Blocking and Non-blocking Checkpointing and Rollback Recovery for Networks-on-Chip

Claudia Rusu[1], Cristian Grecu[2], Lorena Anghel[1]

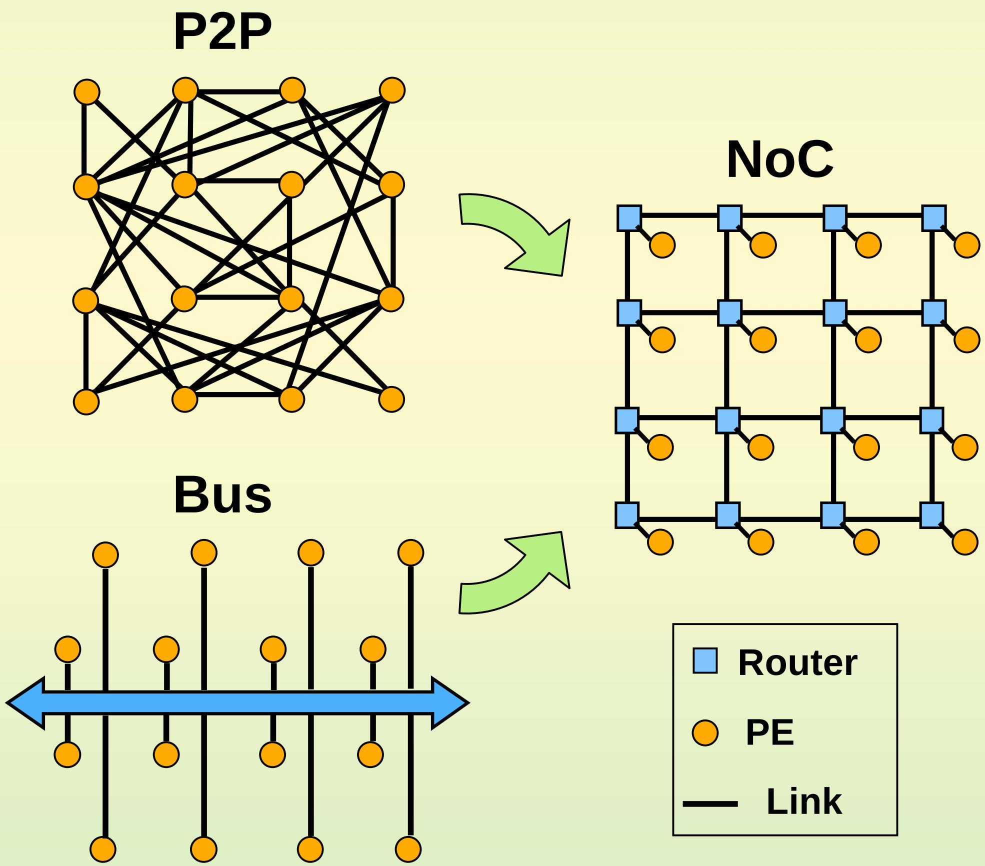
[1] TIMA Laboratory, CNRS-UJF-INP, Grenoble, France
[2] SoC Laboratory, University of British Columbia, Vancouver, Canada

OUTLINE

- Introduction
  - Networks-on-Chip
  - Checkpoint and rollback recovery
- Coordinated checkpointing
- Blocking and non-blocking coordinated checkpointing
- Case study
- Conclusions and future work

Network-on-Chip based Systems

- NoC vs. traditional connection systems



- NoC advantages
  - Efficient sharing of wires
  - Shorter design time, lower effort
  - Scalability

NoC QoS vs. Faults

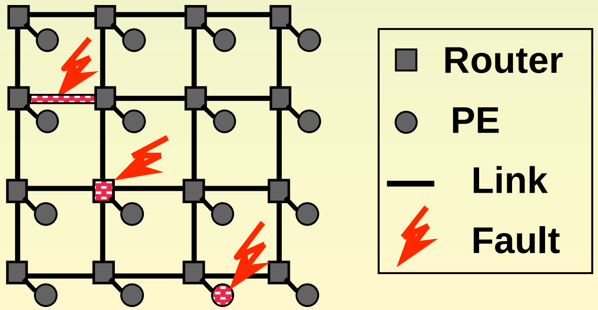
- Quality of service (QoS)
  - reliability, throughput, latency, bandwidth
- Unreliable signal transmission medium
  - timing and data errors
  - process variation, crosstalk, electromagnetic interference, radiations

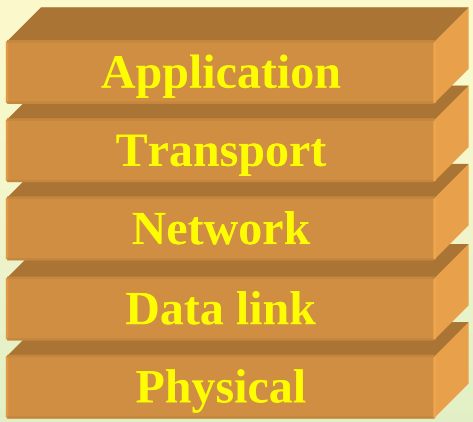
- Technology down scaling
- Increased system complexity

=> Increased vulnerability to faults

Fault Tolerance in Networks-on-Chip

- **Faults and Fault Tolerance**
  - At different NoC components
    - Links
    - Routers
      - switching blocks
      - memories
  - At different levels of the communication protocol stack





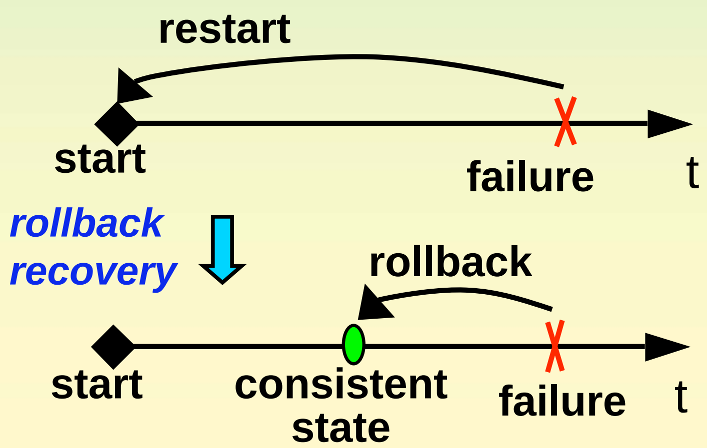
- **Fault tolerant solutions**
  - adaptive routing
  - stochastic communication
  - EDC, ECC, NMR

OUTLINE

- Introduction
  - Networks-on-Chip
  - Checkpoint and rollback recovery
- Coordinated checkpointing
- Blocking and non-blocking coordinated checkpointing
- Case study
- Conclusions and future work

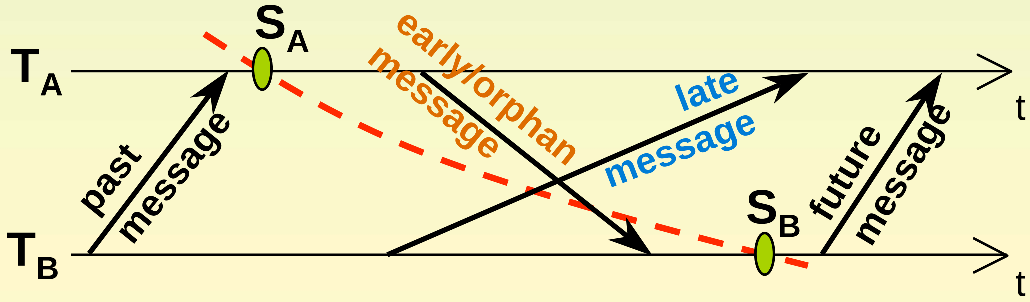
Checkpoint and Rollback Recovery. Principle

- **No failure tolerance**
  - **Failure => Restart**
- **Checkpoint and rollback recovery**
  - **Failure => Resume from a more recent state**
  - **Principle**
    - **Failure-free**
      - **periodically store states on stable storage**
    - **Failure**
      - **rollback to the last consistent stored state**

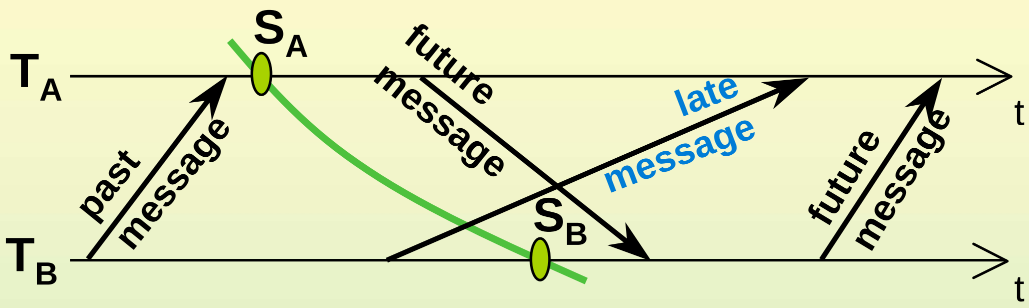


Checkpoint and Rollback Recovery. Consistent State

- **Message types vs. recovery line**



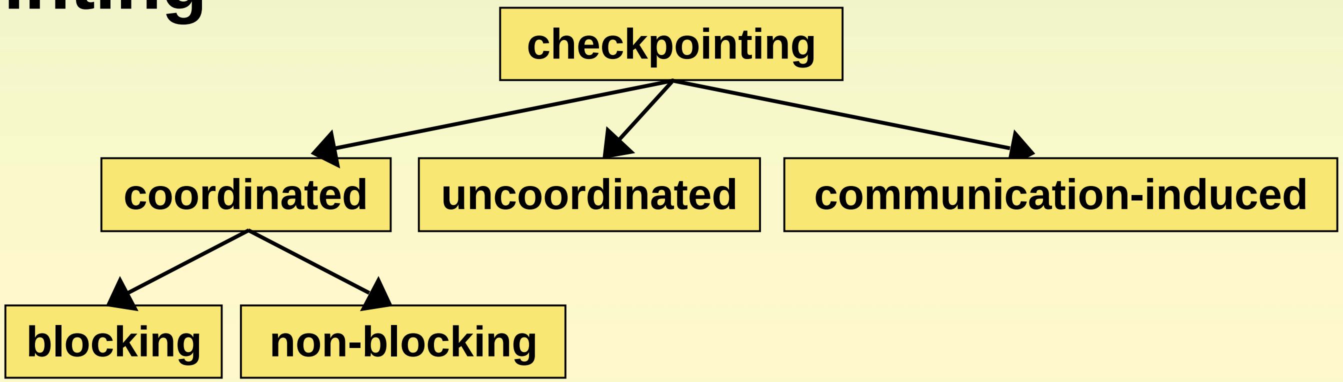
- **Consistent state with late messages**



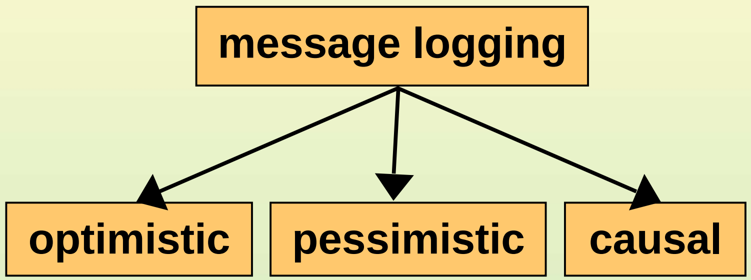
  - **early messages are avoided**
  - **late messages are to be replayed after rollback**

Checkpoint and Rollback Recovery. Classification

- Checkpointing



- Message logging

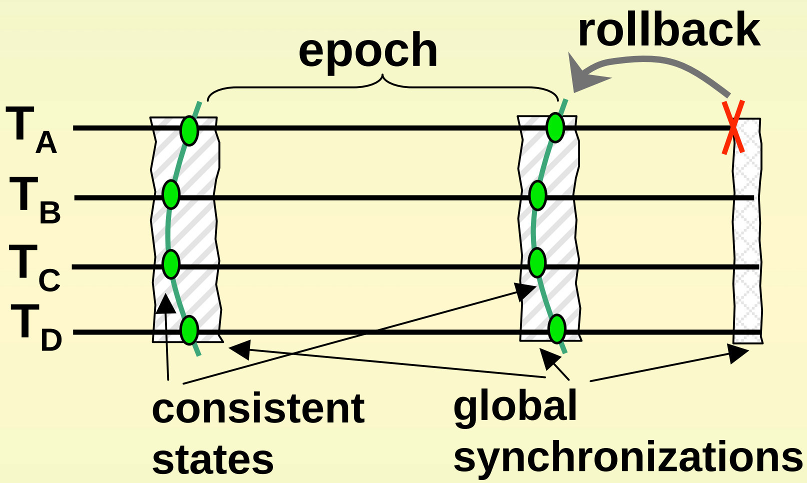


OUTLINE

- Introduction
  - Networks-on-Chip
  - Checkpoint and rollback recovery
- Coordinated checkpointing
- Blocking and non-blocking coordinated checkpointing
- Case study
- Conclusions and future work

Coordinated Checkpointing

- **Principle**



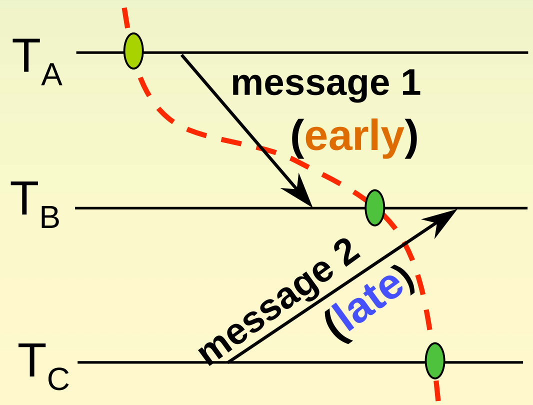
- **Failure-free**
  - synchronization –> consistent state
- **Failure**
  - rollback to the last consistent state

- **Task checkpoint**
  - task state
  - list of late messages
- **Late messages log**
  - optimistic approach
    -> small latency on failure-free
  - logged at receiver
    -> small recovery overhead
- **Unique coordinator**
  - reduced overhead
- **Unique blocking and non-blocking protocol**
  - allows for the same checkpoint the blocking of a task set and the non-blocking of another

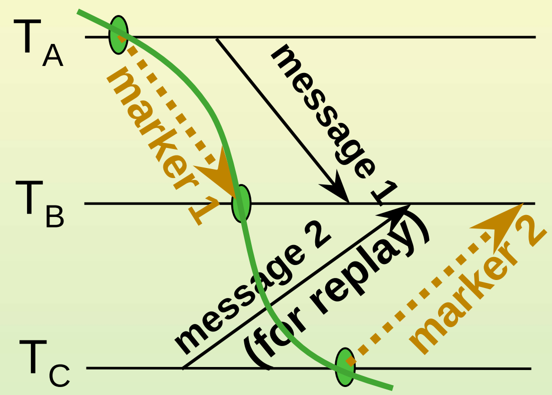
Synchronization. Markers

- **Markers**
  - are used to
    - avoid early messages
    - identify late messages and to end the log of late messages
  - dedicated messages (avoid long checkpointing durations when communication among certain tasks is scarce)
- **A task has taken the checkpoint only after state and late messages form other tasks are on stable storage**

Inconsistent state



Consistent state using markers

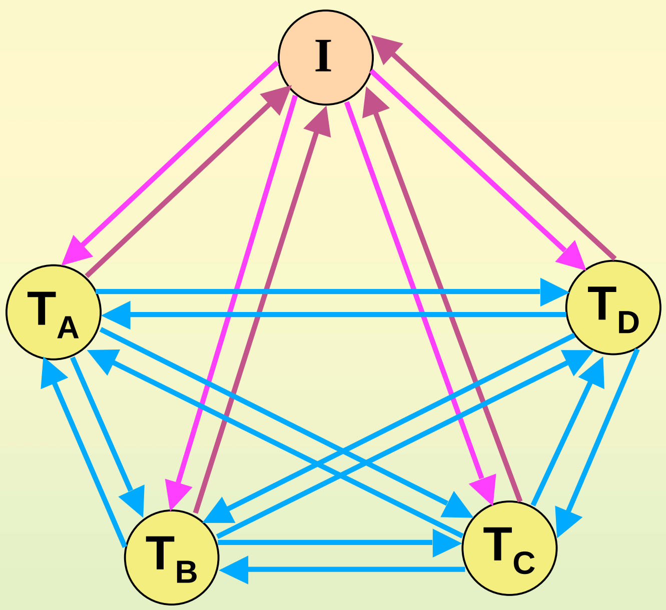


OUTLINE

- Introduction
  - Networks-on-Chip
  - Checkpoint and rollback recovery
- Coordinated checkpointing
- Blocking and non-blocking coordinated checkpointing
- Case study
- Conclusions and future work

Blocking and Non-blocking Coordinated Checkpointing Protocol

- **Synchronization messages**



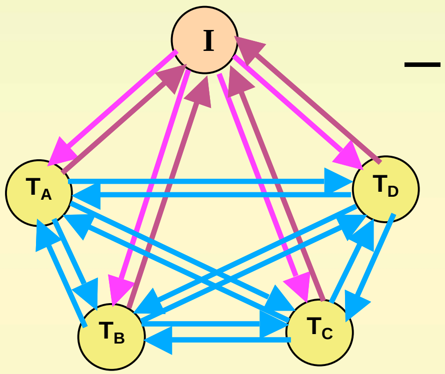
- **Checkpointing protocol**

Initiator	Non-initiator (blocking or not)

- broadcast CK_REQ
 	
- when CK_TAKEN received from all tasks
  - validate global checkpoint
	- on CK_REQ receipt
	  - broadcast CK_START
	  - when CK_START received from all tasks
	    - take local checkpoint
	    - send to initiator CK_TAKEN

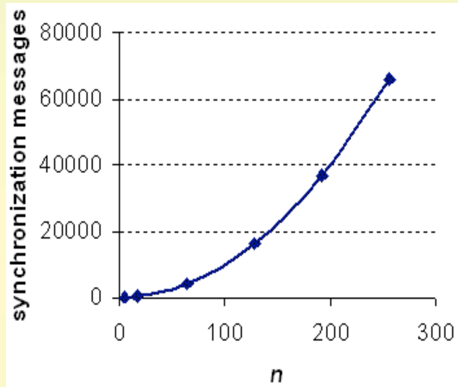
Blocking and Non-blocking Overhead

- **Synchronization messages**



– *n* nodes
  - CK_REQ n
  - CK_START $n*(n-1)$
  - CK_TAKEN n

$\Big\}\ O(n^2)$



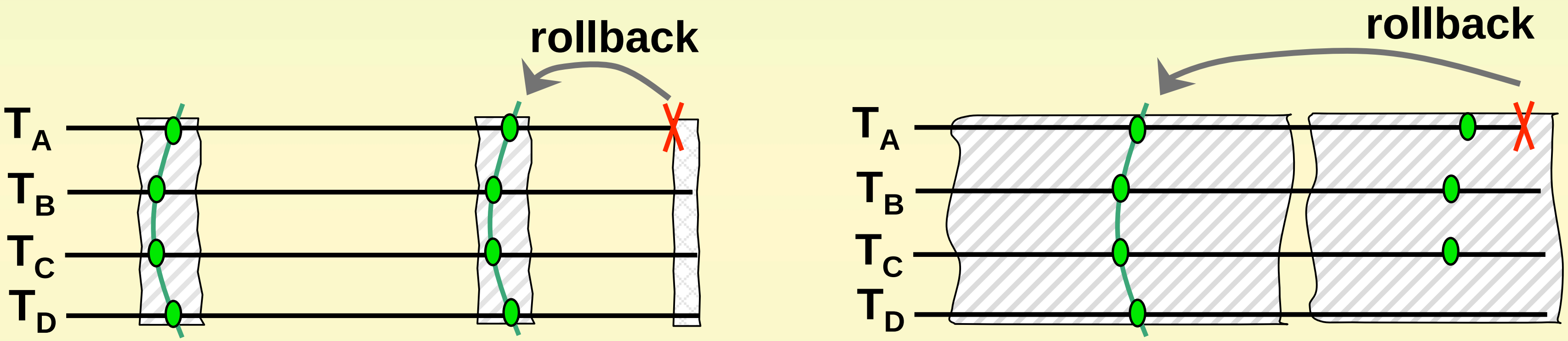
- **Messages in NoC during checkpointing**

❖ **Blocking**
  – synchronization messages

❖ **Non-blocking**
  – synchronization messages
  – application messages

Checkpointing Duration

- **High overhead during checkpointing**
  –> checkpointing phase reduced



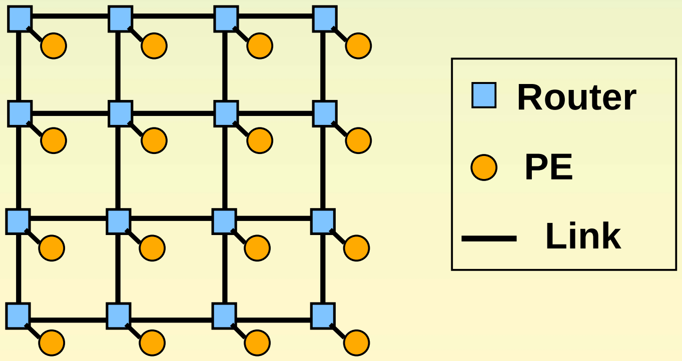
- **Long checkpointing durations**
  –> reduced number of checkpoints
- **When failure rate is comparable with checkpointing duration**
  -> rollbacks to the same old checkpoint

OUTLINE

- Introduction
  - Networks-on-Chip
  - Checkpoint and rollback recovery
- Coordinated checkpointing
- Blocking and non-blocking coordinated checkpointing
- Case study
- Conclusions and future work

Case Study

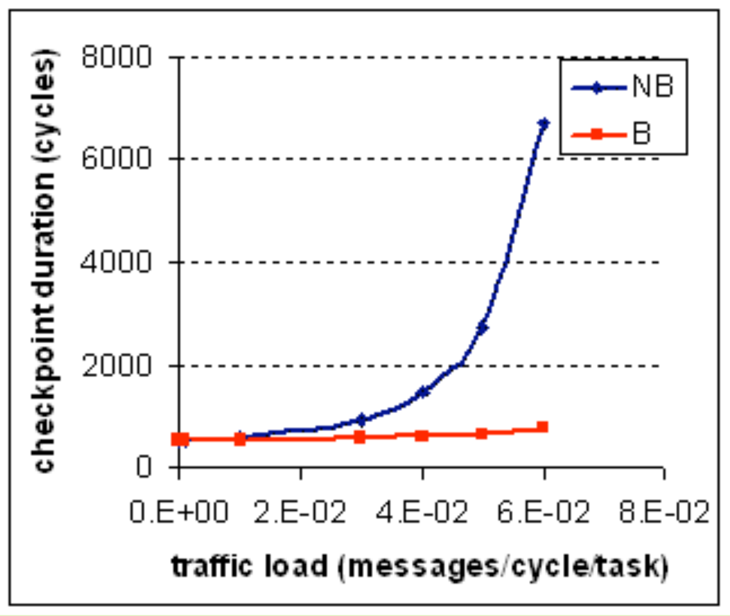
- **4x4 mesh direct NoC**
  - XY routing
  - Wormhole switching



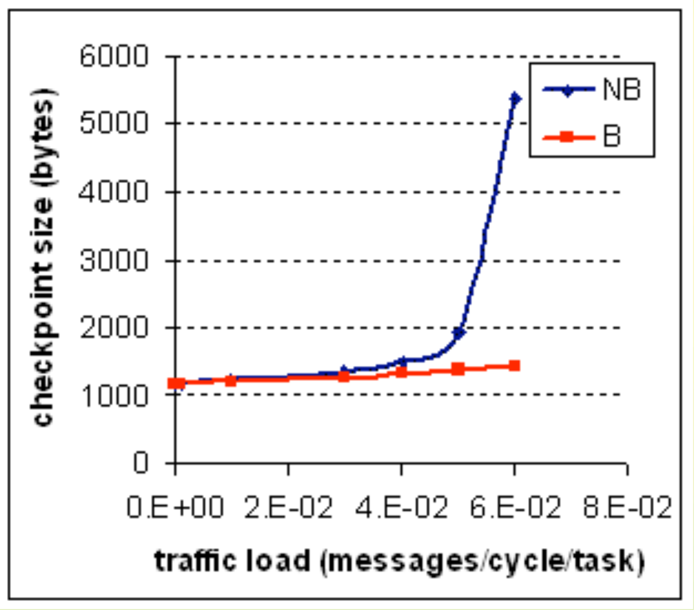
- **Consider**
  - Different traffic loads
    - uniform traffic loads
    - constant message length
  - Different failure rates
- **Analyze**
  - Checkpointing duration and overhead
  - Application latency

Checkpointing Duration and Overhead

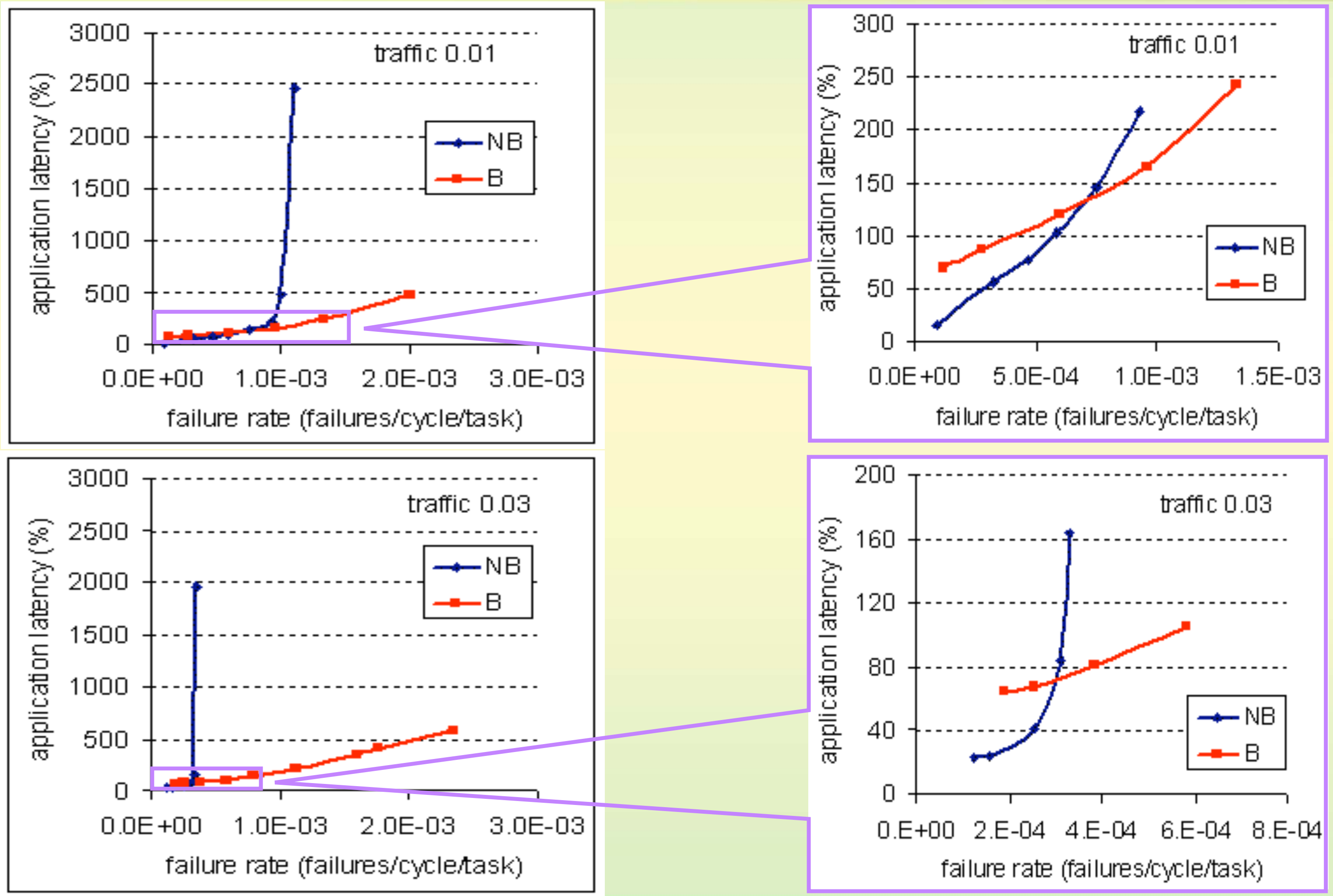
- **Checkpointing Duration**



- **Memory Overhead**



Application Latency



OUTLINE

- Introduction
  - Networks-on-Chip
  - Checkpoint and rollback recovery
- Coordinated checkpointing
- Blocking and non-blocking coordinated checkpointing
- Case study
- Conclusions and future work

Conclusions and Future Work

- Blocking and Non-blocking coordinated checkpointing
  - unique protocol
- Analyze and compare overhead and latency
  - Checkpointing duration increases with the traffic load
    - Non-blocking: significantly
    - Blocking: lesser
  - Application latency increases with the traffic load and the failure rate
    - Non-blocking: significantly
    - Blocking: lesser

  –> For higher traffic loads and higher failure rates, the blocking approach becomes mandatory

- Future work
  - Evaluate the proposed protocol
    - on other traffic patterns
    - on application with high traffic loads and critical tasks

      –> subsets of blocking and non-blocking tasks

Thank you!