ARTIST-Relevant Research from Linköping

Petru Eles

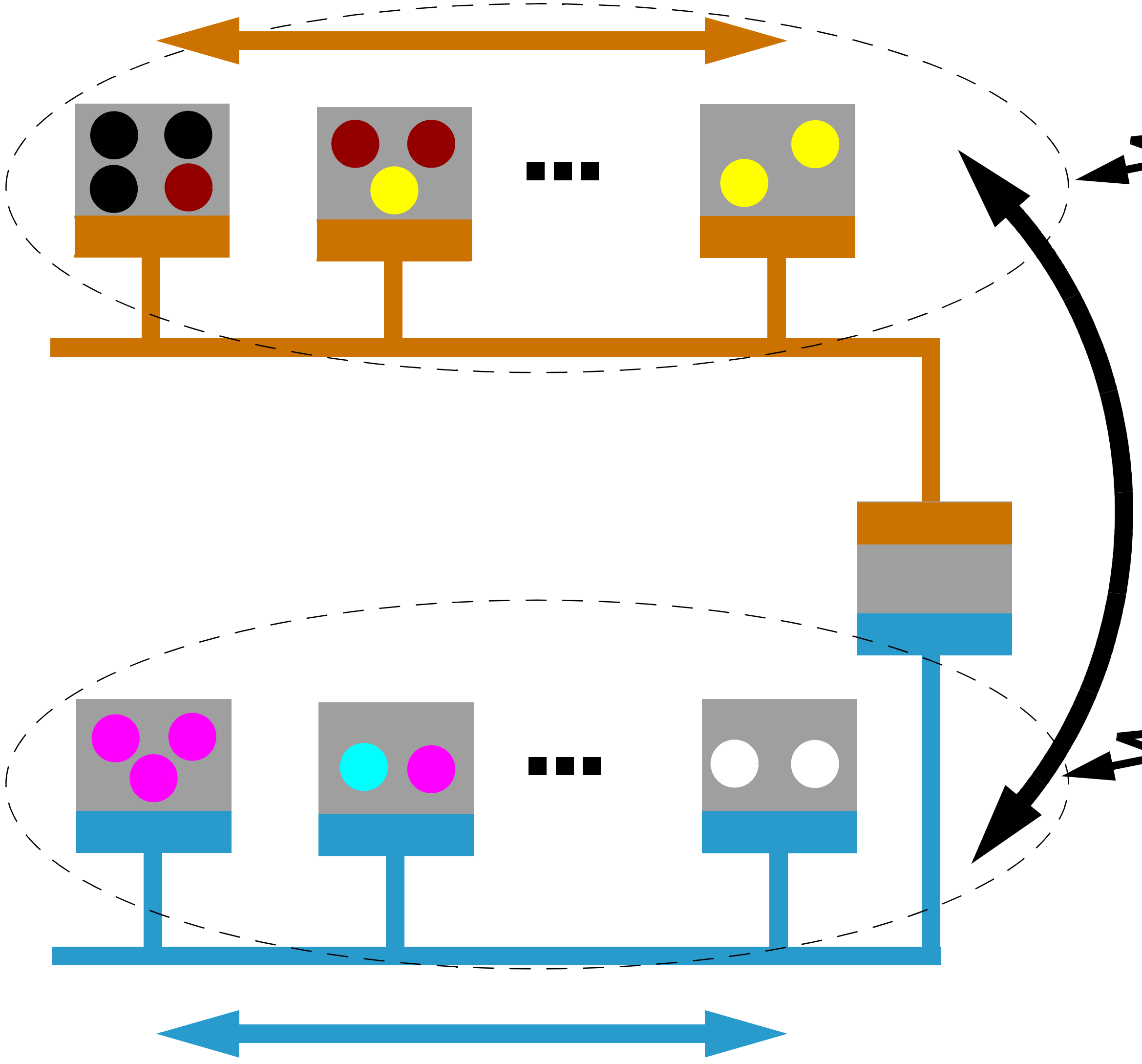
Department of Computer and Information Science (IDA)
Linköping University
http://www.ida.liu.se/~eslab/

Outline

- Communication-Intensive Real-Time Systems
  - Timing Analysis and Optimisation with FlexRay
  - Time -and Buffer Space Analysis for NoCs
  - A Simulator for Distributed Embedded Applications

- Predictability (even in the presence of faults)
  - Timing Predictability for Multiprocessors
  - Predictability in the Presence of Faults

Heterogeneous Distributed Embedded Systems



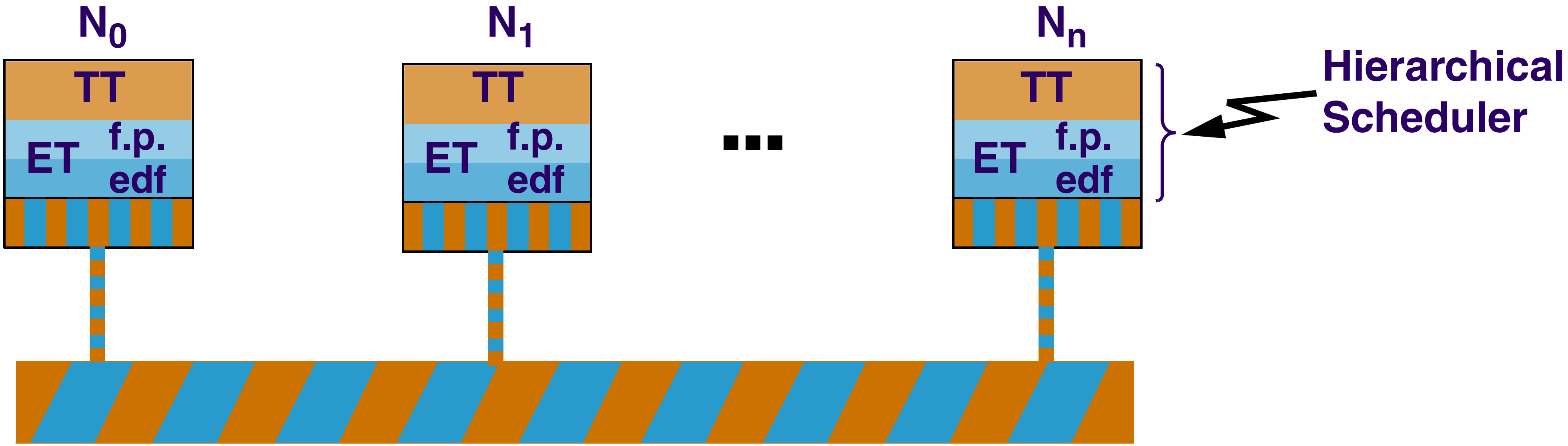
Time triggered cluster:

- TT tasks
- Static communication

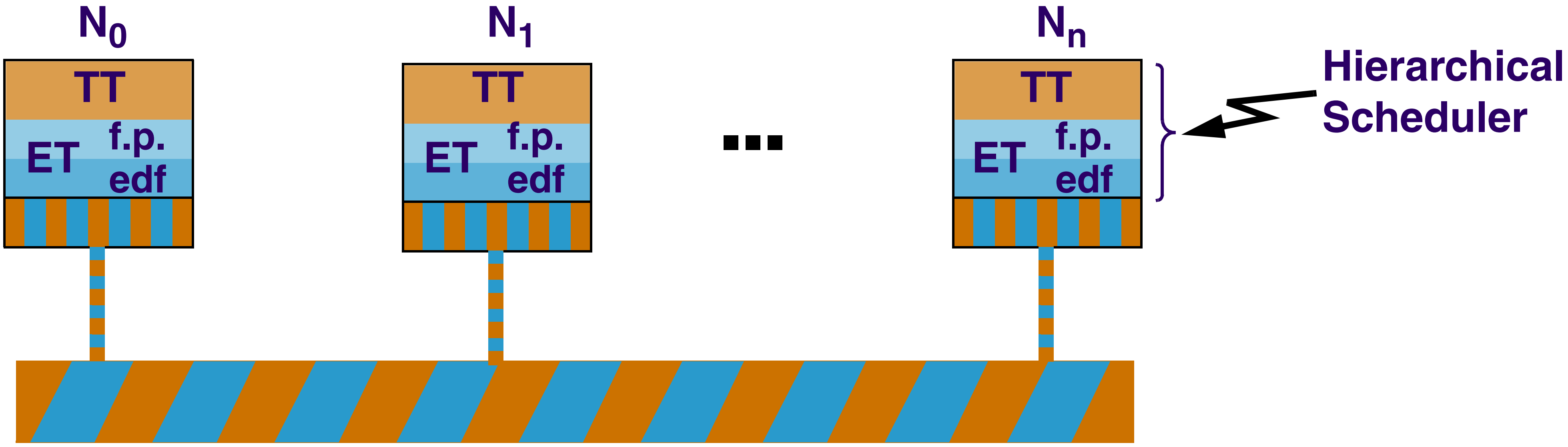
Event triggered cluster:

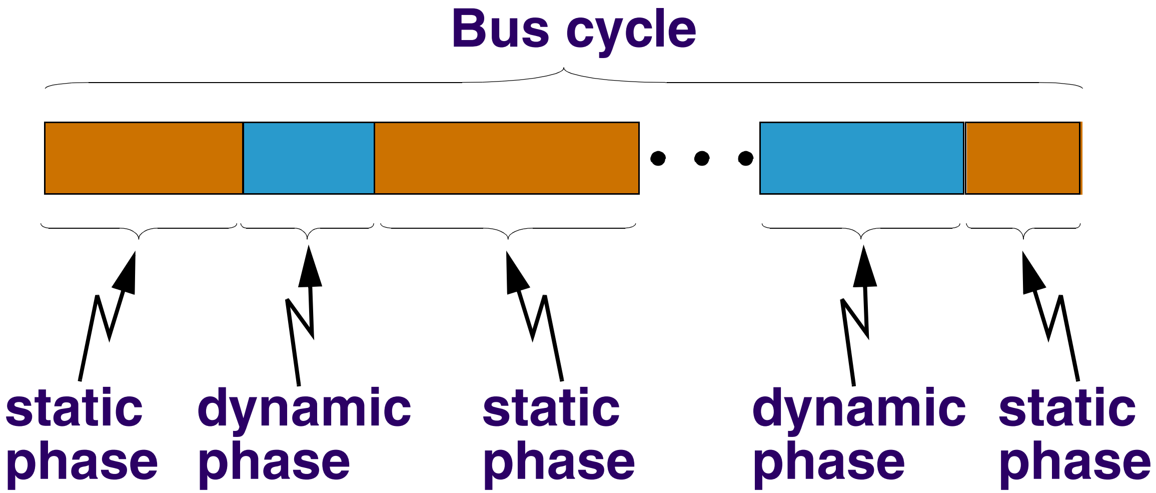
- ET tasks
- Dynamic communication

Heterogeneous Distributed Embedded Systems



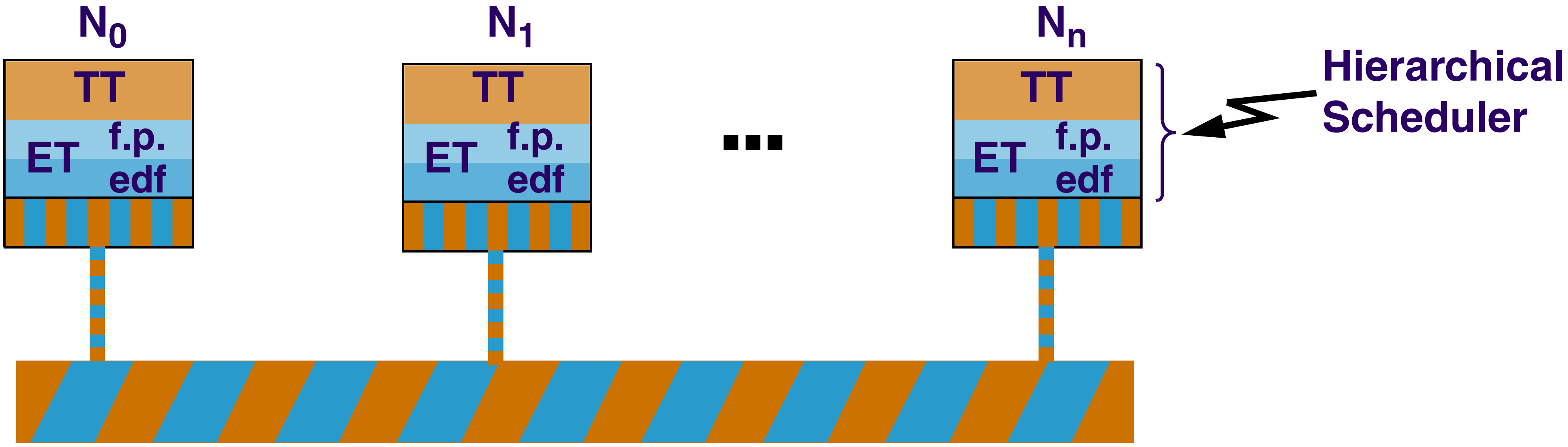
Heterogeneous Distributed Embedded Systems

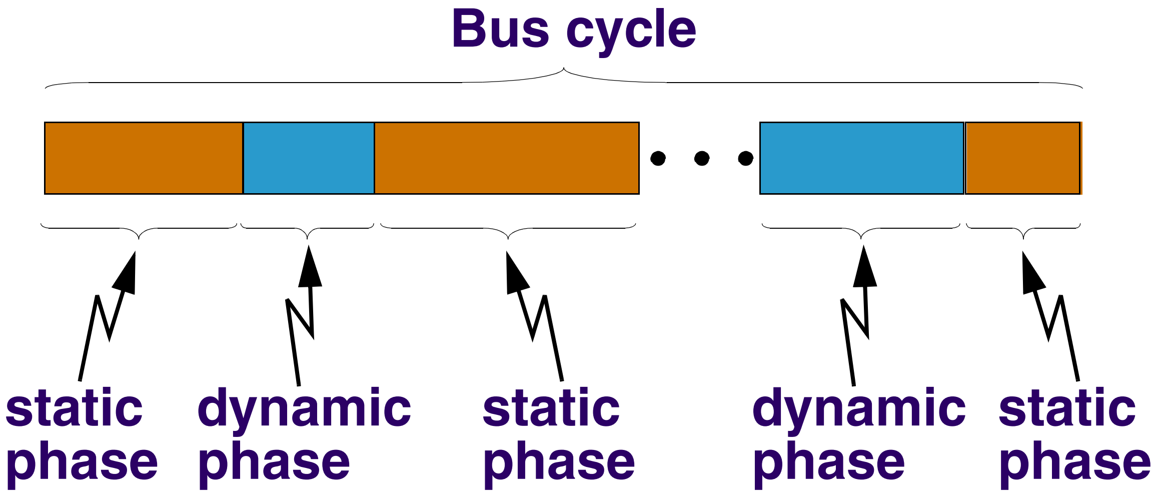




- UCM (TTP&CAN)
- FlexRay

FlexRey-Based System

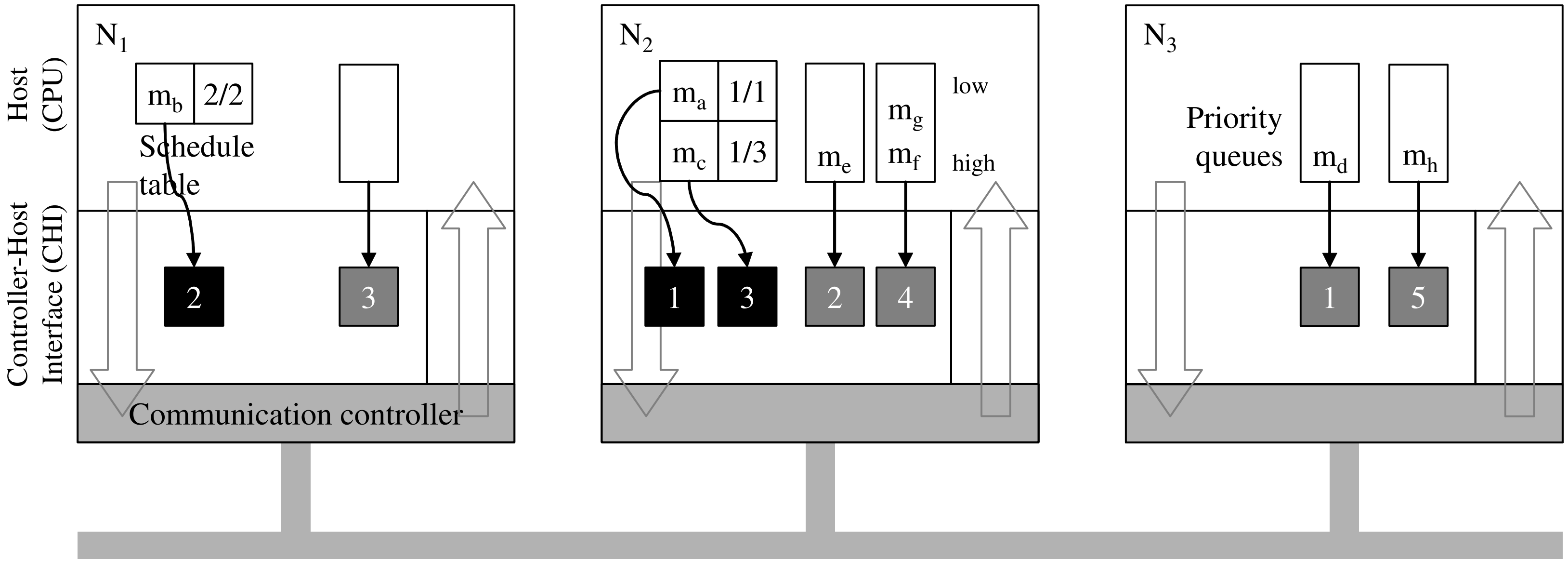


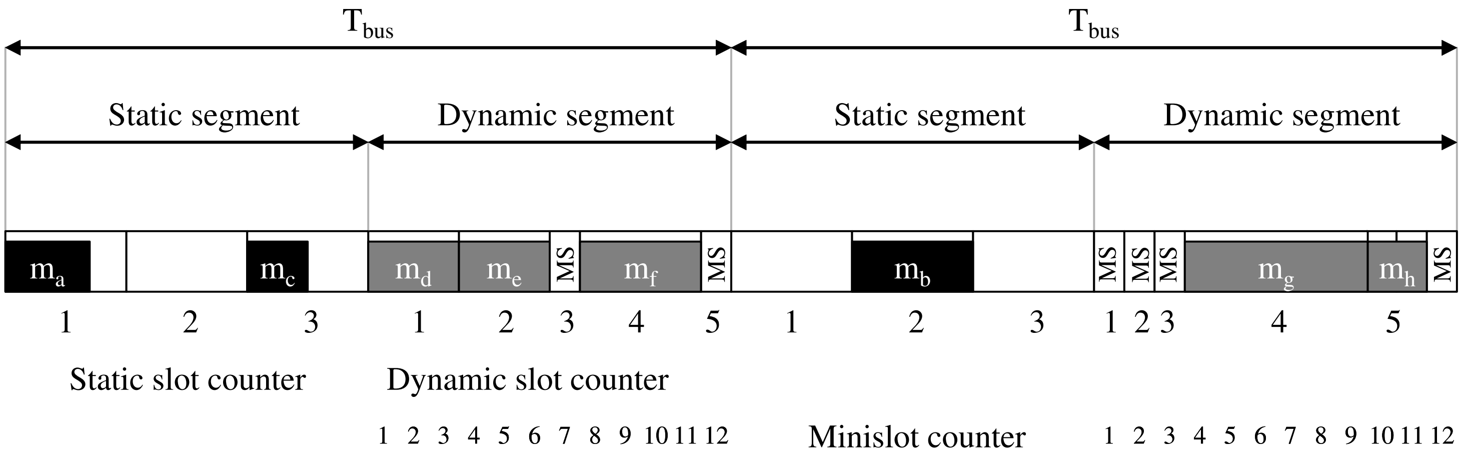


☞ FlexRay

- Static phase: TDMA
- Dynamic phase: Flexible TDMA

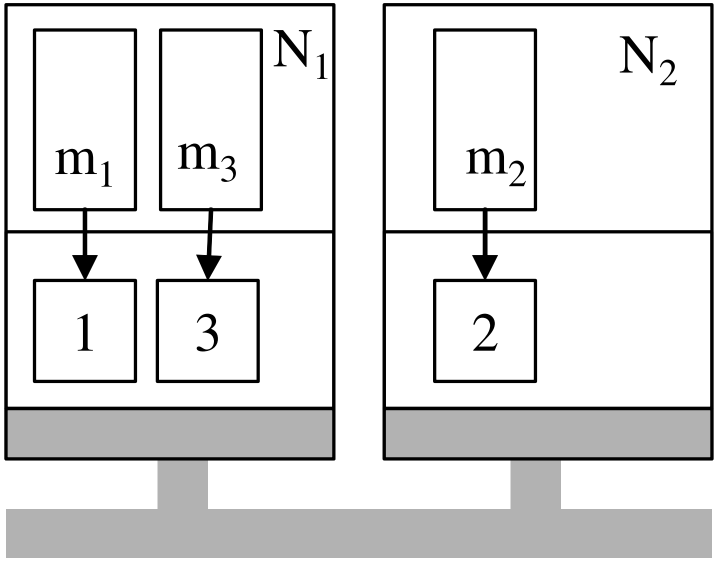
FlexRey-Based System

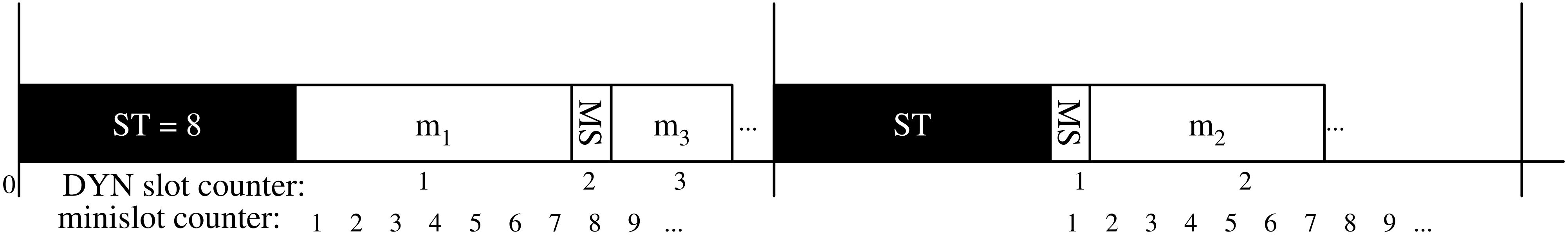




FlexRey-Based System

Strange Priority Inversions!





FlexRey-Based System

- Response-time analysis cannot be solved by “just” extending response time analysis for priority-based scheduling, like for CAN.

- Determining the number of bus cycles a message has to wait, is - in a simplified formulation - a *bin covering problem*.

FlexRey-Based System

- Response-time analysis cannot be solved by “just” extending response time analysis for priority-based scheduling, like for CAN.

- Determining the number of bus cycles a message has to wait, is - in a simplified formulation - a *bin covering problem*.

- Bus access optimisation
  - assign FrameID to nodes and messages
  - determine size of dynamic/static segment
  - determine number of static slots

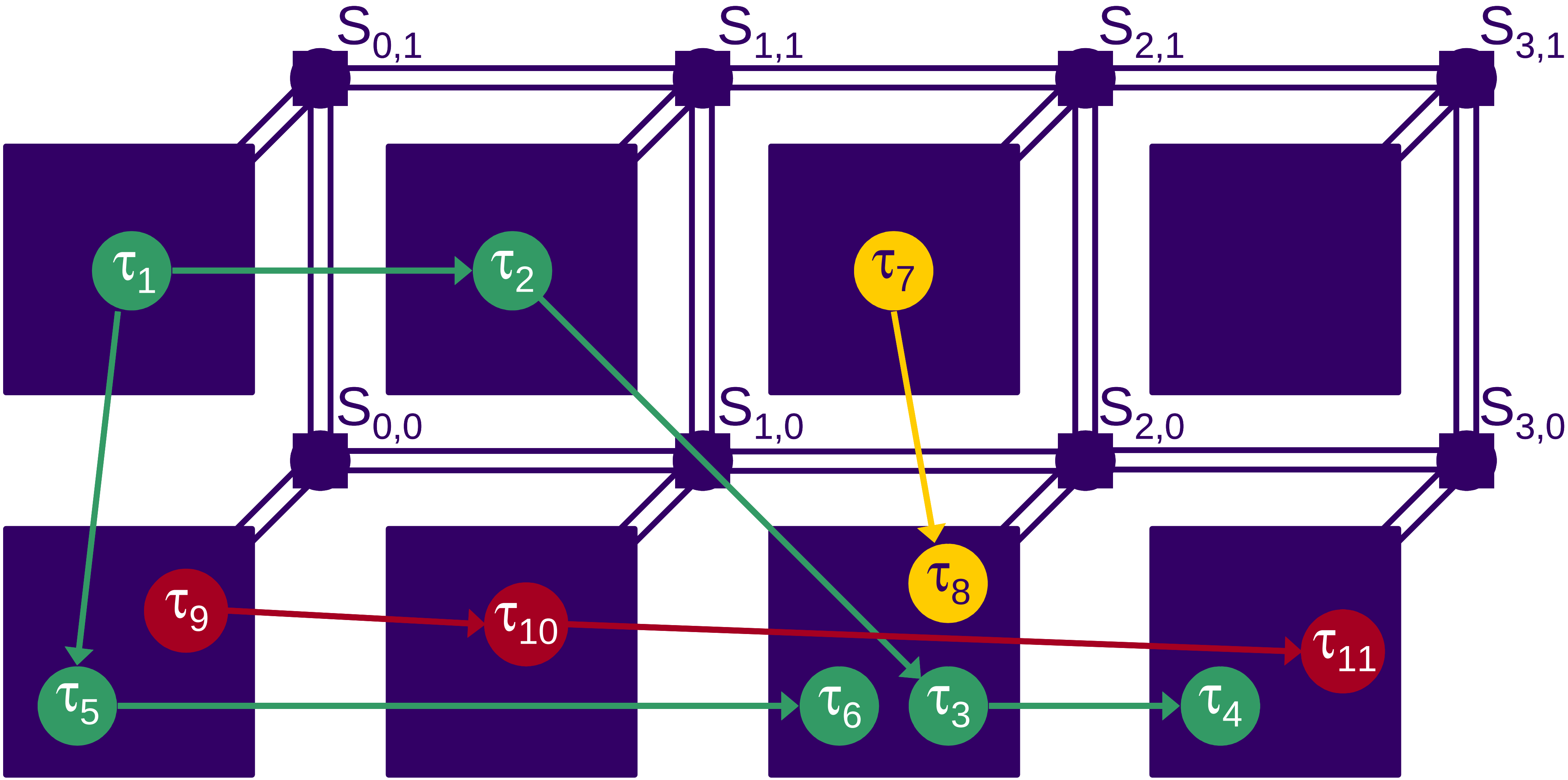
ECRTS 06

Outline

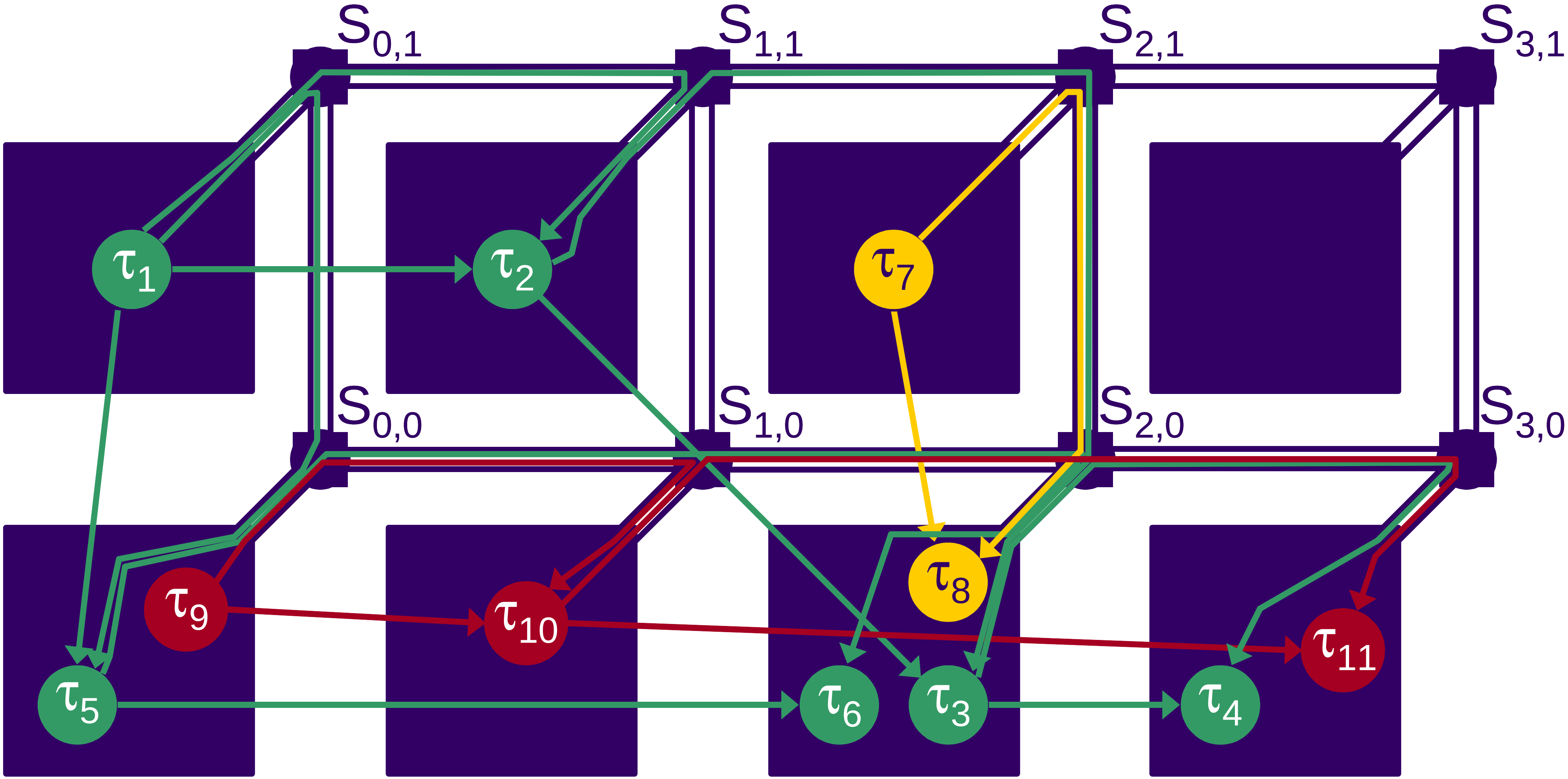
- Communication-Intensive Real-Time Systems
  - Timing Analysis and Optimisation with FlexRay
  - Time -and Buffer Space Analysis for NoCs
  - A Simulator for Distributed Embedded Applications

- Predictability (even in the presence of faults)
  - Timing Predictability for Multiprocessors
  - Predictability in the Presence of Faults

Time -and Buffer Space Analysis for NoCs



Time -and Buffer Space Analysis for NoCs



Time -and Buffer Space Analysis for NoCs

☞ Scenario in which an *application-specific* NoC is built

- Find a communication mapping and the packet release times of all packets and determine the amount of buffer memory at each switch such that
  - No deadline is missed and no buffer overflow occurs
  - The total amount of buffer memory is minimised
  - Message arrival probability is above a specified threshold given a link failure model.

Time -and Buffer Space Analysis for NoCs

☞ Scenario in which the application is implemented on an *existing NoC with given buffer memory* at each switch

- Find a communication mapping and the packet release times of all packets such that
  - No deadline is missed and no buffer overflow occurs
  - Message arrival probability is above a specified threshold given a link failure model.

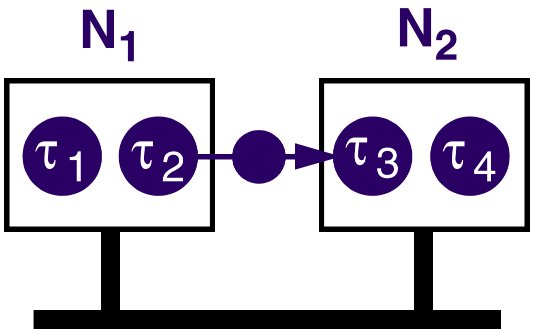
DATE 06

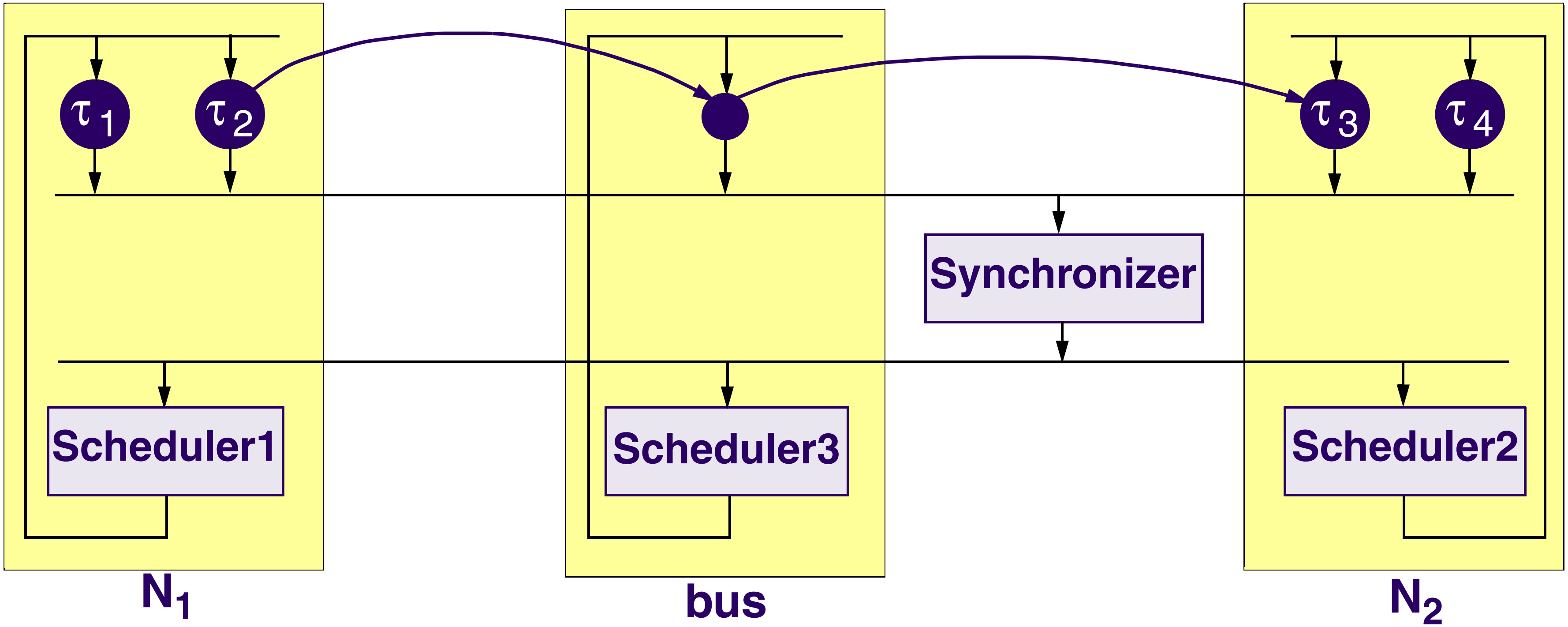
Outline

- Communication-Intensive Real-Time Systems
  - Timing Analysis and Optimisation with FlexRay
  - Time -and Buffer Space Analysis for NoCs
  - A Simulator for Distributed Embedded Applications

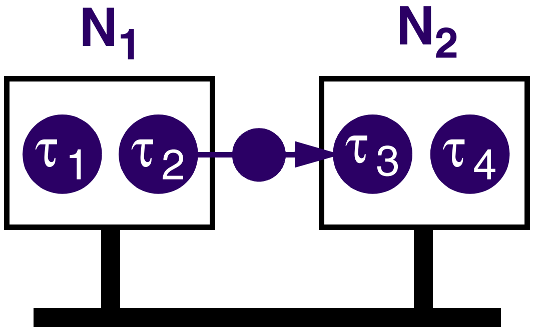
- Predictability (even in the presence of faults)
  - Timing Predictability for Multiprocessors
  - Predictability in the Presence of Faults

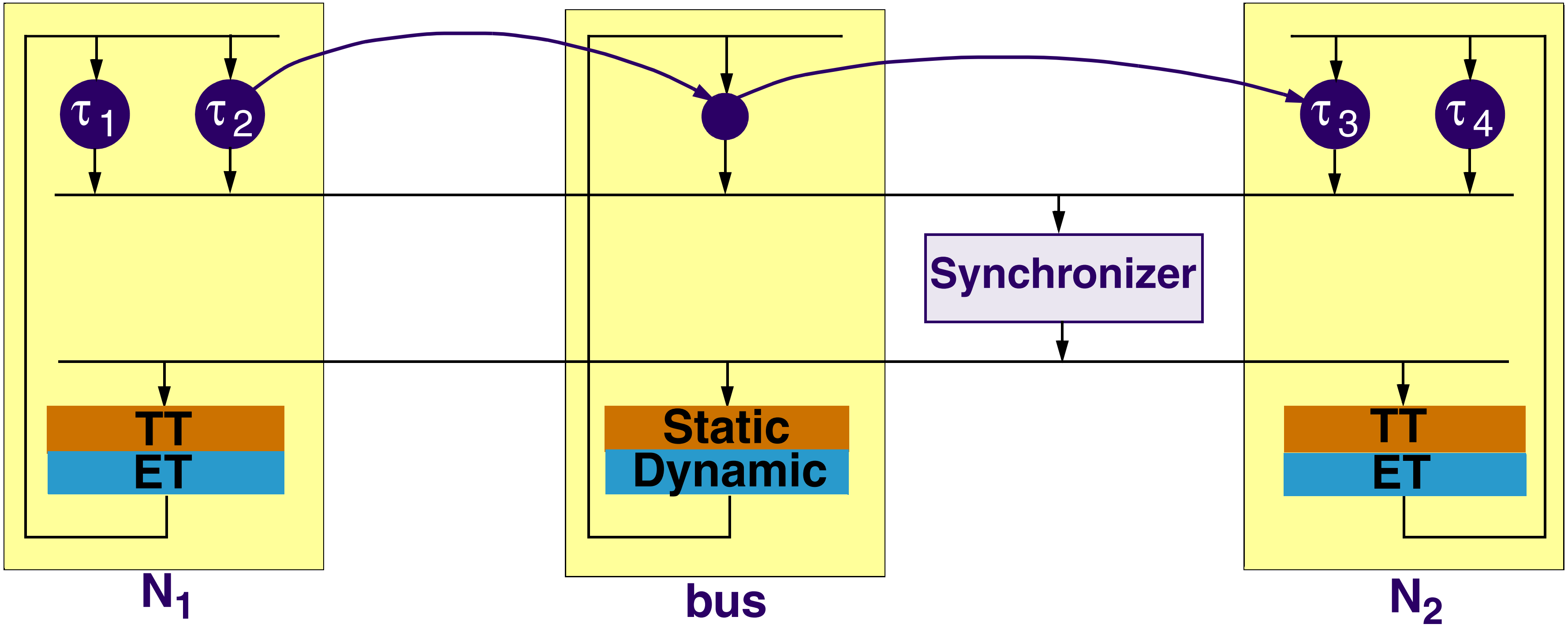
A Simulator for Distributed Embedded Applications



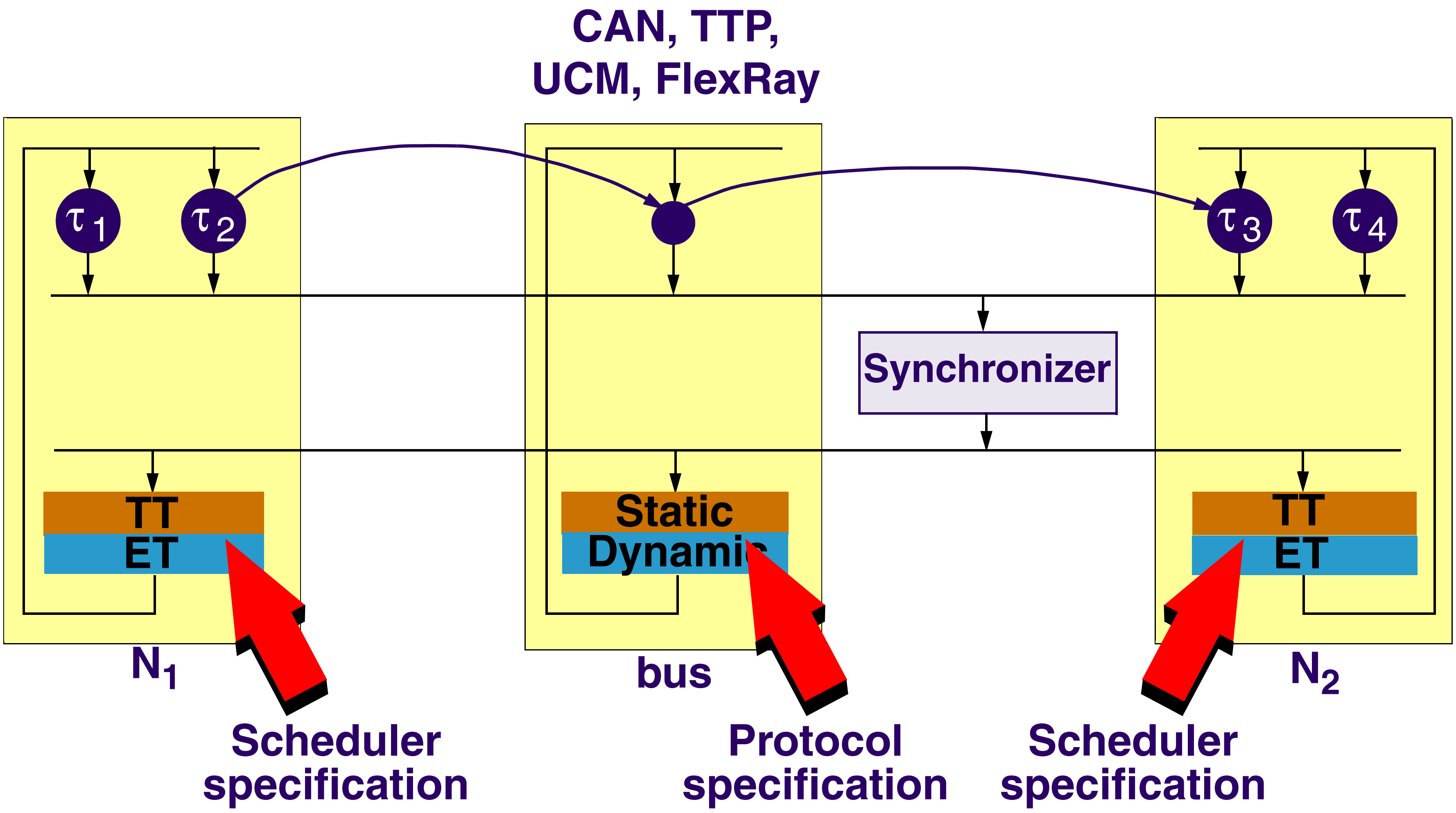


A Simulator for Distributed Embedded Applications





A Simulator for Distributed Embedded Applications



A Simulator for Distributed Embedded Applications

- Compare
  - Scheduling approaches
  - Communication protocols

A Simulator for Distributed Embedded Applications

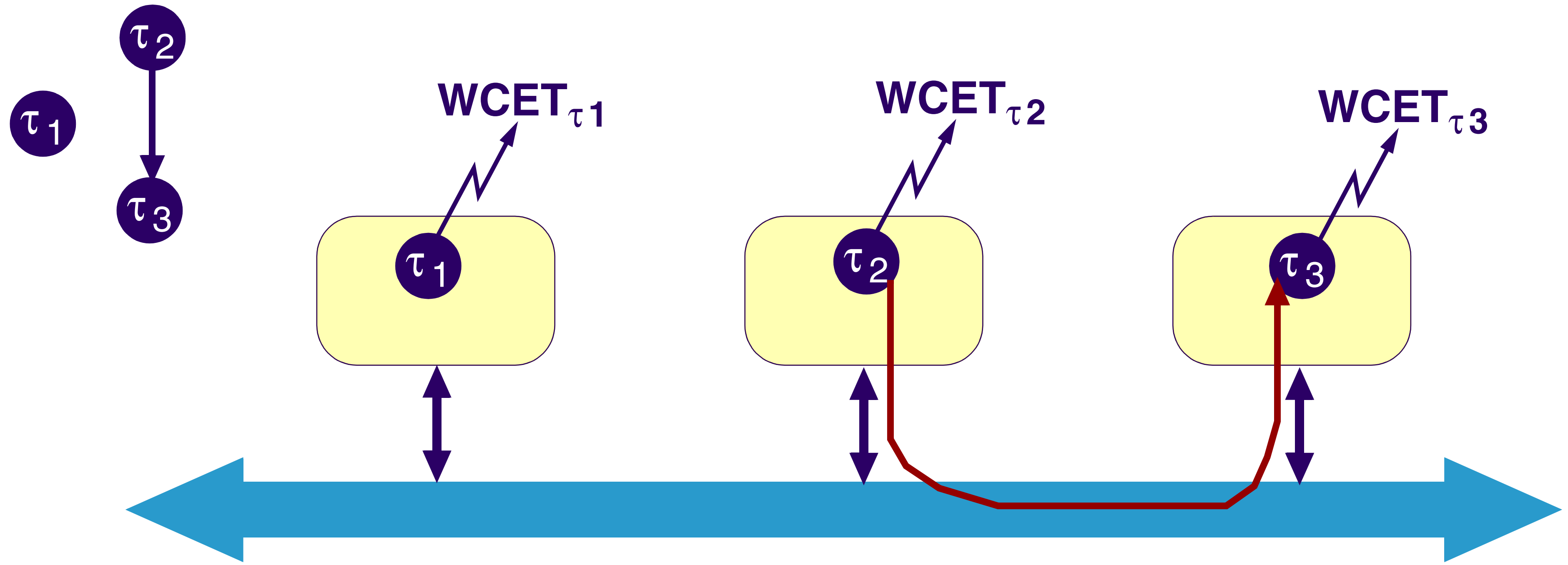
- Compare
  - Scheduling approaches
  - Communication protocols
- Interesting issues to look at:
  - Pessimism of analysis
  - Jitter, delay
  - Quality of Control
- Syntetic applications and actual code

Outline

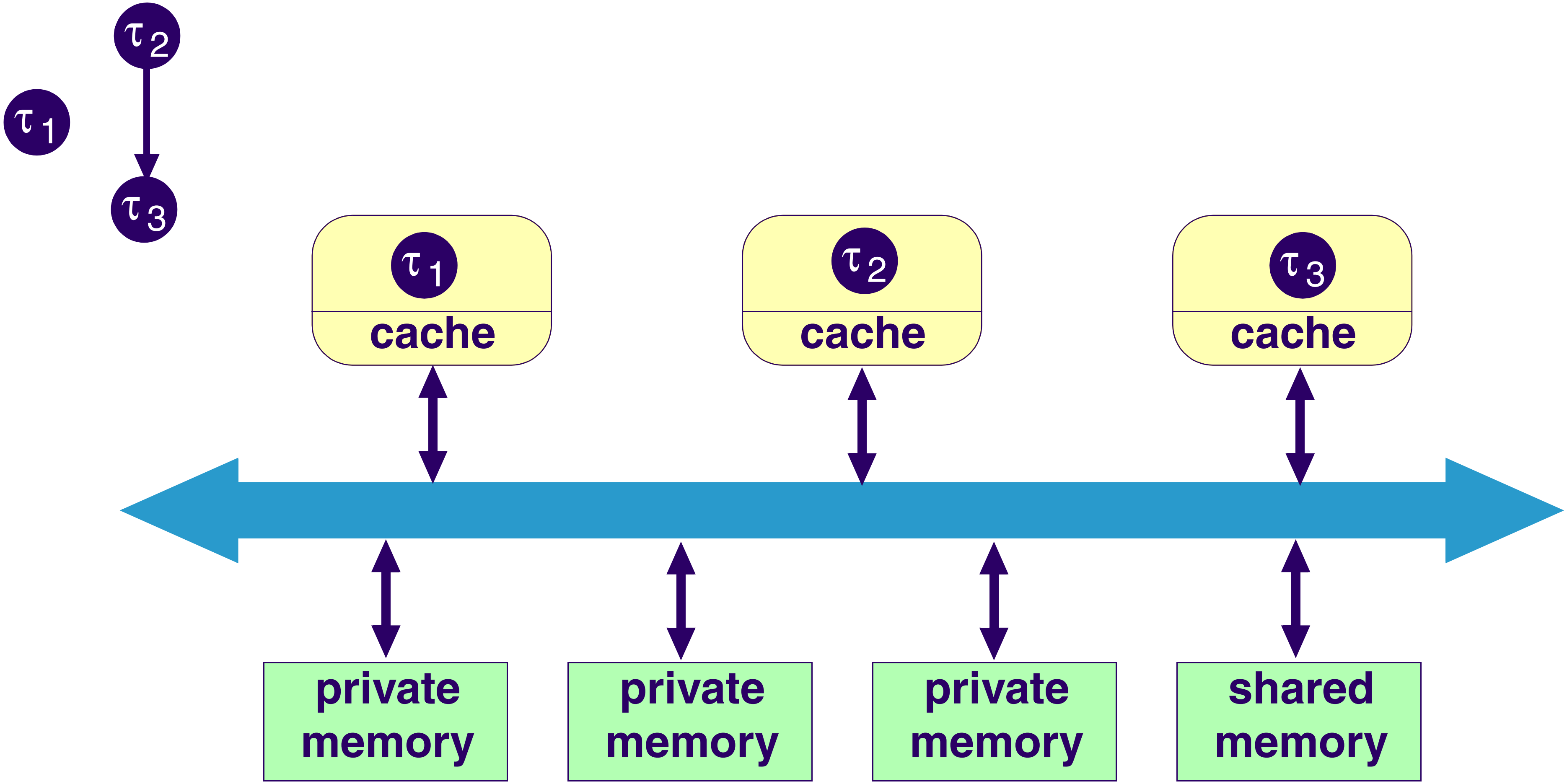
- Communication-Intensive Real-Time Systems
  - Timing Analysis and Optimisation with FlexRay
  - Time -and Buffer Space Analysis for NoCs
  - A Simulator for Distributed Embedded Applications

- Predictability (even in the presence of faults)
  - Timing Predictability for Multiprocessors
  - Predictability in the Presence of Faults

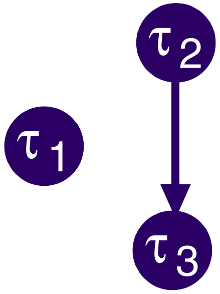
Predictability



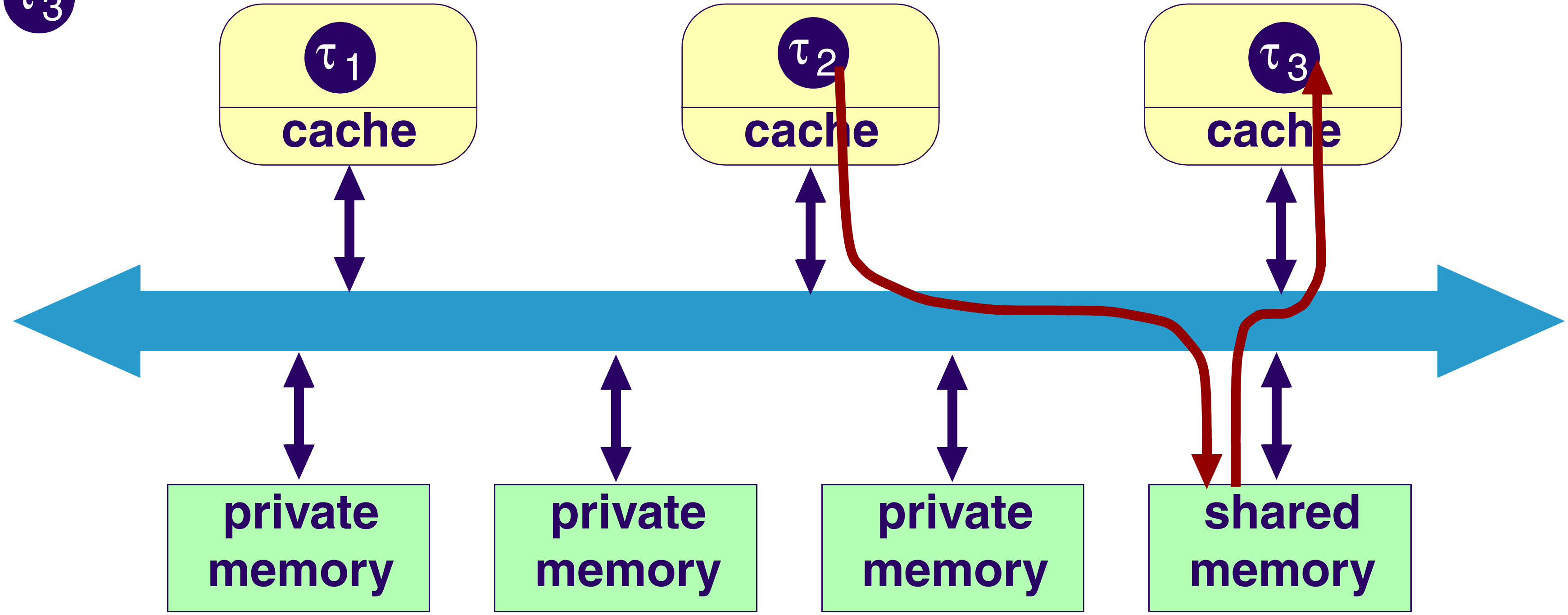
Predictability



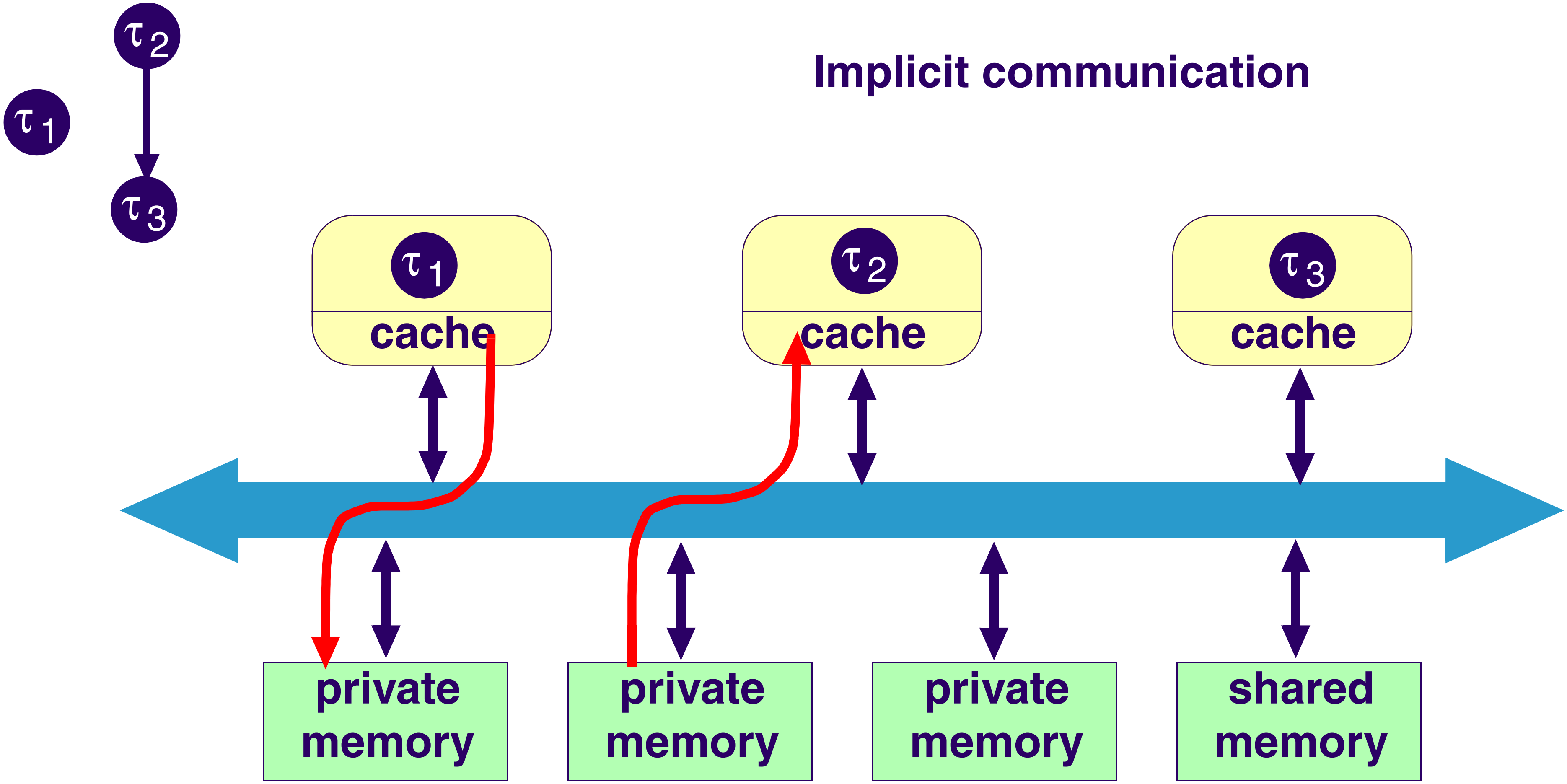
Predictability



Explicit communication

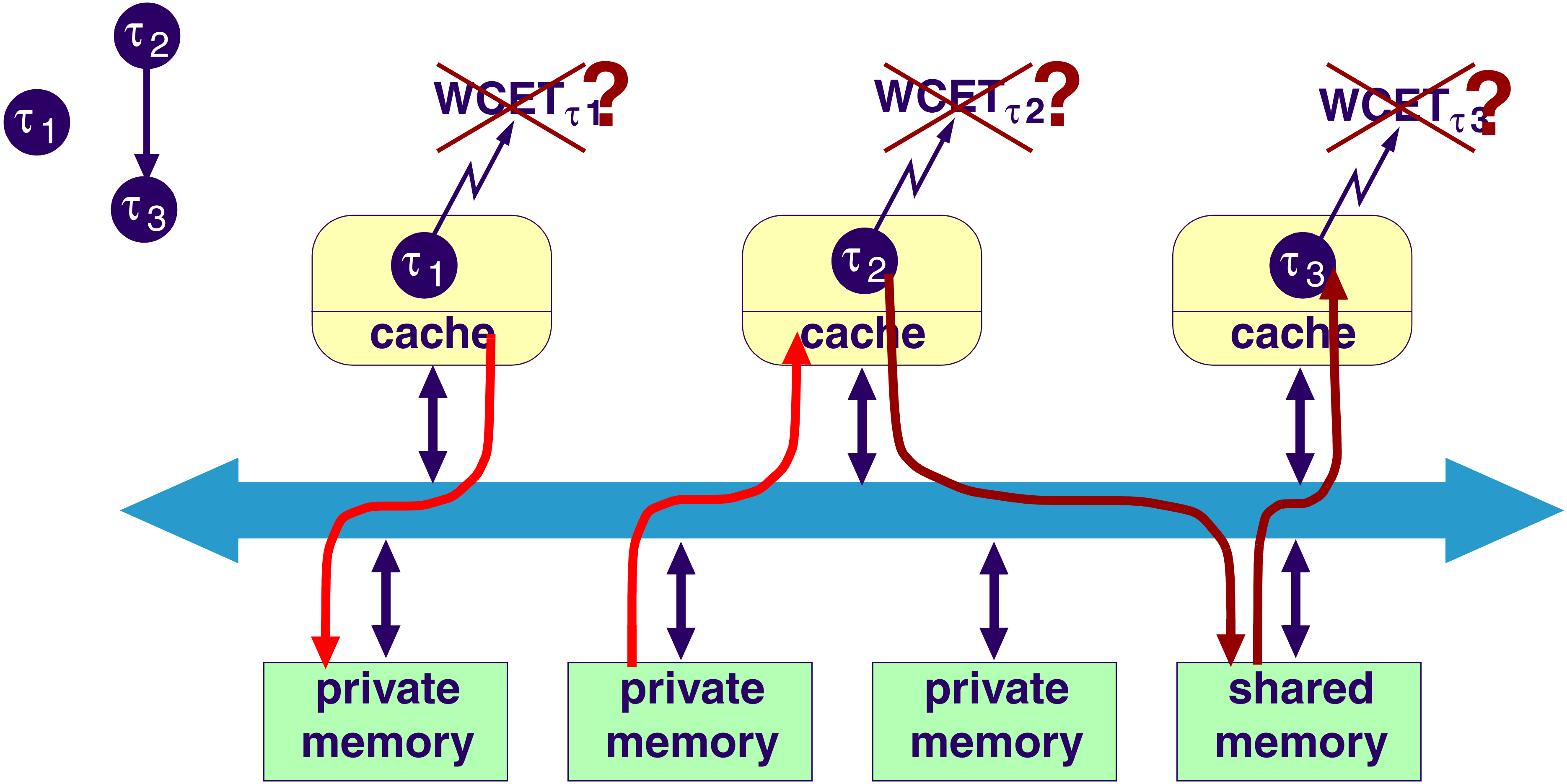


Predictability



Implicit communication

Predictability



Predictability

- **WCET cannot be determined by taking tasks in isolation.**

- **WCET analysis has to be brought into the context of system analysis and optimisation.**

Predictability

- WCET cannot be determined by taking tasks in isolation.

- WCET analysis has to be brought into the context of system analysis and optimisation.

- Trade-offs:
  - Local WCET vs. global schedulability
  - What is the cost of predictability?

Another Issue: Faults

- ☞ **Transient faults**
  - **Their number can be much larger than that of permanent faults.**

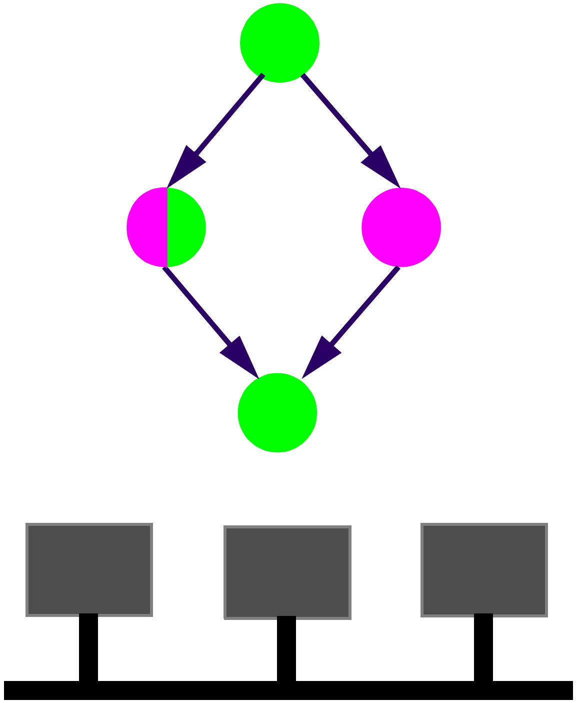
- **Find cost-effective implementations that are fault tolerant and satisfy time constraints.**

  Some Interesting trade-offs!

Fault Tolerance

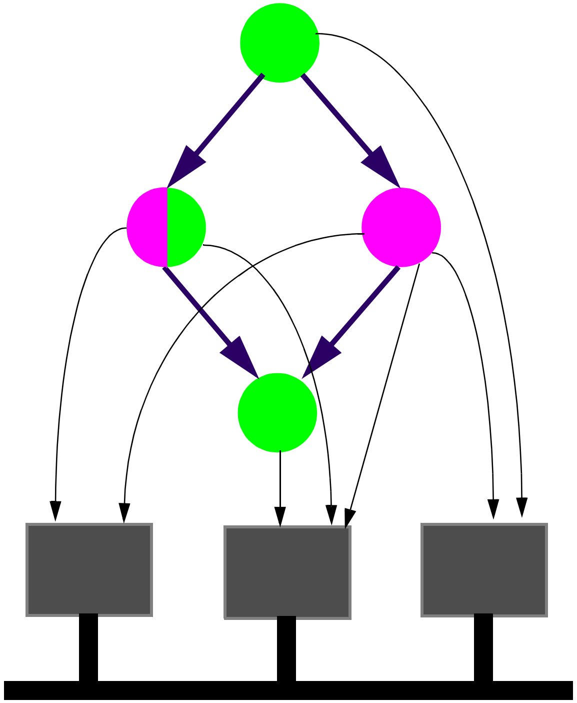
- Decide which fault tolerance technique to apply:
  - re-execution
  - re-exution&checkpointing
  - replication

different techniques can be applied to different tasks



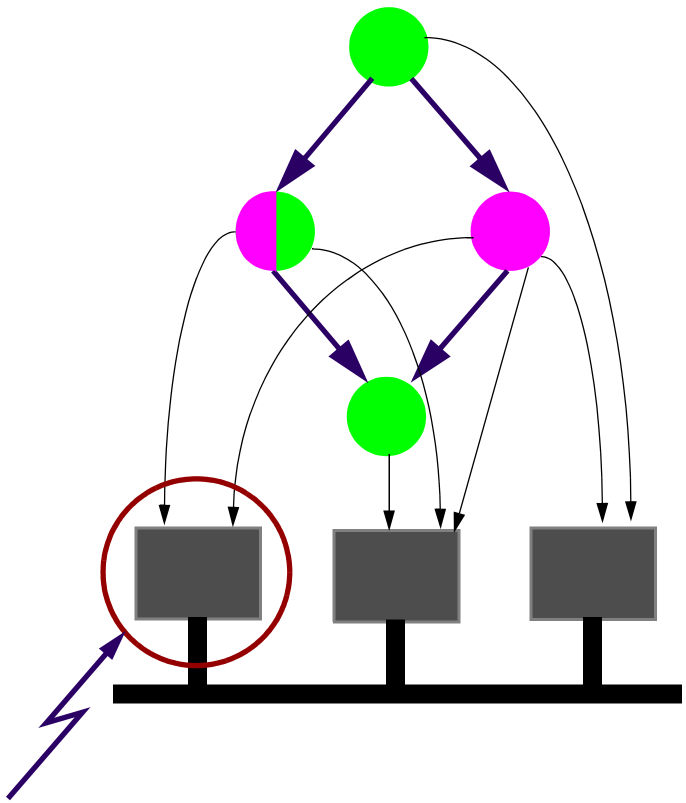
Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-exution&checkpointing
  - replication
- Map the tasks (including eventual replicas)



Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-exution&checkpointing
  - replication
- Map the tasks (including eventual replicas)
- Decide on transparency



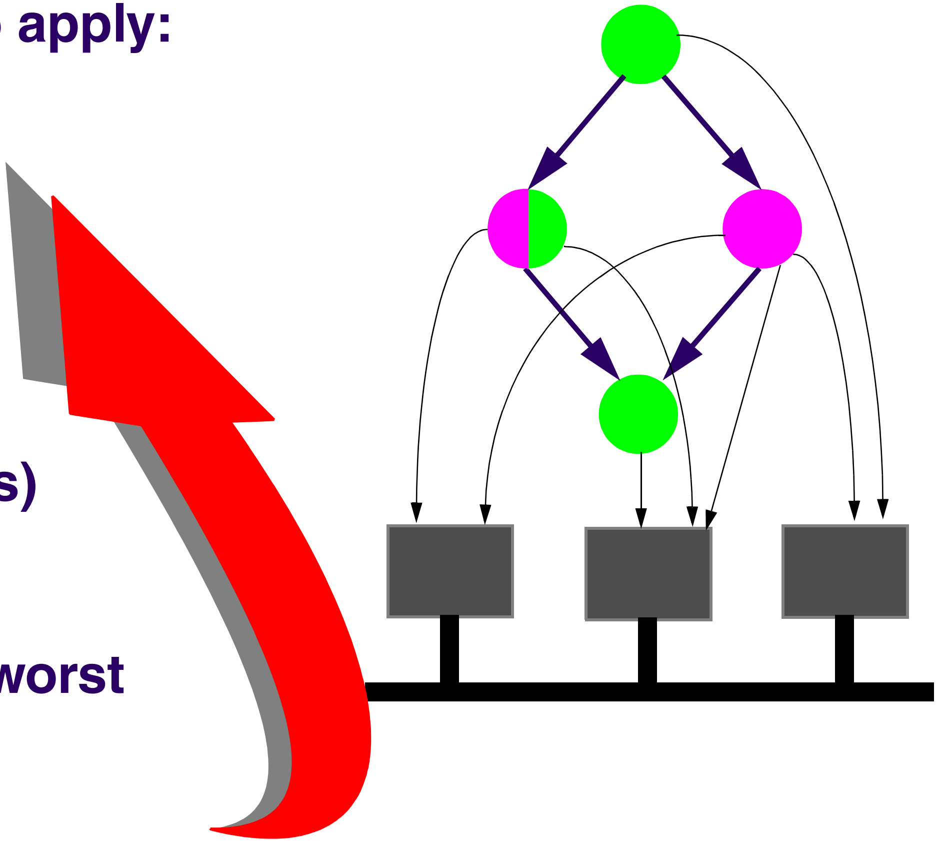
Transparent: The schedule of outgoing messages does not depend on occurrence of faults (faults are not visible to the outside).

Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-exution&checkpointing
  - replication

- Map the tasks (including eventual replicas)
- Decide on transparency
- Do the analysis/scheduling, considering worst case number of faults (re-executions).

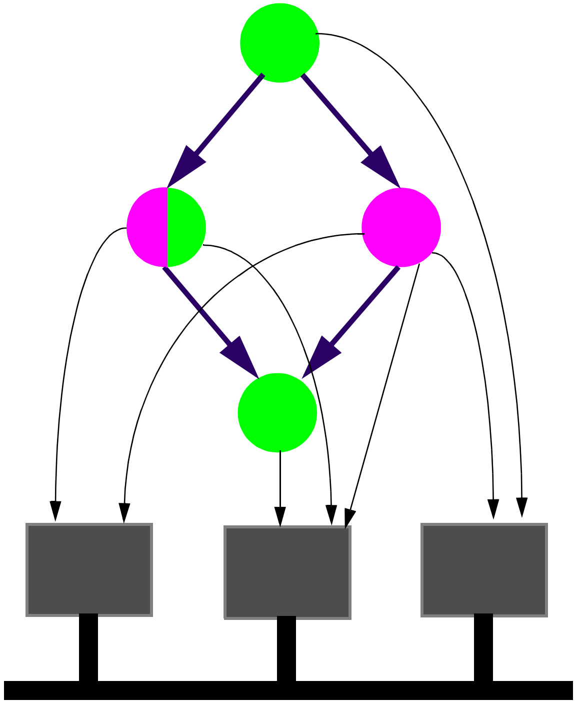
  Are time constraints satisfied? If not, go back!



Fault Tolerance

- Decide which fault tolerance technique to apply:
  - re-execution
  - re-exution&checkpointing
  - replication
- Map the tasks (including eventual replicas)
- Decide on transparency
- Do the analysis/scheduling, considering worst case number of faults (re-executions). Are time constraints satisfied? If not, go back!

☞ Which is the optimal number of check-points?



DATE 05
DATE 06