assured by clients**

Properties

- **adaptive scheduling** between SC and BE traffic
  - resource arbitration based on the global state of the NoC
  - **re-using links** whenever possible
  - **detouring** senders instead of blocking
- exclusive access to the NoC for SC senders
  - **reduced blocking** and **decreased size of buffers** in routers
  - permits **mixed-critical setups**

Implementation

Bottom Layer

- **work-conserving** arbitration in routers e.g. round-robin, iSLIP
- **predictable** behavior of **routers**
  - analyzable with one of the existing analysis

Safe communication for control messages

- **control NoC** for maximum efficiency e.g. D-NoC &C-NoC in MPPA, Tile64
- **dedicated VC** capable of giving latency guarantees
  - e.g. traffic shaping, priority based scheduling for VCs

Clients and RM

- **Hardware** e.g. clients as extension of NI, RM independent unit or in “hotspot”
- **Software**, similarly to Software Defined Networks
- **Combination** of both depending on infrastructure

Overview

- Motivation
- Existing solutions – Mixed-Critical NoCs
- Adaptive Load Distribution
- Predictability
- Evaluation
- Conclusion

Predictability

- mechanism description → mathematical model
- calculate the worst-case response-time
- incl. end-to-end latency for transmissions – corner cases
- validate against the deadlines

- ***busy window*** approach
- transmissions abstracted with ***event models***
  - $\boldsymbol{\eta}^{+}(\Delta \boldsymbol{t})$, $\boldsymbol{\eta}^{-}(\Delta \boldsymbol{t})$ maximum and minimum number of initiated transmissions during time period $\boldsymbol{\Delta t}$
  - framework: **Compositional Performance Analysis (CPA)**
- **only overview - details in the paper**

Predictability

- **the worst-case time** necessary to conduct ***q SC*** **transmissions** on the **particular path h ($w^{+}{}_{i}(q, h)$)**

$$w^{+}{}_{i}(q,h) = \underbrace{q * C^{+}{}_{i}(h)}_{\text{duration of q trans.}} + \underbrace{B_i\left(w^{+}{}_{i}(q,h)\right)}_{\text{the maximum blocking resulting from other synchronized transm.}}$$

Analysis of blocking for different scheduling setups:
"Dynamic Control for Mixed-Critical Networks-On-Chip" Kostrzewa et al. RTSS 2015
Or "Dynamic Admission Control for Real-Time Networks-On-Chips" Kostrzewa et. al. ASP-DAC 2016

Including complex synchronization protocols!

Slack Constrains

BE Overhead – latency which q SC transmissions may experience from BE traffic on a path h

- $$O_i(q,h) = q * \Big(\underbrace{\max_{\forall j \epsilon BE(h)} \mathbf{C}^{+}_{j,ctrl}}_{\text{maximum latency of blocking message}} + \underbrace{\max_{\forall j \epsilon BE(h)} \mathbf{C}^{+}_{j,pckt}}_{\text{maximum latency of the last BE packet}} \Big)$$

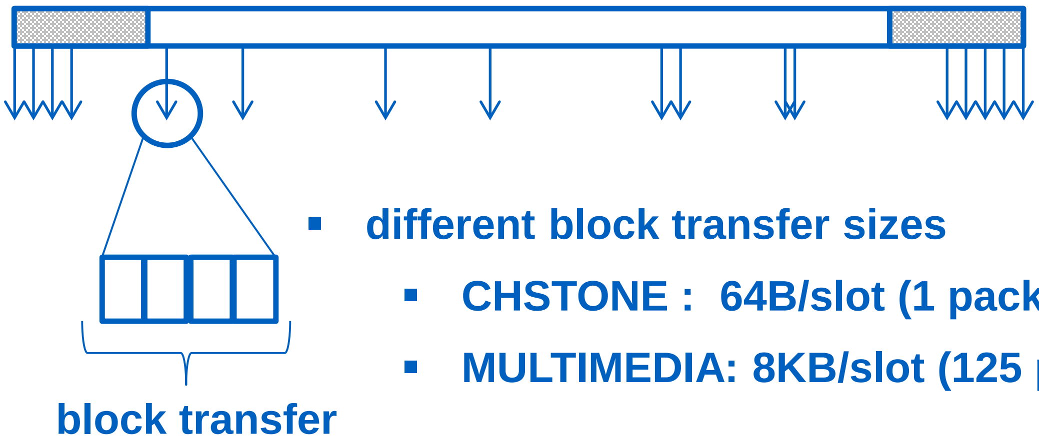
- **SCHEDULABILITY CONDITION**
  - $\forall i \; \epsilon \; SC_i(q) \; \geq \; O_i(q,h)$
  - the worst-case slack $SC_i(q) = D_i(q) - \mathrm{R}_i(q)$
  - each SC sender must have enough slack to cover the detouring overhead
  - otherwise BE can not share paths with SC

Overview

- Motivation
- Existing solutions – Mixed-Critical NoCs
- Adaptive Load Distribution
- Predictability
- **Evaluation**
- **Conclusion**

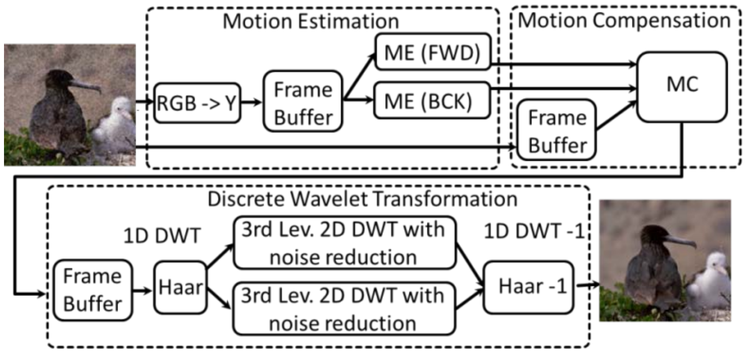
Experiments

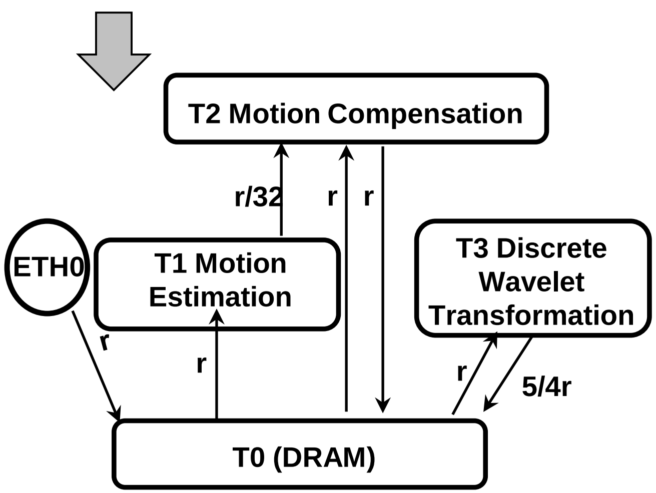
- **analytical experiments**
  - Compositional Performance Analysis (CPA) framework
  - pyCPA python library
- **simulations**
  - OMNeT++ event-based simulation framework
  - HNOCS library
- **input data**
  - memory access traces
  - multimedia modules with block transfers

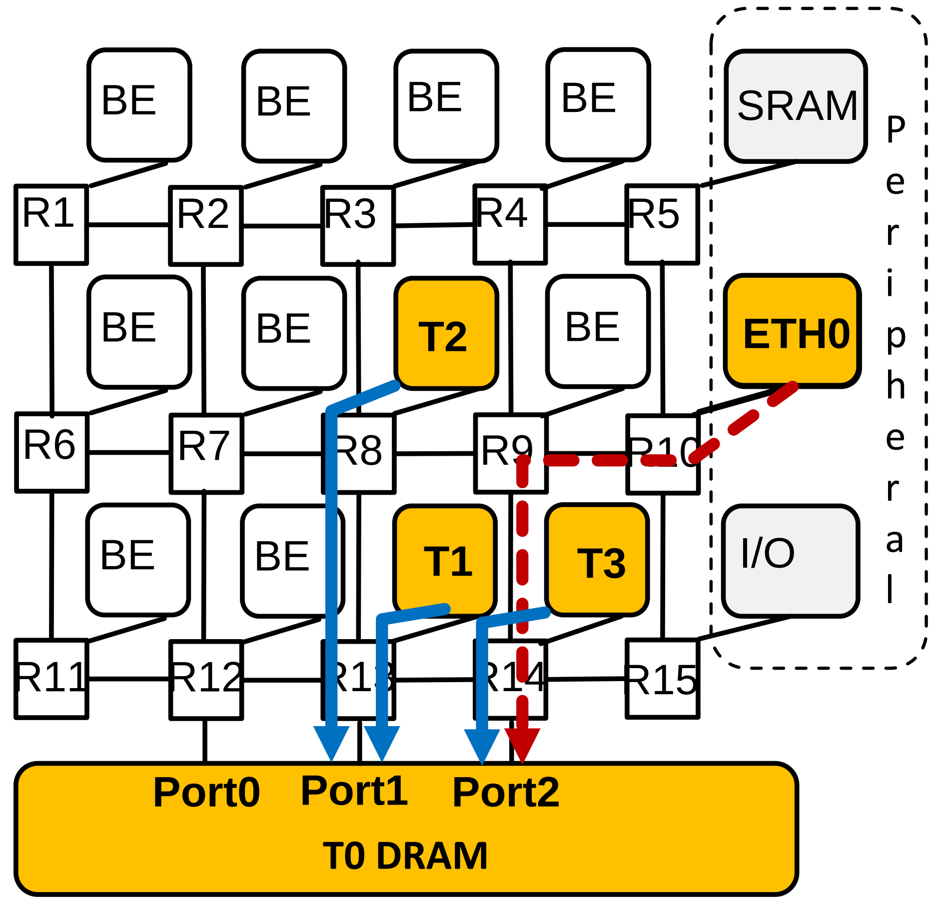


- **different block transfer sizes**
  - **CHSTONE : 64B/slot (1 packet)**
  - **MULTIMEDIA: 8KB/slot (125 packets)**

Use-case: Real-Time Video Denoising

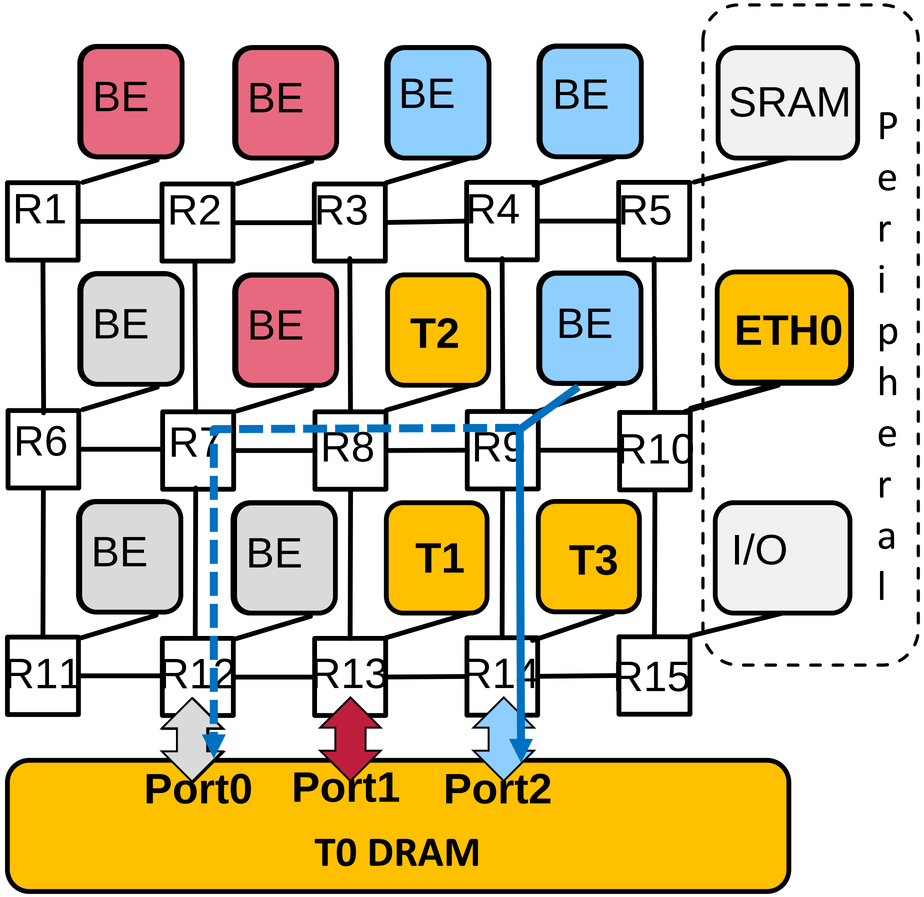




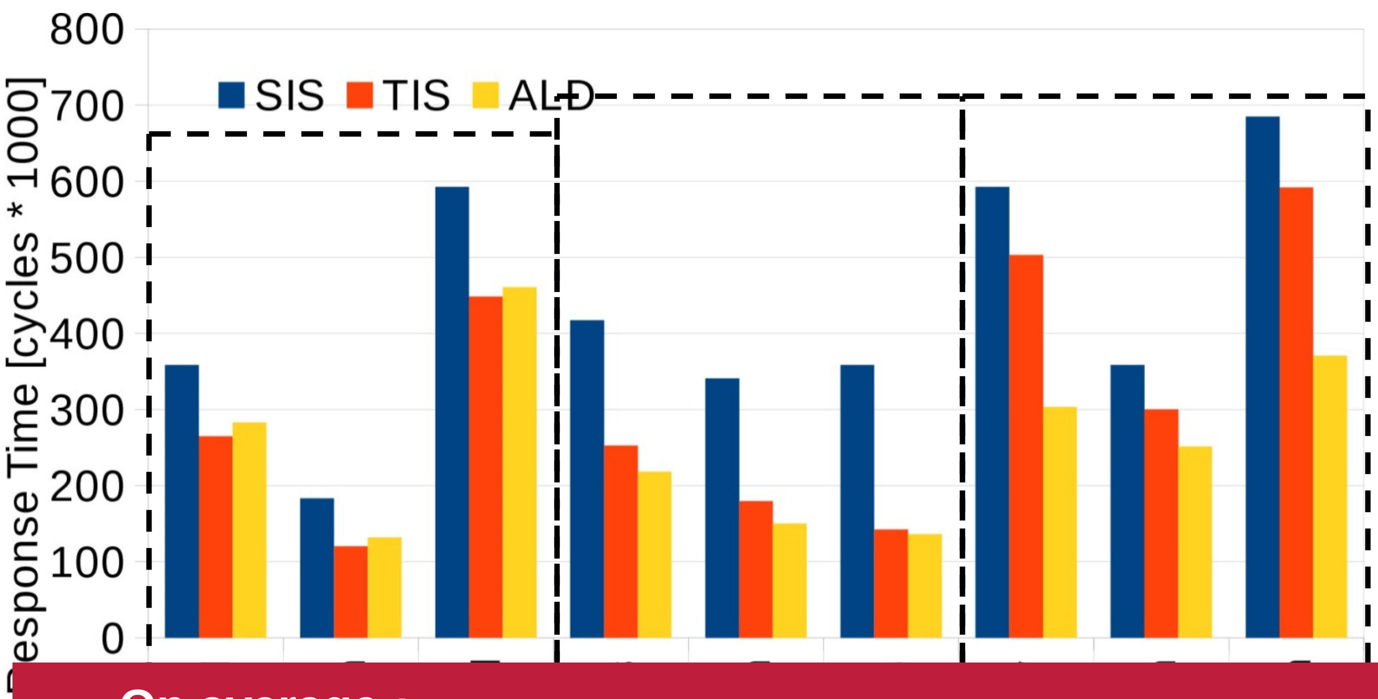


Experiments

- **spatial isolation (SIS)**
  - all BE senders must use Port0
  - no link shared between BE and SC
- **temporal isolation (TIS)**
  - priority assignments for VCs
  - distribute the load between ports
  - BE blocked when SC are active
- **adaptive load distribution (ALD)**
  - each BE has a detoured path to Port0
  - when SC sender is active load is detoured



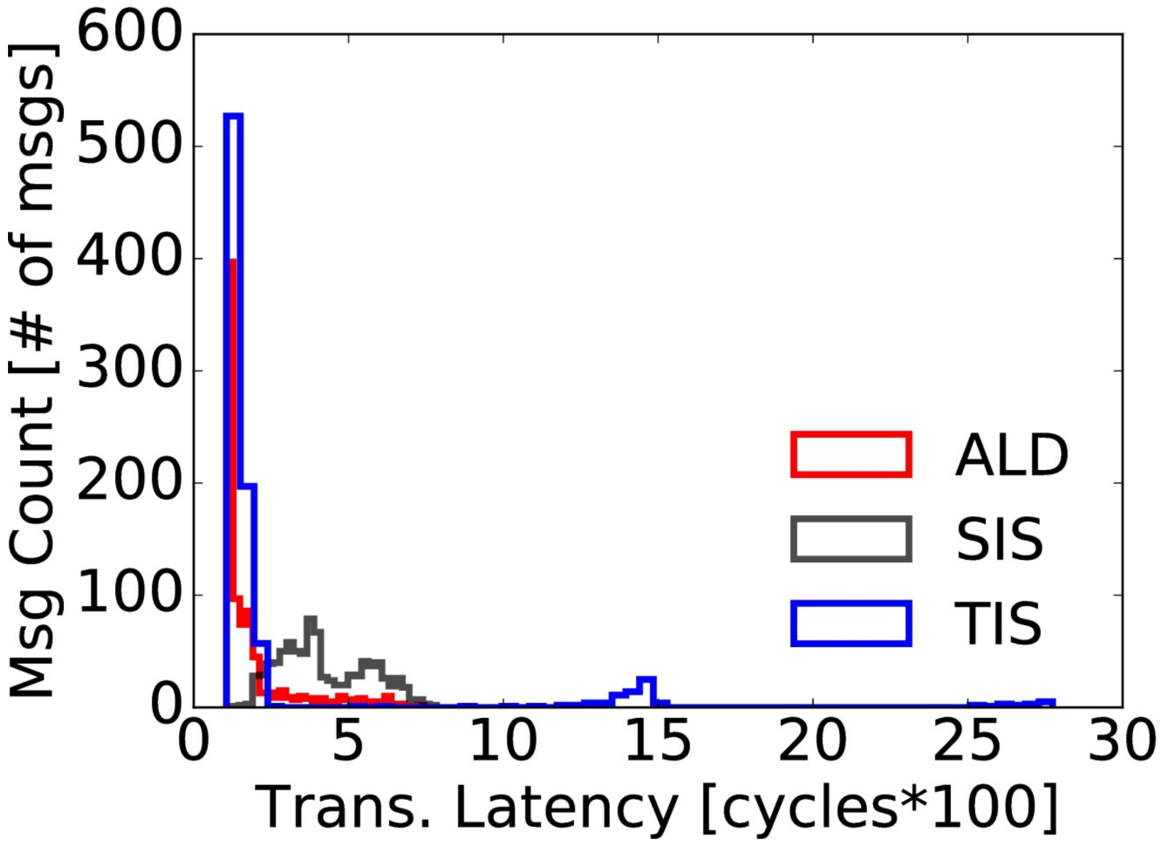
Benchmark-Based Results



On average :

- **~ 50% improvement in comparison with SIS**
- **~ 35% improvement in comparison with TIS**

Transmission Latencies



Histogram of transmission latencies for adpcm benchmark.

Overview

- Motivation
- Existing solutions – Mixed-Critical NoCs
- Adaptive Load Distribution
- Predictability
- Evaluation
- **Conclusion**

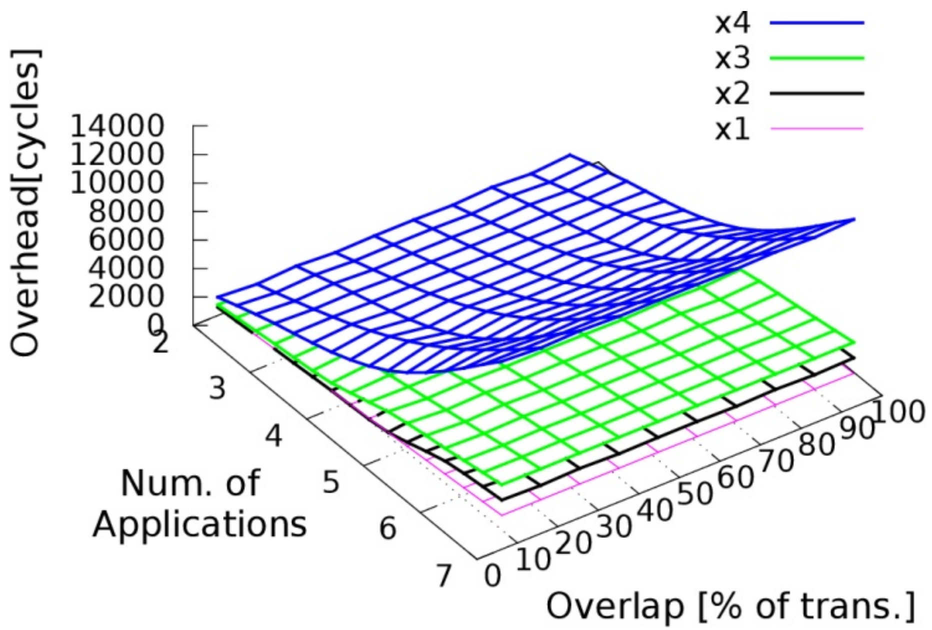
Conclusions

- new method for **safe sharing of resources in NoCs**
  - **arbitration in space and time**
- global and dynamic scheduling
  - **safe and efficient guarantees for SC senders**
  - proved through the formal worst-case analysis
  - significantly **improved BE performance**
- low-hardware overhead
  - **no modifications of routers**
  - possibility of software implementation
  - **applicable in majority of existing NoCs**

Thank you for your attention!
Questions?

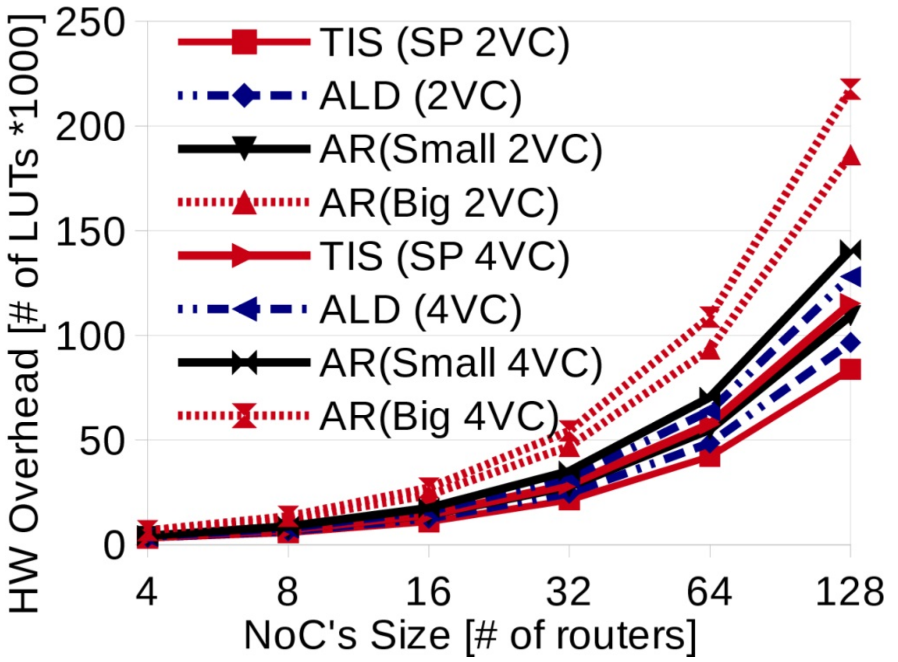
Backup Slides

Control Messages



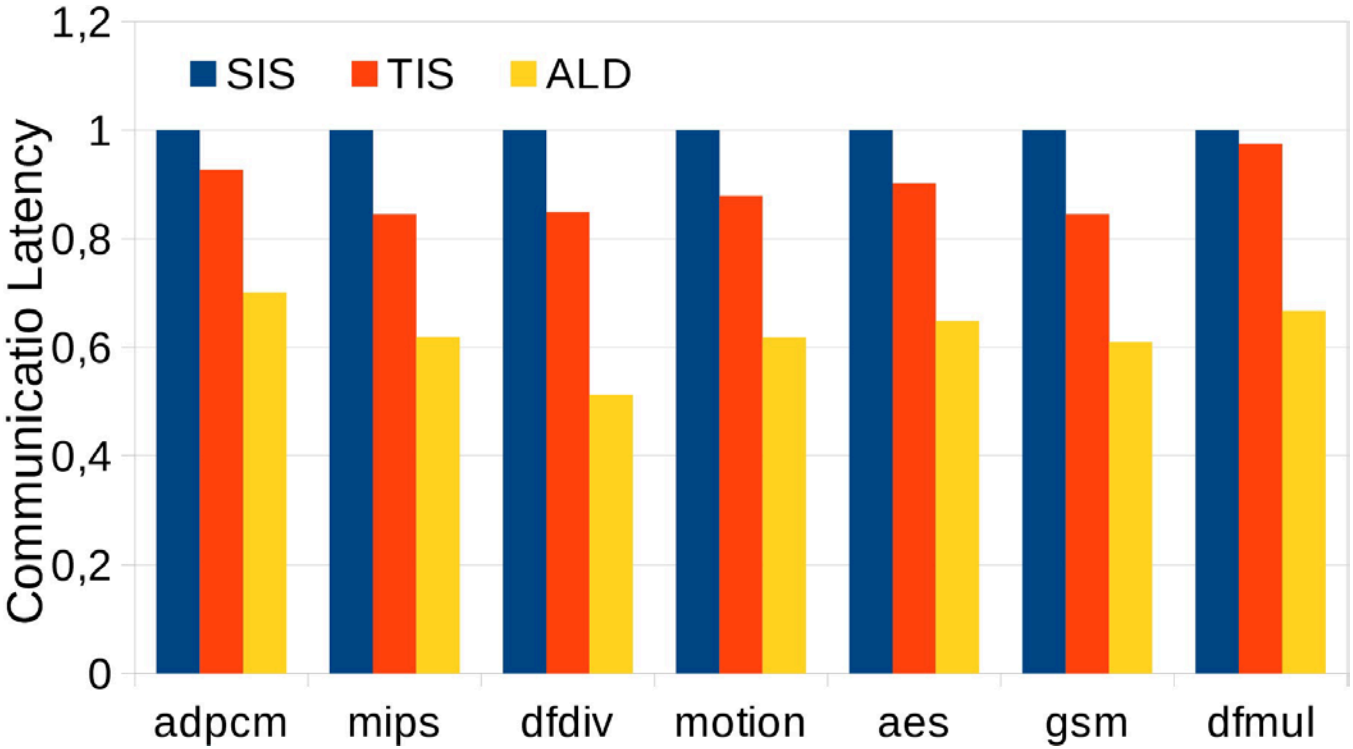
- proportional to the number of synchronized senders
- constant w.r.t transmission length
  - longer transmissions e.g. DMA transfers

Control Layer Overhead



Hardware (area) overhead resulting from synchronization in NoC.

Improvement From Adaptive Load Distribution



Improvement for different BE benchmarks (normalized to SIC).