Analysis and Optimization of Distributed Real-Time Embedded Systems

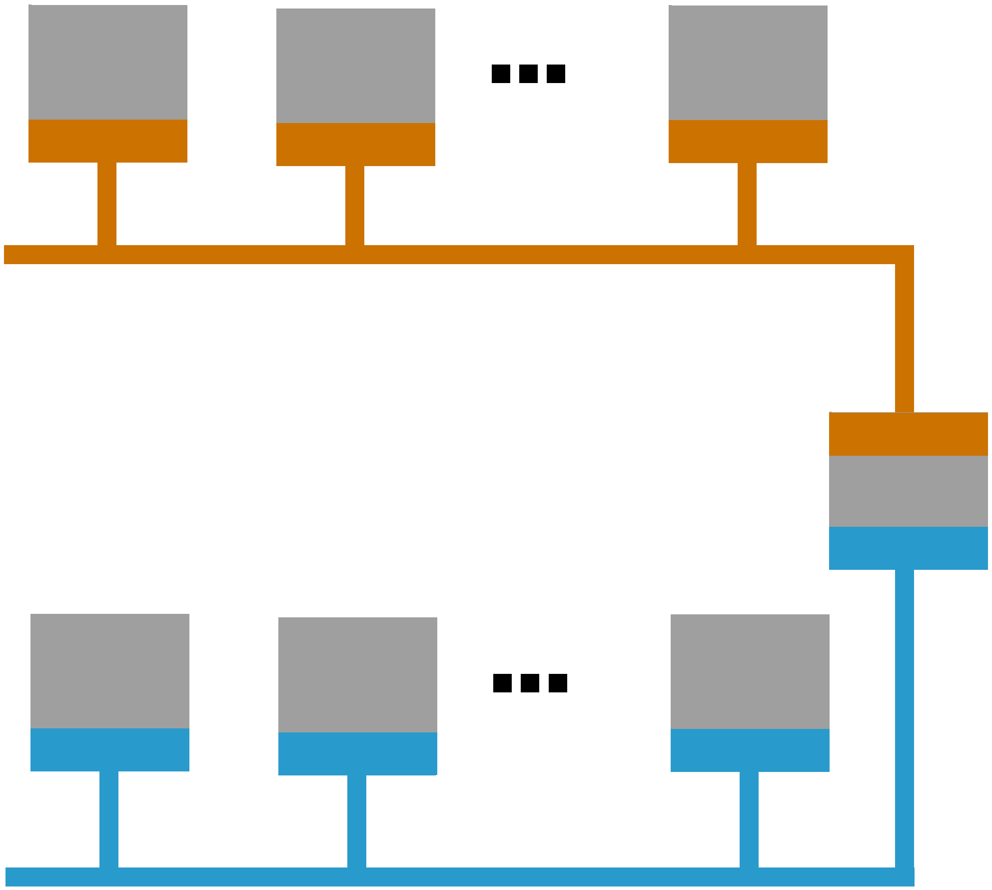
Paul Pop, Traian Pop, Sorin Manolache,
Petru Eles, Zebo Peng

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

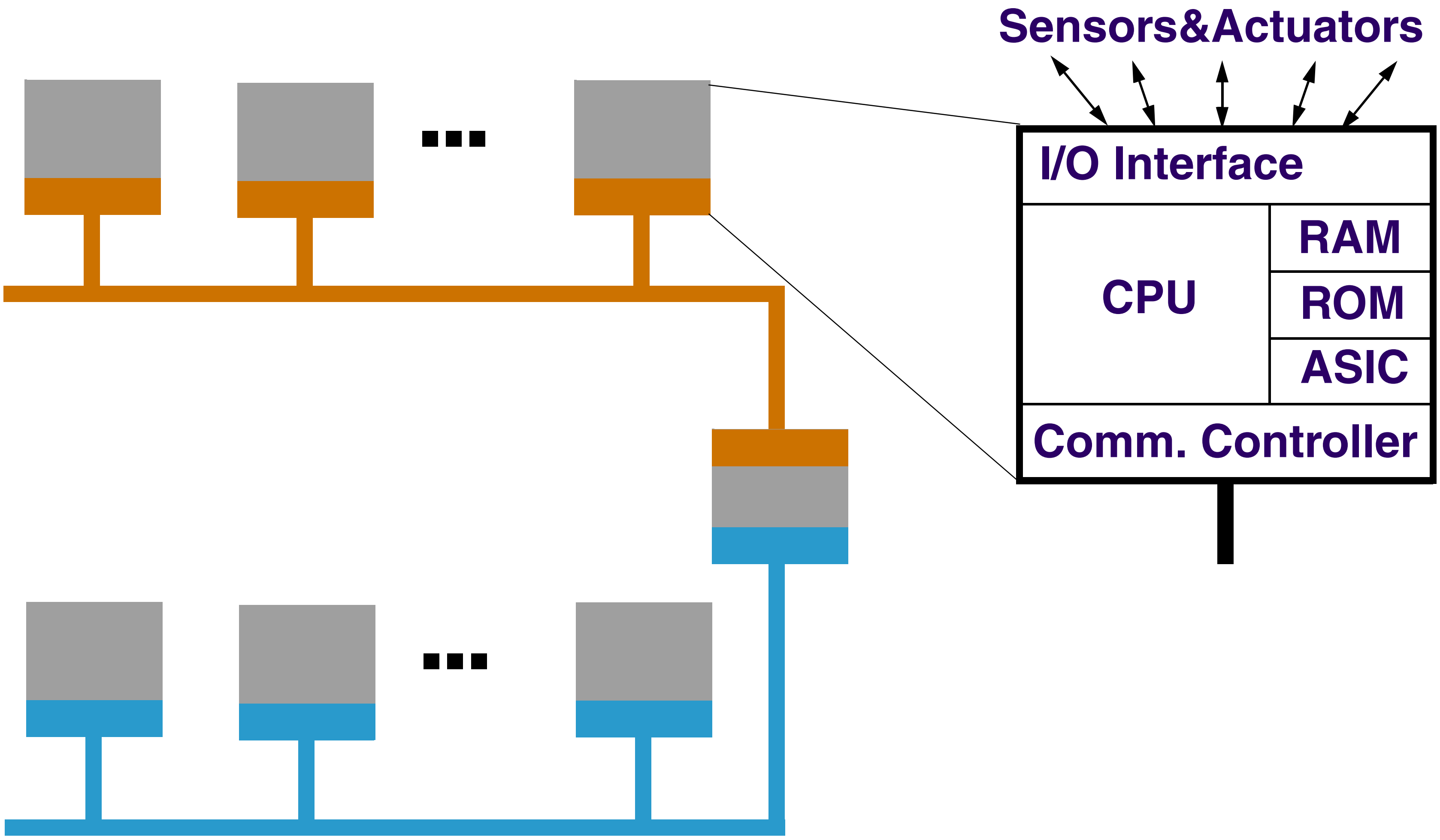
Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization

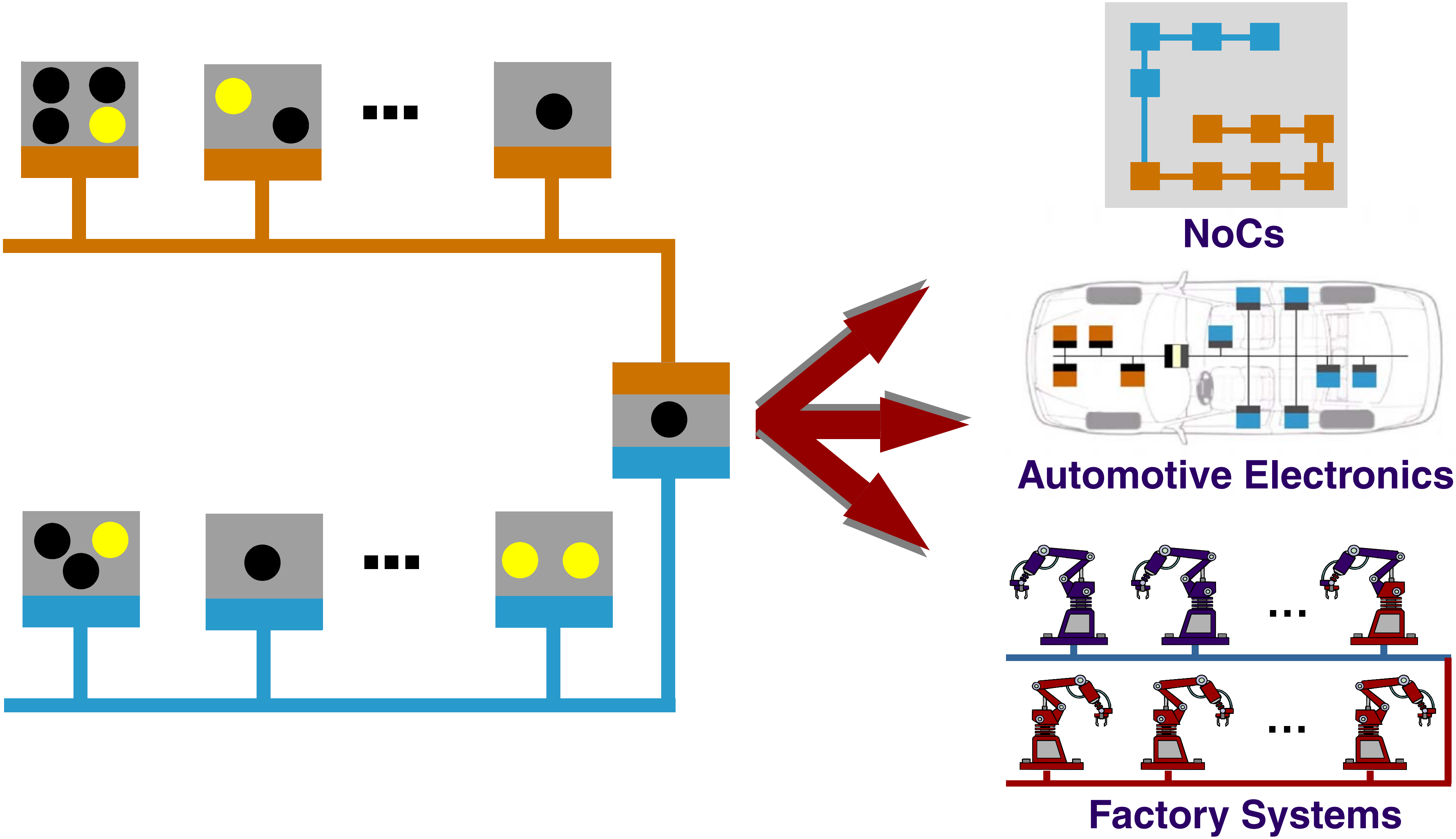
Distributed Embedded Systems



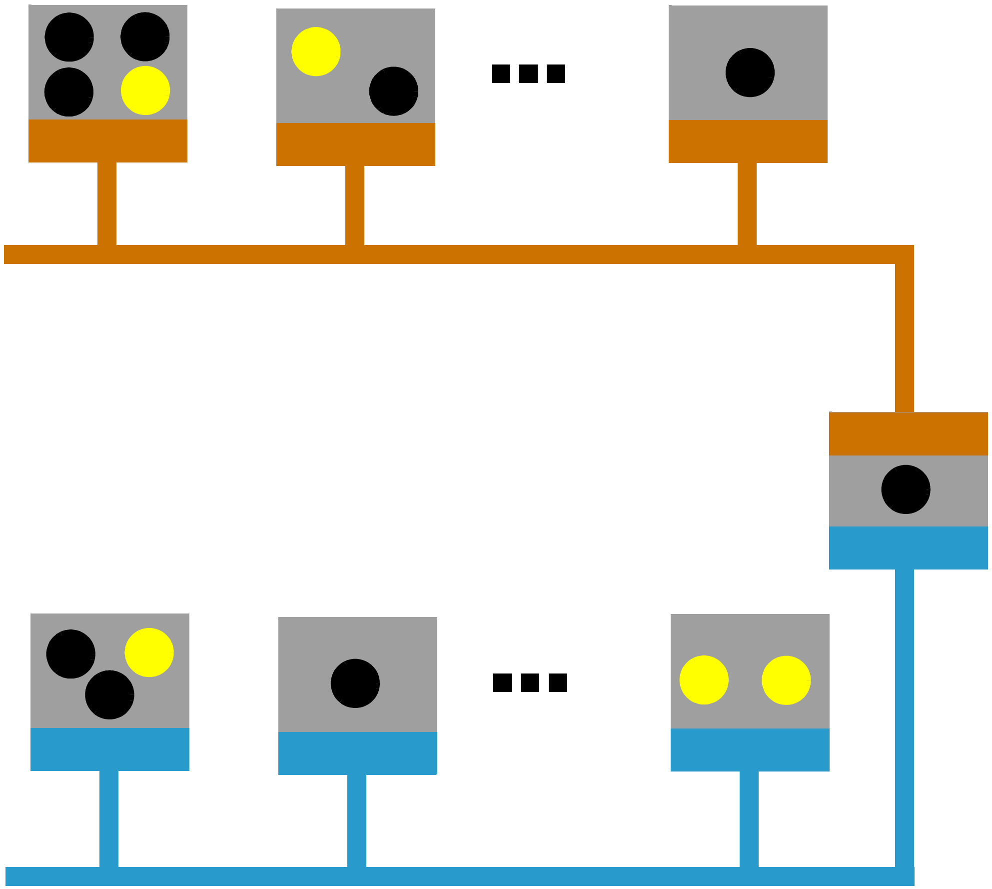
Distributed Embedded Systems



Distributed Embedded Systems



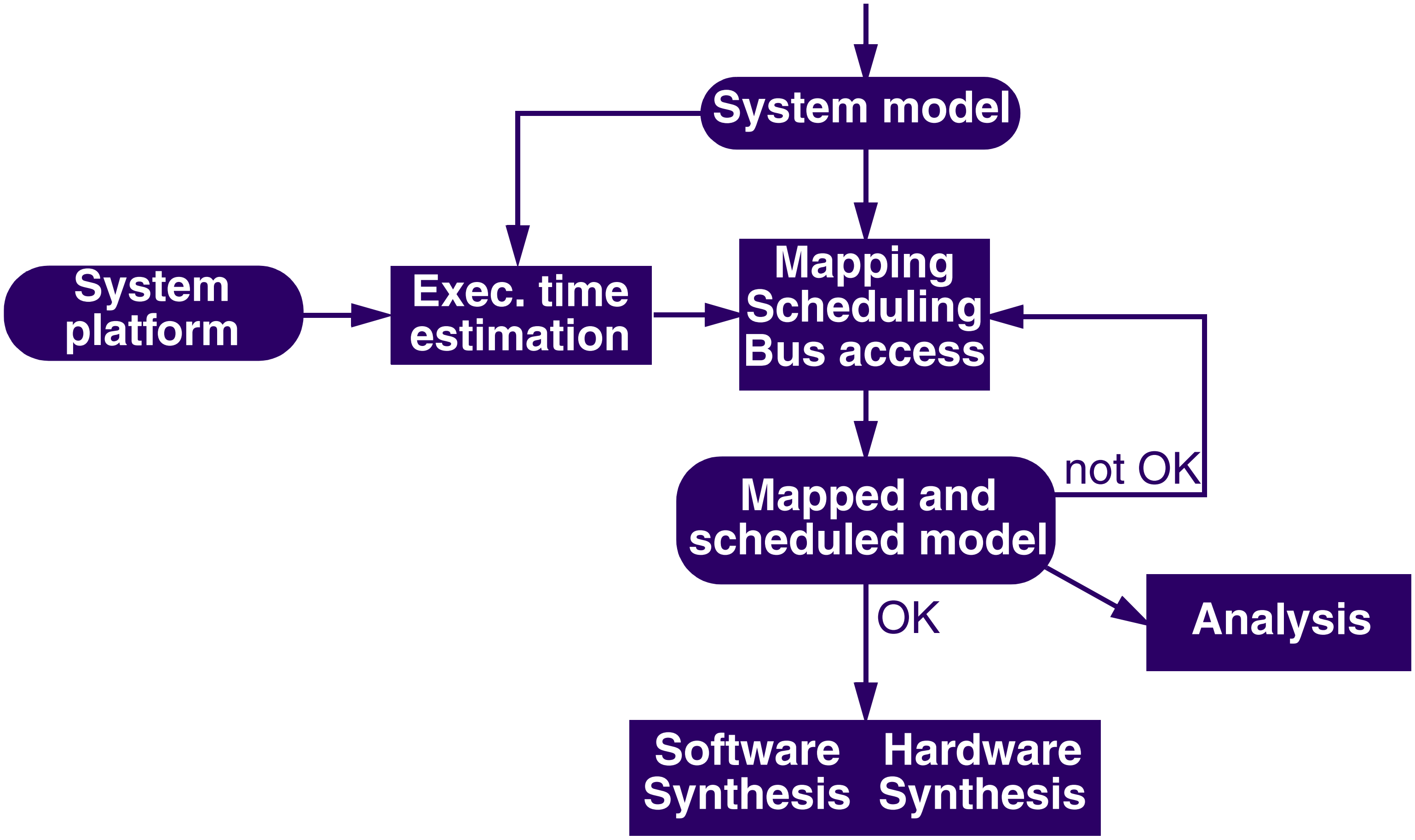
Distributed Embedded Systems



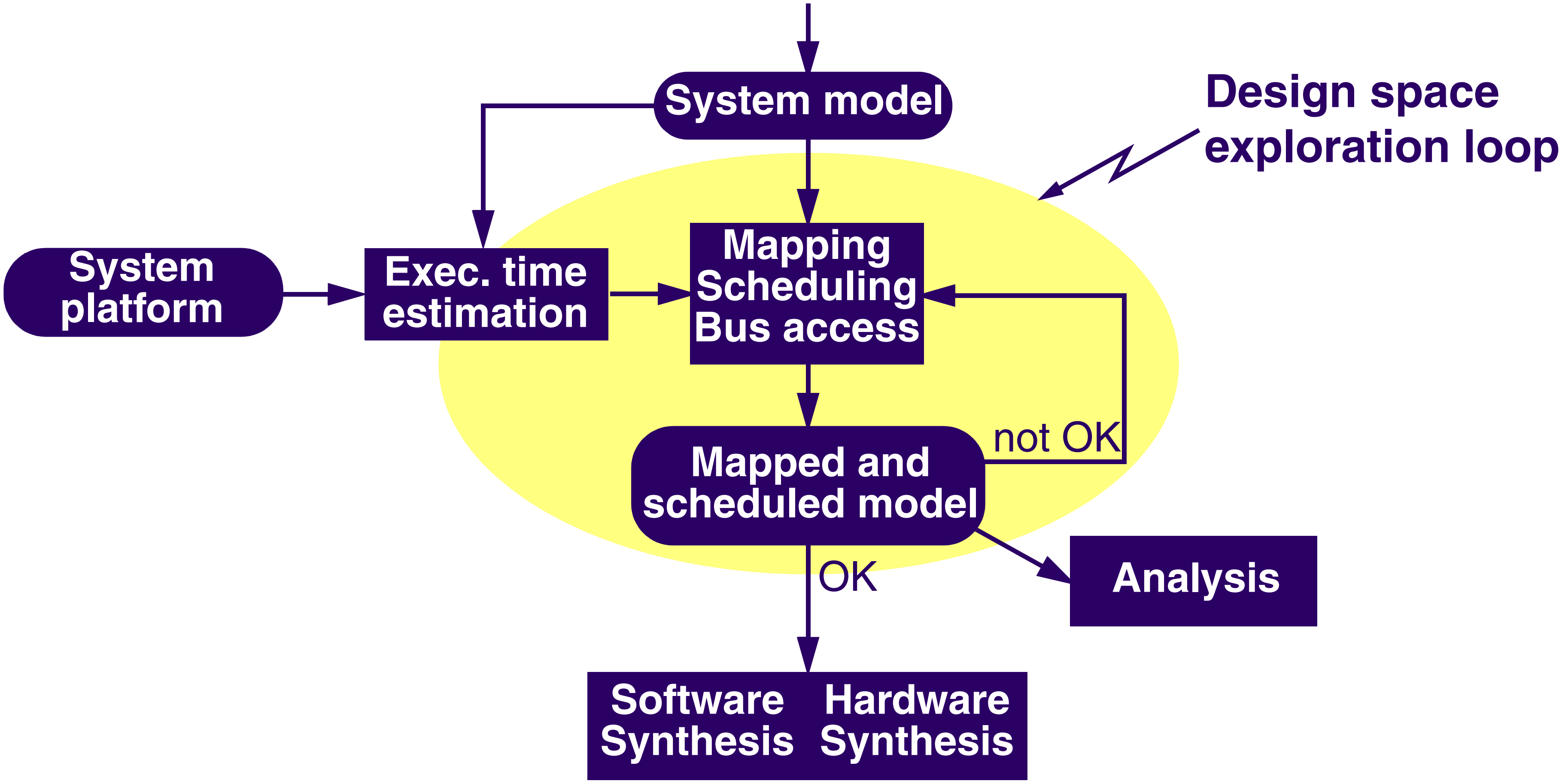
Why?

- **Physical constraints**
  - **Operation close to sensor;**
- **Modularity constraints**
- **Safety Constraints**
- **Performance**

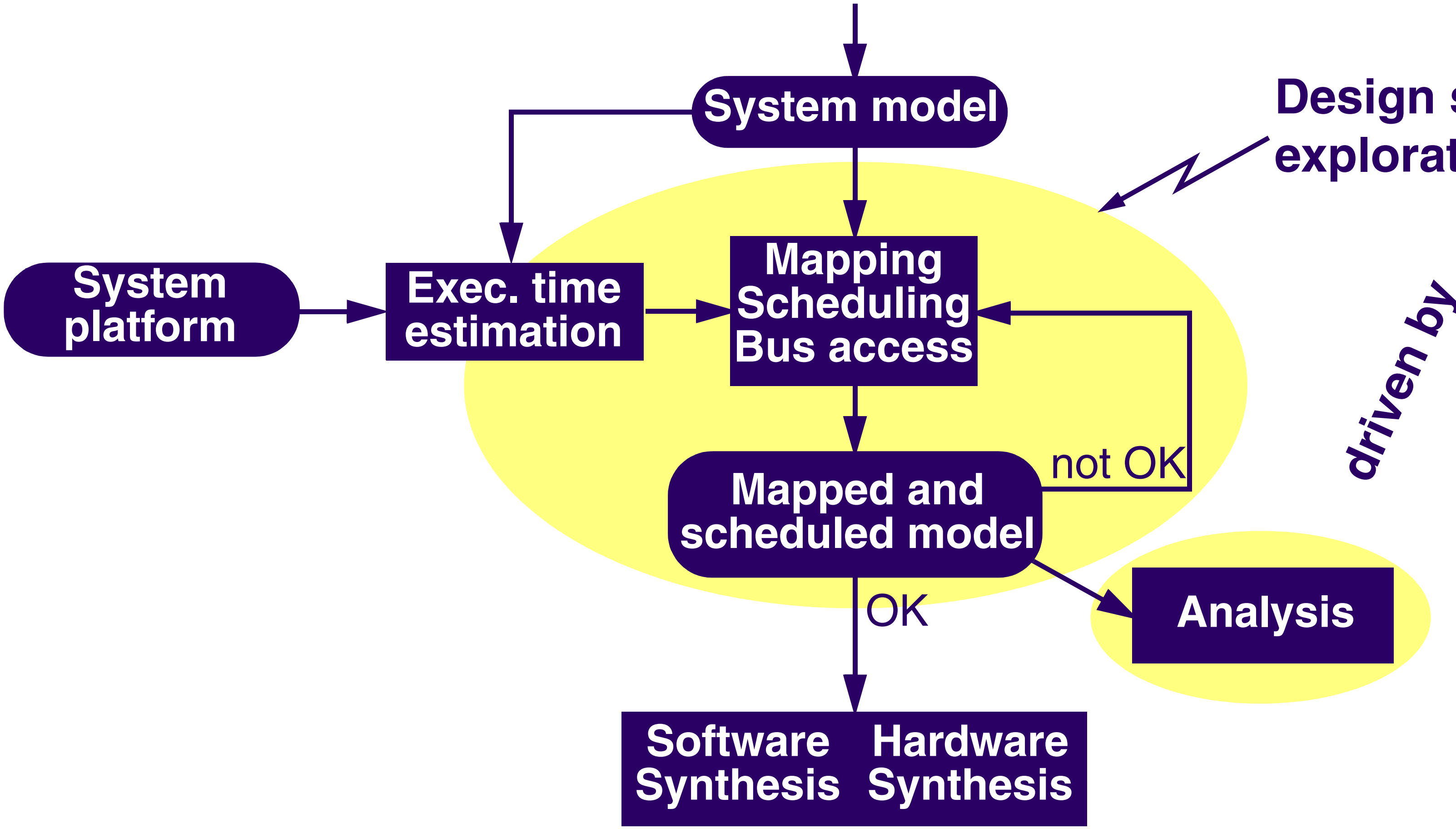
System Level Design Flow



System Level Design Flow

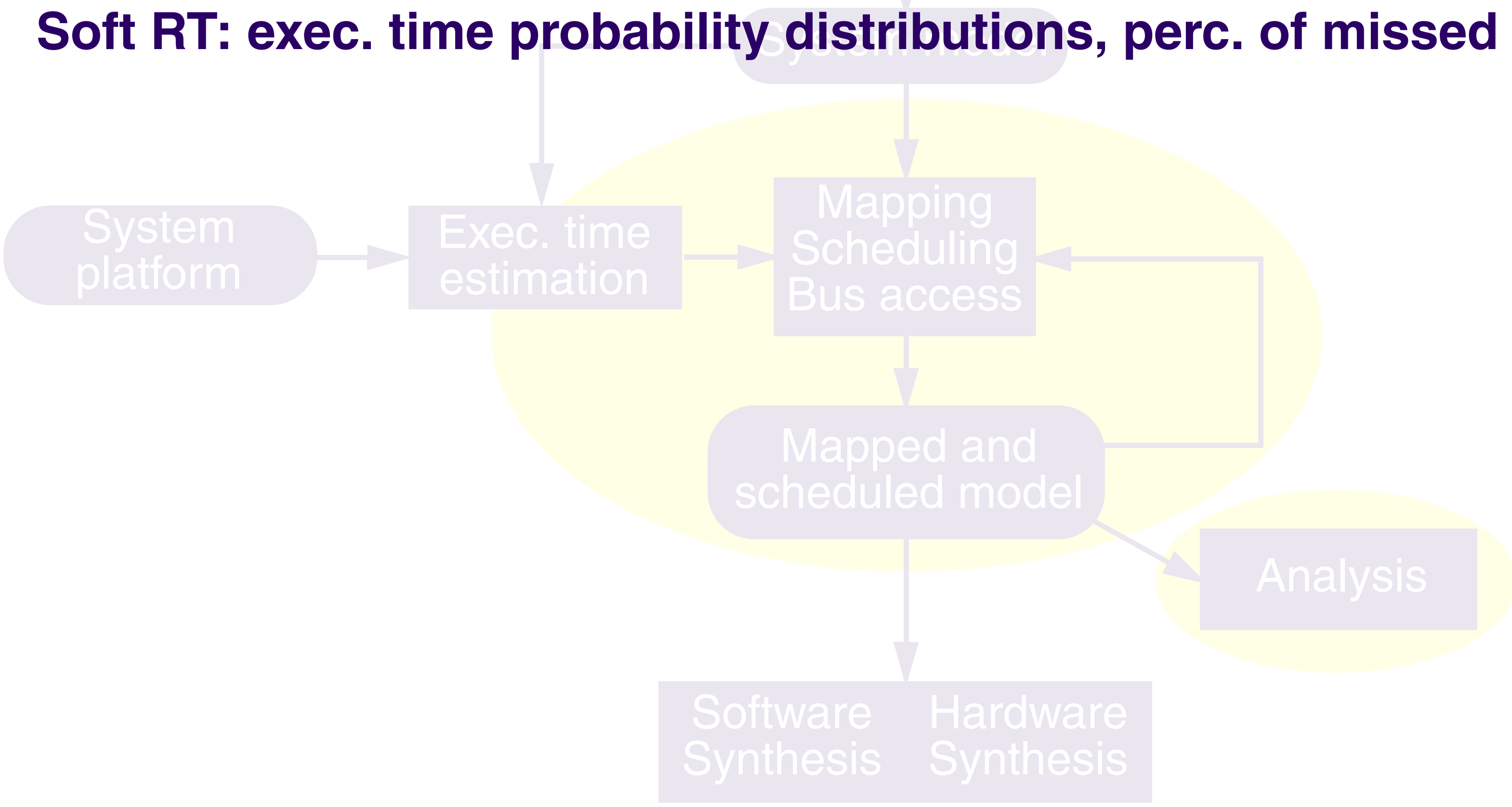


System Level Design Flow



System Level Design Flow

- **Category of applications:**
  - **Hard RT: WCET, hard deadlines to be satisfied**
  - **Soft RT: exec. time probability distributions, perc. of missed deadlines**



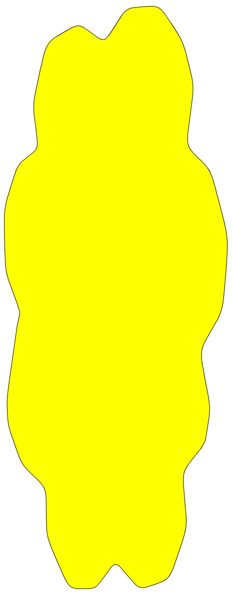
System Level Design Flow

- **Category of applications:**
  - **Hard RT: WCET, hard deadlines to be satisfied**
  - **Soft RT: exec. time probability distributions, perc. of missed deadlines**
- **Constraints:**
  - **time constraints (hard, soft)**
  - **cost (amount of resources)**
  - **fault tolerance**

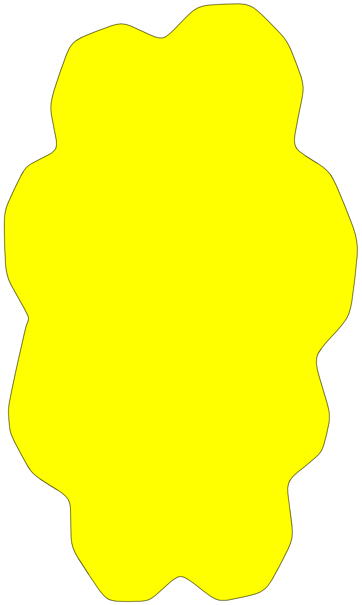
System Level Design Flow

- **Category of applications:**
  - **Hard RT: WCET, hard deadlines to be satisfied**
  - **Soft RT: exec. time probability distributions, perc. of missed deadlines**
- **Constraints:**
  - **time constraints (hard, soft)**
  - **cost (amount of resources)**
  - **fault tolerance**
- **Optimization problems:**
  - **schedule generation/priority assignment**
  - **task mapping**
  - **bus configuration**
  - **scheduling policy assignment**
  - **fault tolerance policy assignment**

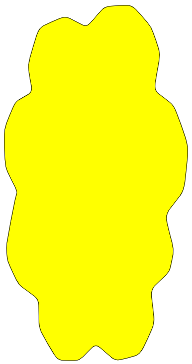
- **An application is modelled as a set of task graphs:**



Γ_1
Period: $T_{\Gamma 1}$
Deadline: $D_{\Gamma 1}$



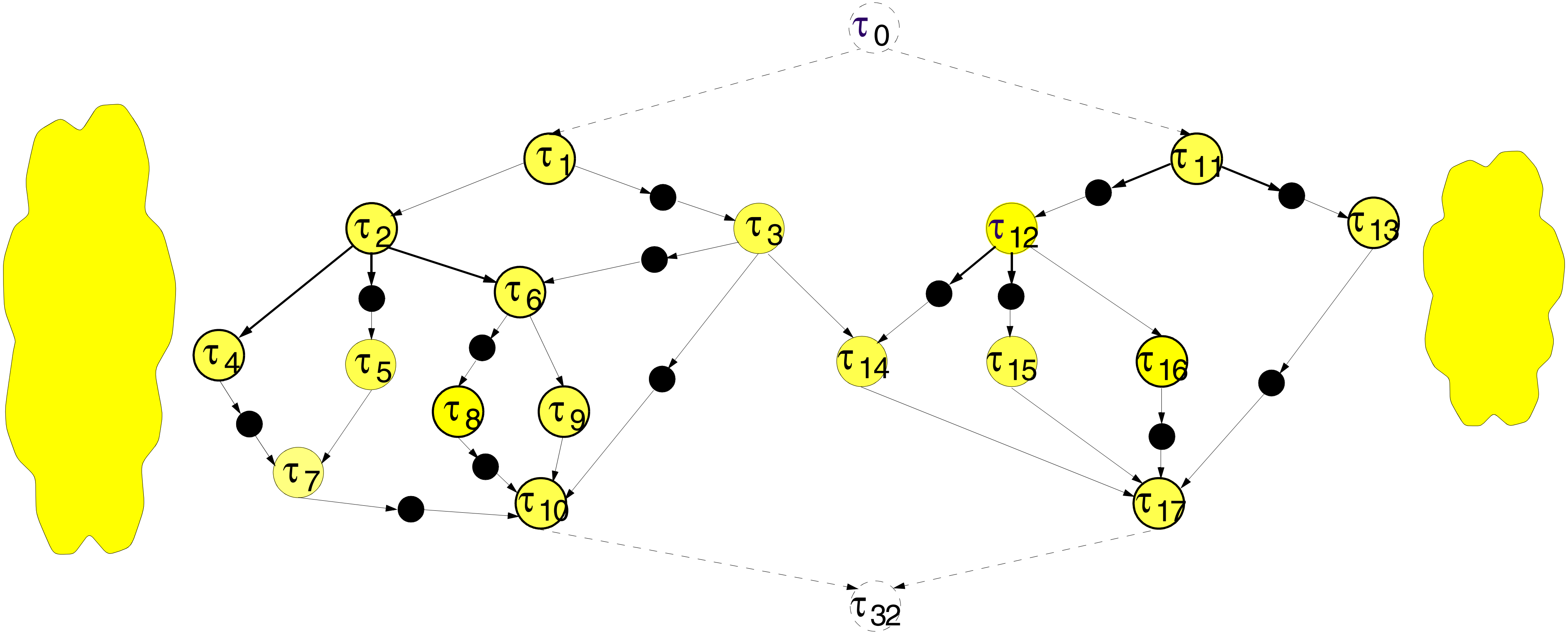
Γ_2
Period: $T_{\Gamma 2}$
Deadline: $D_{\Gamma 2}$



Γ_3
Period: $T_{\Gamma 3}$
Deadline: $D_{\Gamma 3}$

Application Model

☞ **An application is modelled as a set of task graphs:**



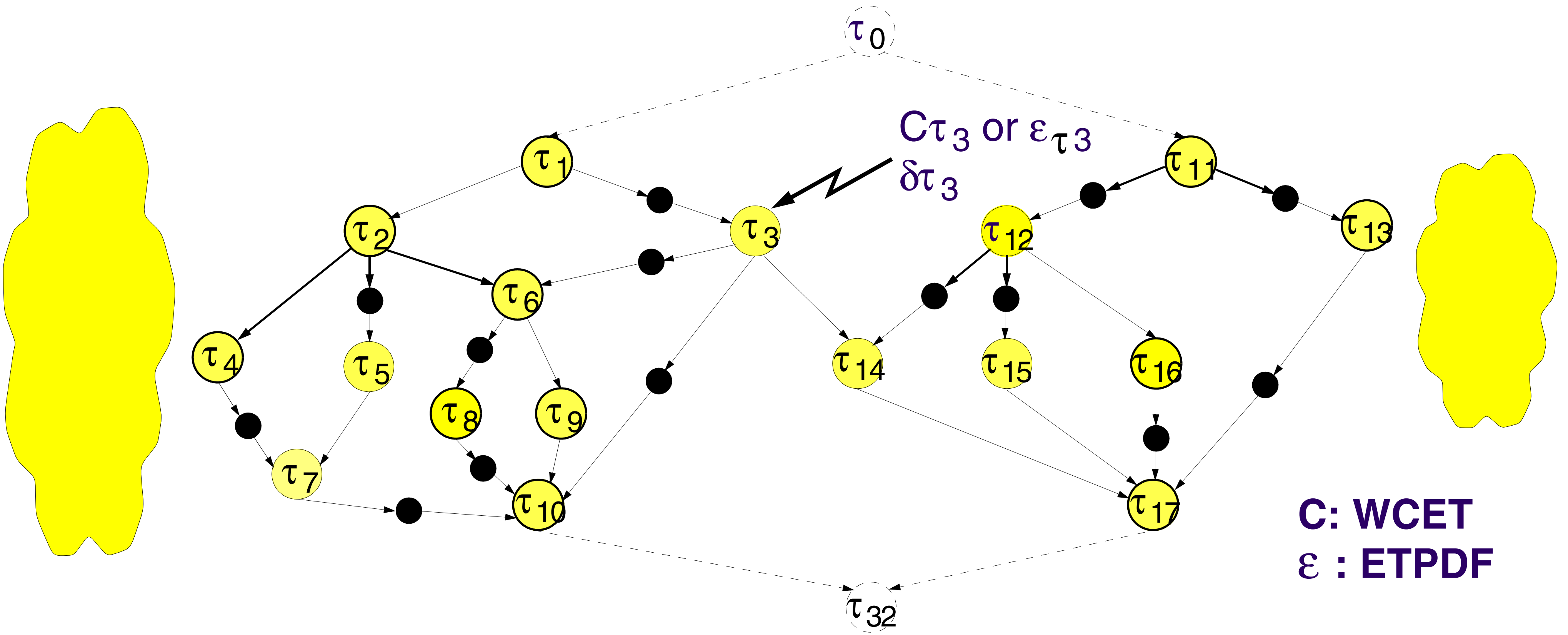
Γ_1
Period: $T_{\Gamma 1}$
Deadline: $D_{\Gamma 1}$

Γ_2
Period: $T_{\Gamma 2}$
Deadline: $D_{\Gamma 2}$

Γ_3
Period: $T_{\Gamma 3}$
Deadline: $D_{\Gamma 3}$

Application Model

☞ **An application is modelled as a set of task graphs:**



C: WCET
ε : ETPDF

Γ_1
Period: $T_{\Gamma 1}$
Deadline: $D_{\Gamma 1}$

Γ_2
Period: $T_{\Gamma 2}$
Deadline: $D_{\Gamma 2}$

Γ_3
Period: $T_{\Gamma 3}$
Deadline: $D_{\Gamma 3}$

Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization

Distributed Hard Real-Time Systems

- **Given:**
  - **WCET and BCET of each task**
  - **Dimension of messages**
  - **Deadlines for tasks/task graphs**

☞ **All deadlines have to be satisfied!**

Distributed Hard Real-Time Systems

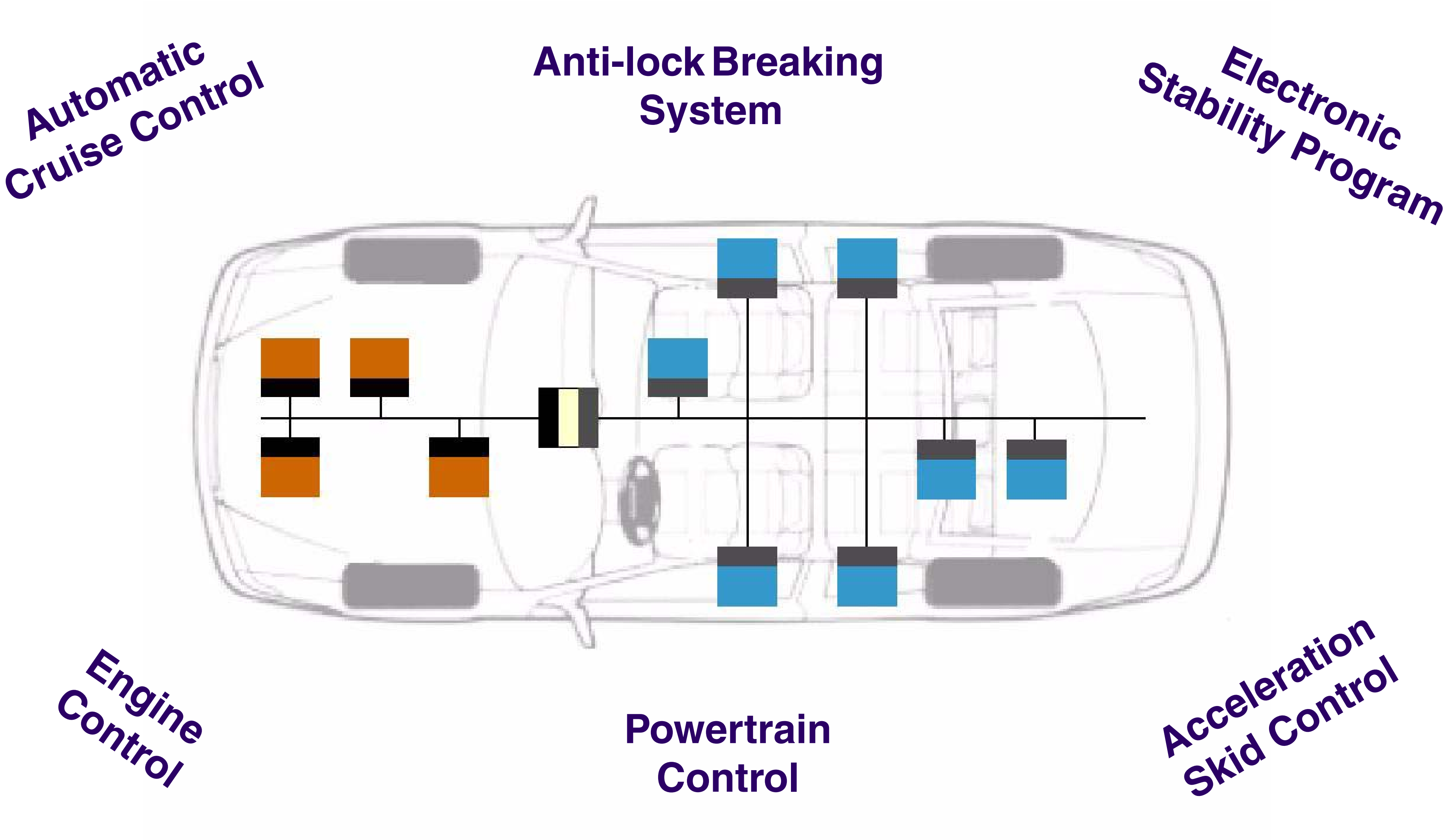
- Given:
  - WCET and BCET of each task
  - Dimension of messages
  - Deadlines for tasks/task graphs

☞ All deadlines have to be satisfied!

- Particular issues taken into consideration
  - Heterogeneous nature of the system
  - Various communication protocols

☞ Try to satisfy constraints at low cost!

Application Area



Heterogeneous Distributed Embedded Systems

- **RT Design Approach**

- **Network Protocol**

Heterogeneous Distributed Embedded Systems

- **RT Design Approach**
  - Time triggered
  - Event triggered

- **Network Protocol**

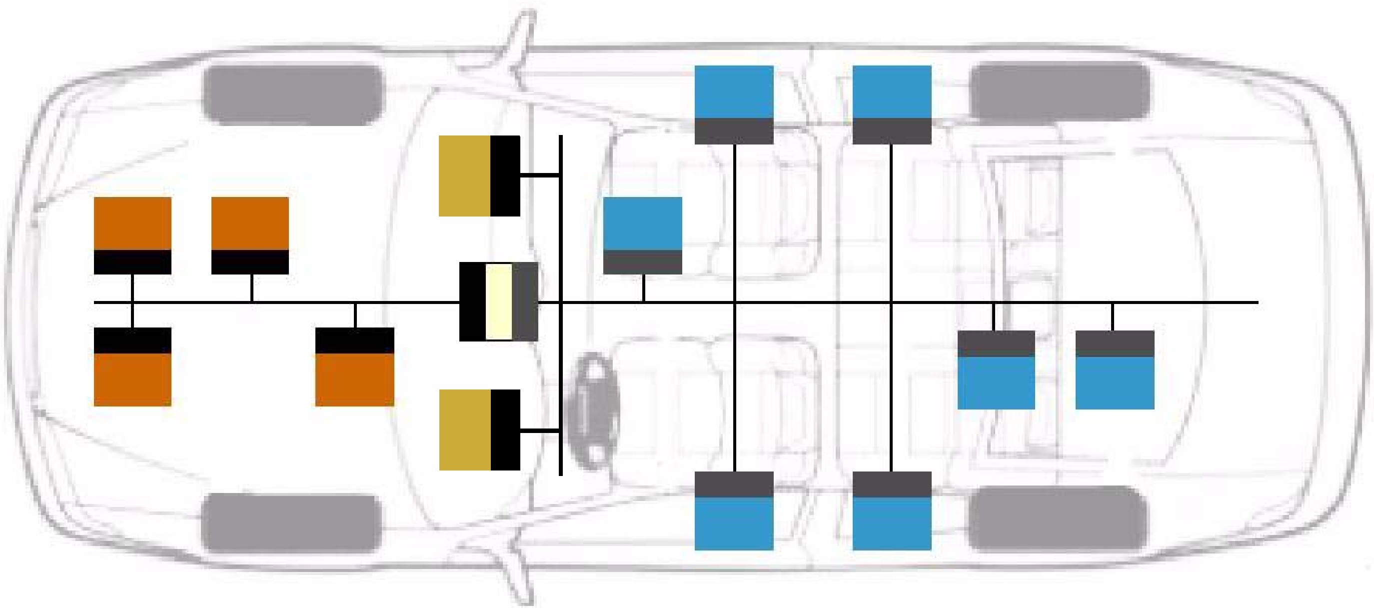
Heterogeneous Distributed Embedded Systems

- **RT Design Approach**
  - Time triggered
  - Event triggered

- **Network Protocol**
  - Static
  - Dynamic

Heterogeneous Distributed Embedded Systems

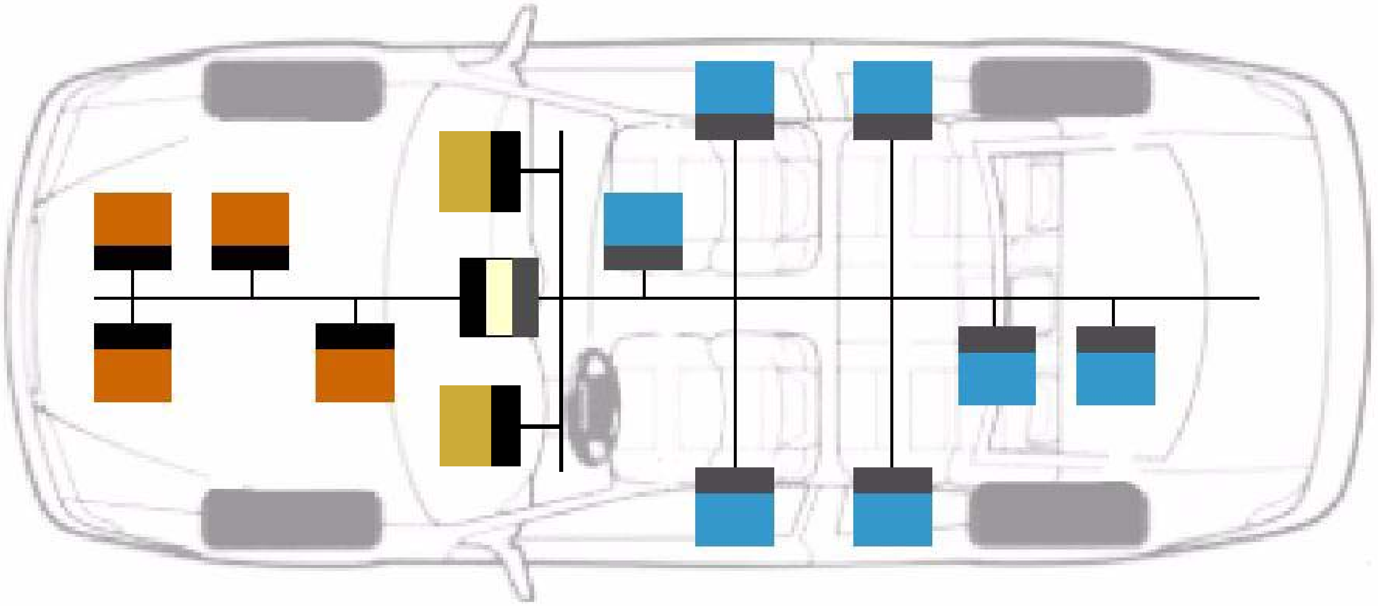
☞ **Heterogeneous Nature of Implemented Functions**



Heterogeneous Distributed Embedded Systems

Engine Control

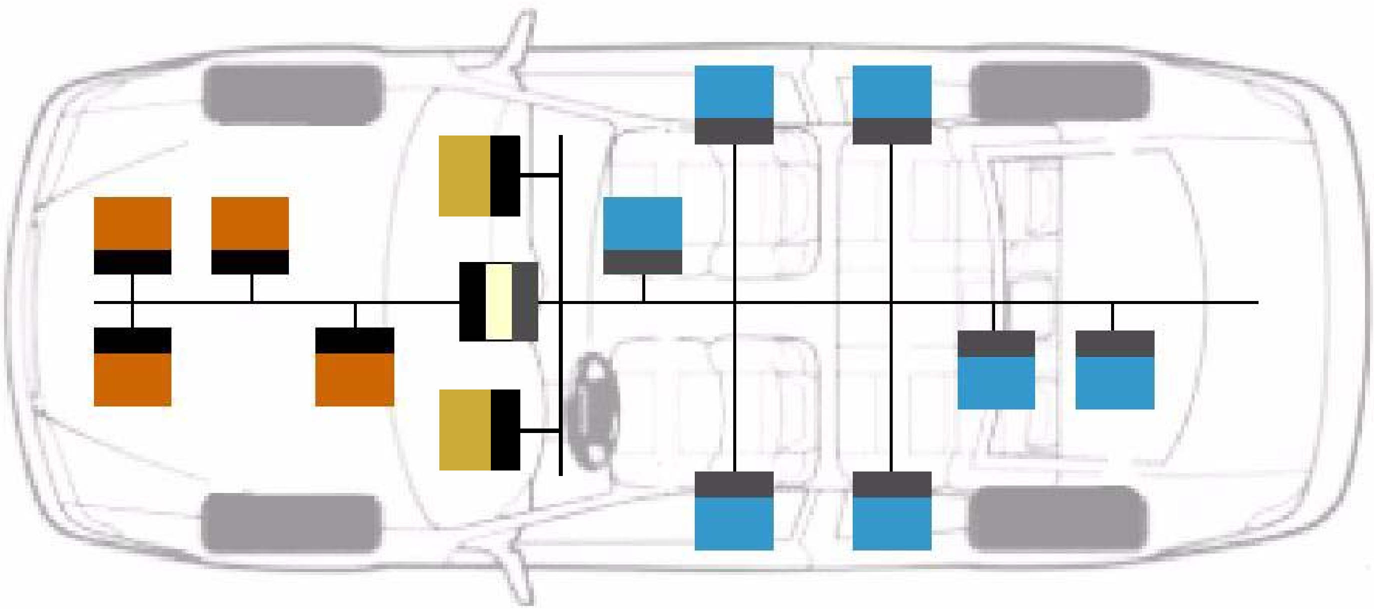
- **hard real-time**



Heterogeneous Distributed Embedded Systems

Engine Control
- hard real-time

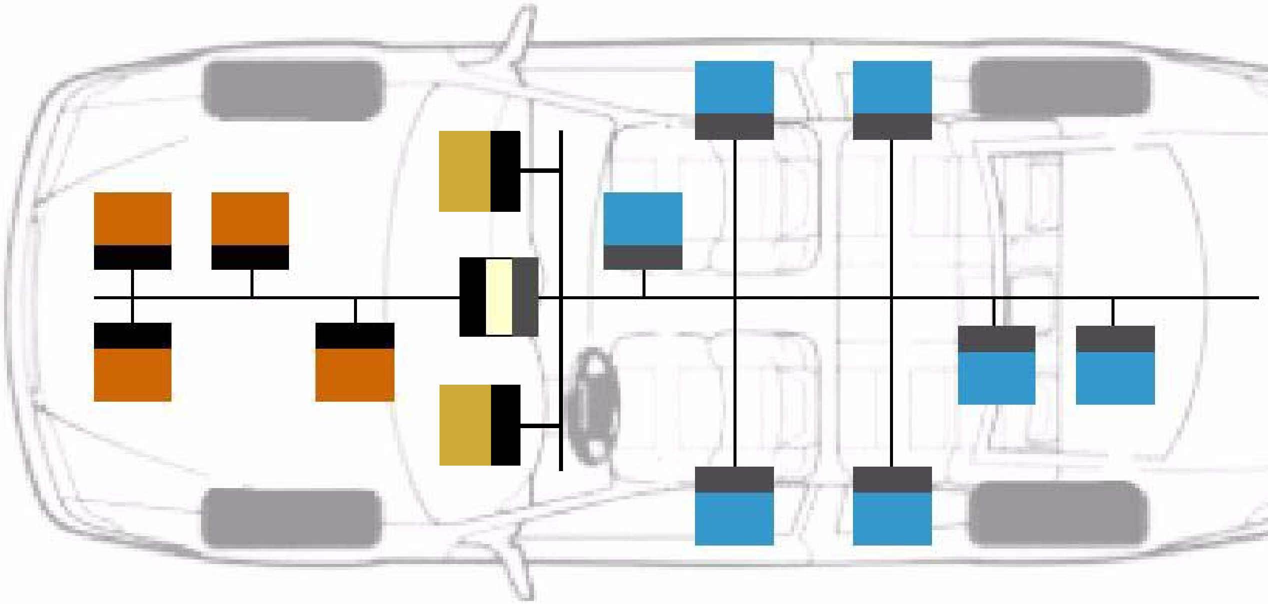
Power Train (break-by-wire, ABS)
- hard real-time
- highly safety-critical



Heterogeneous Distributed Embedded Systems

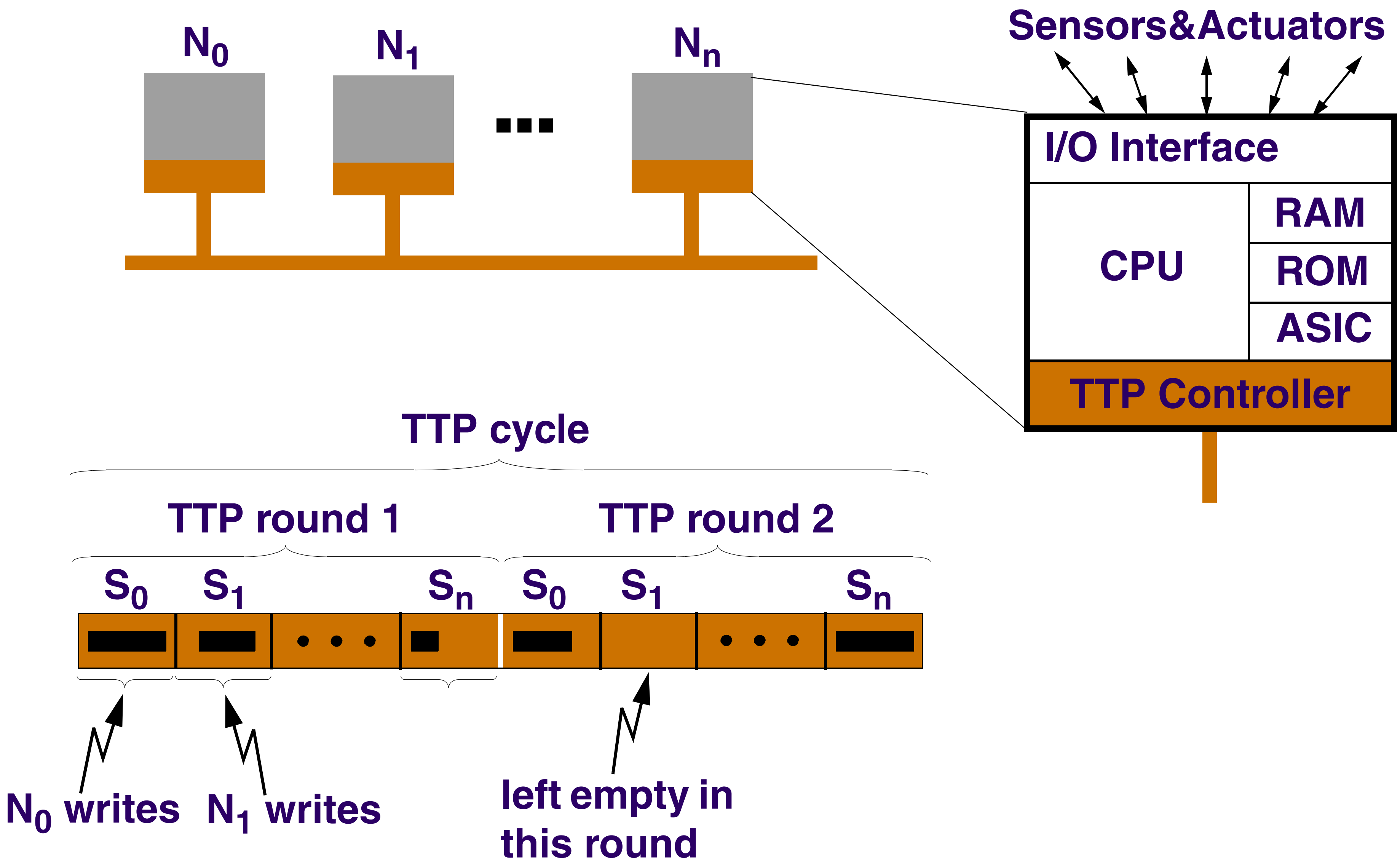
Engine Control
- hard real-time

Power Train (break-by-wire, ABS)
- hard real-time
- highly safety-critical

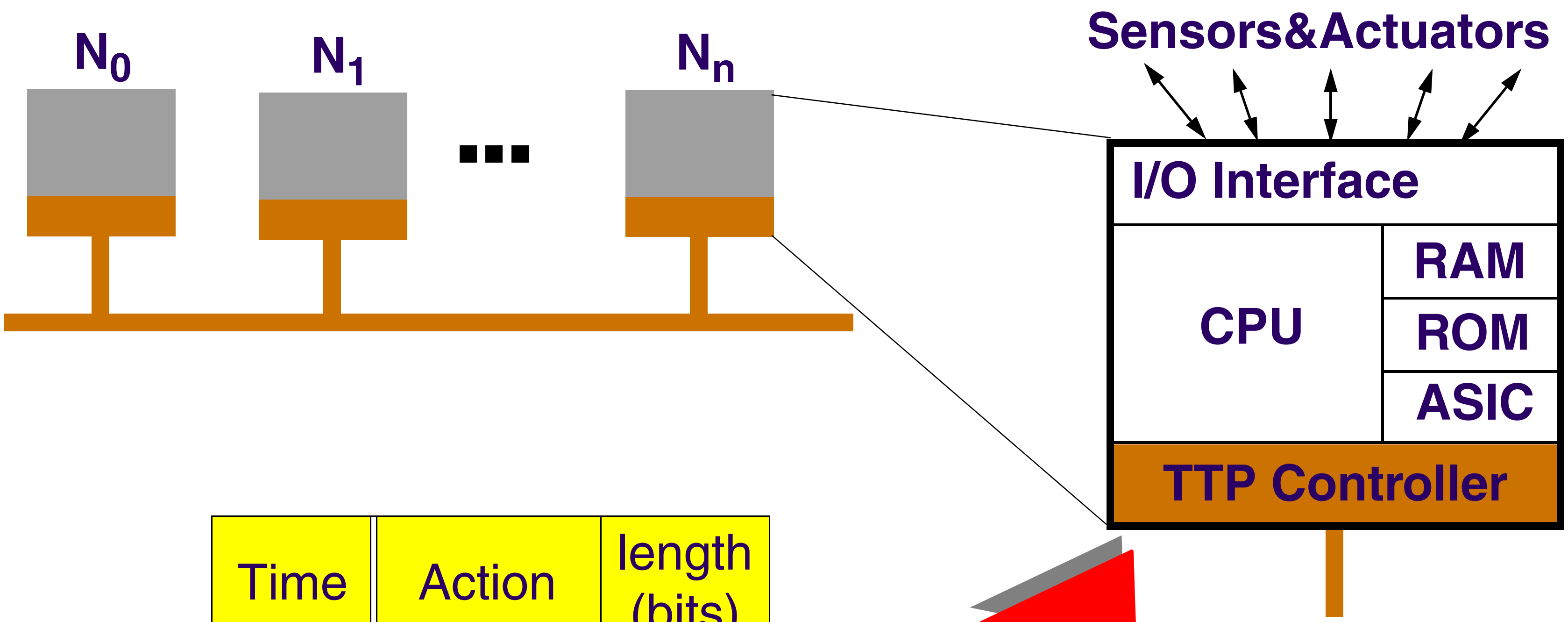


Air Conditioning
- soft real-time

Static Communication: TTP



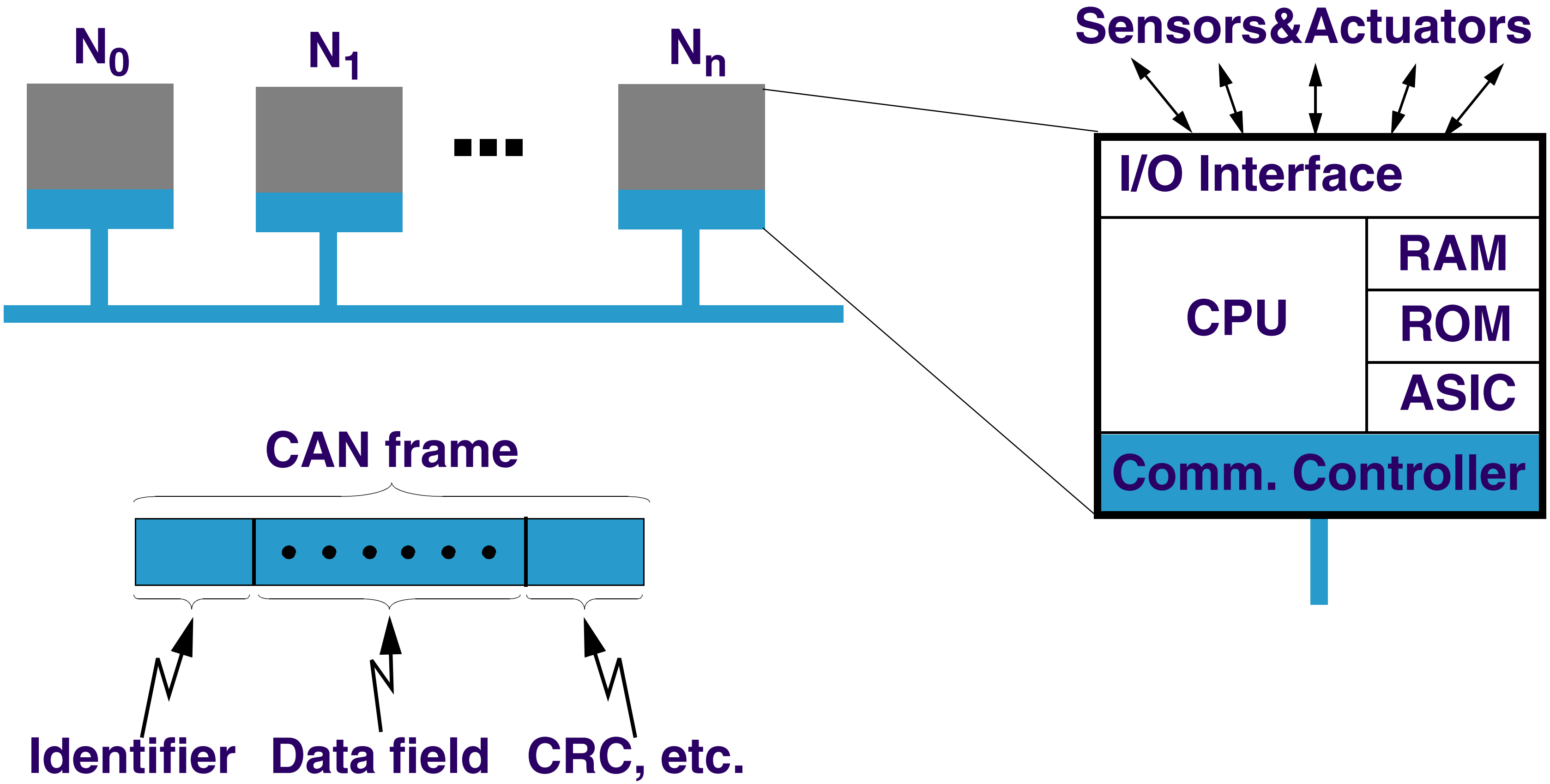
Static Communication: TTP



Over a TTP cycle

Time	Action	length (bits)
t_1	rd-frame	16
t_2	wr-frame	32
t_3	rd-frame	32
...	...	...
t_x	wr-frame	16

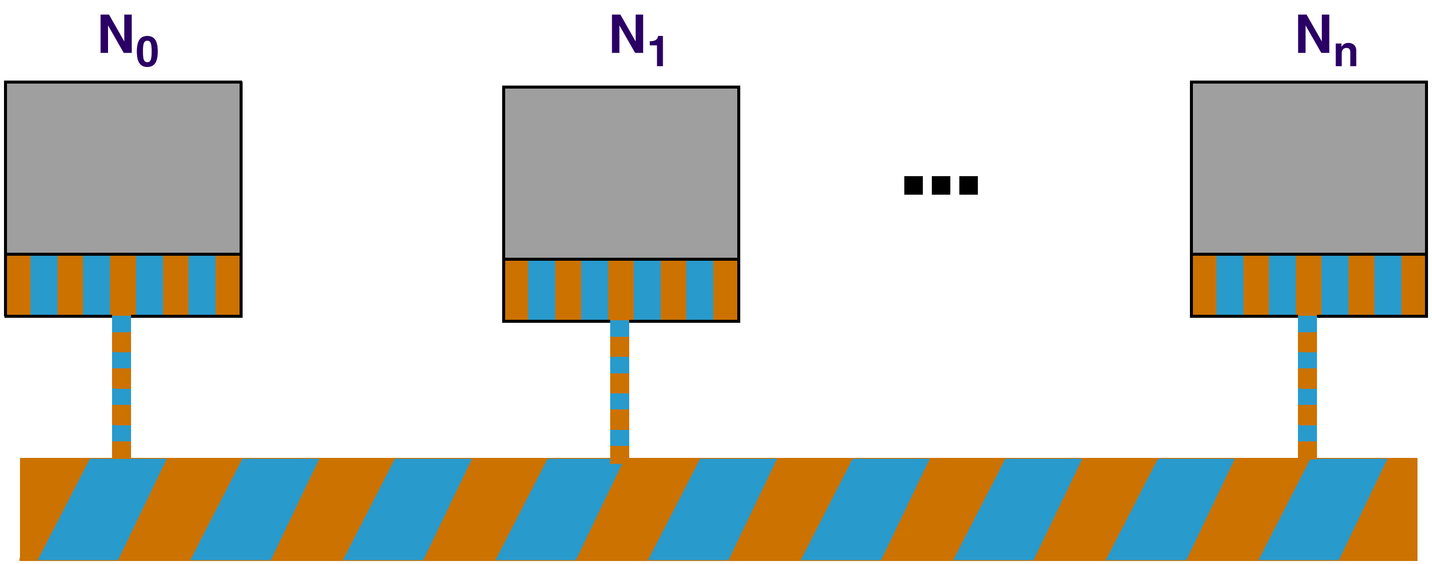
Dynamic Communication: CAN

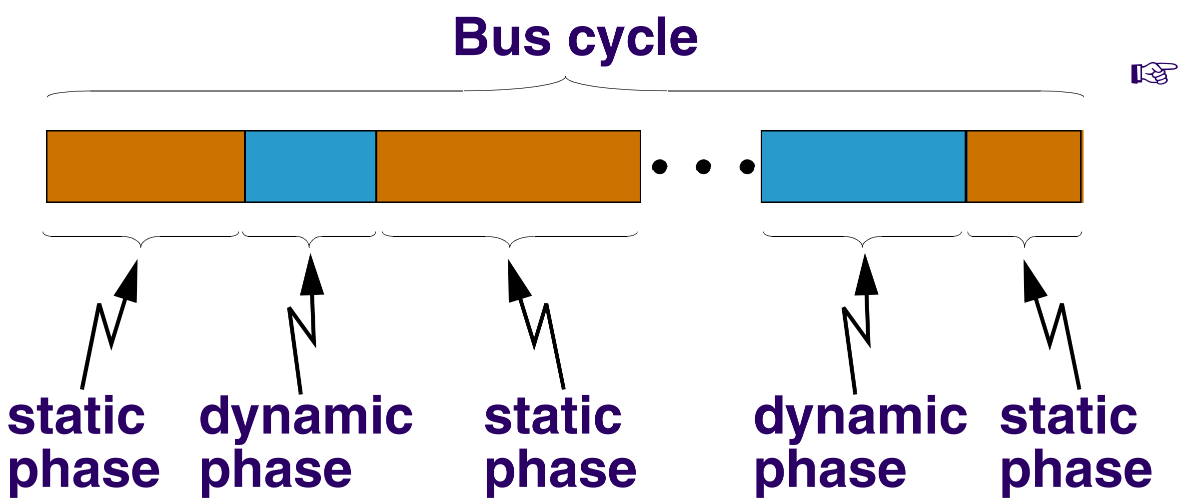


- **Priority bus with collision avoidance mechanism:**

  The node that transmits the highest priority frame wins the contention.

Mixed Communication: UCM

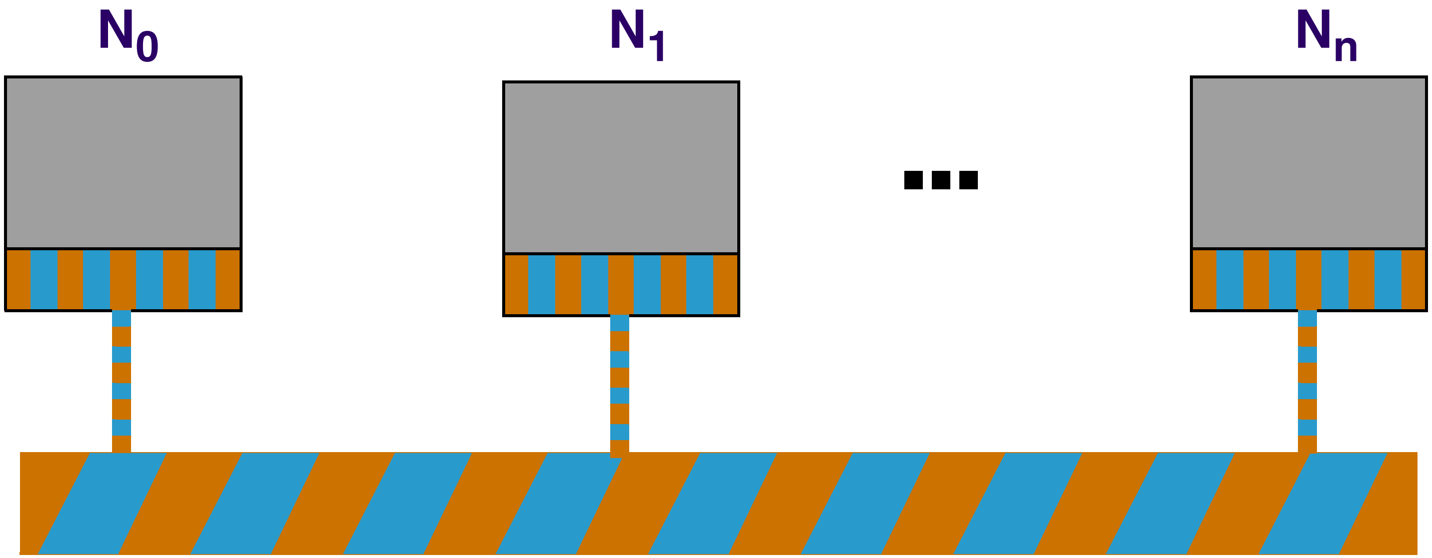


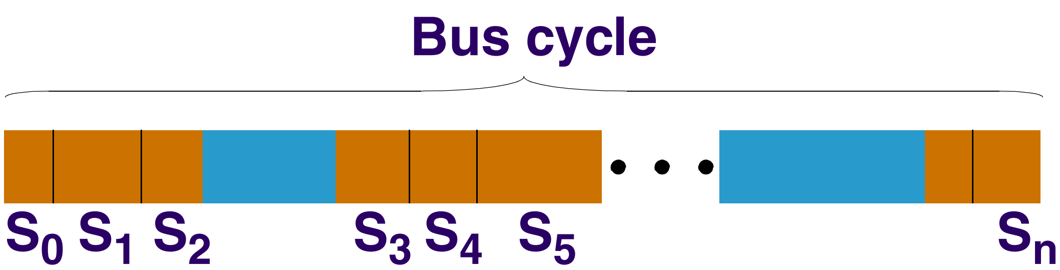


☞ **Universal Communication Model**

- **Static phase: TDMA**
- **Dynamic phase: CAN**

Mixed Communication: UCM

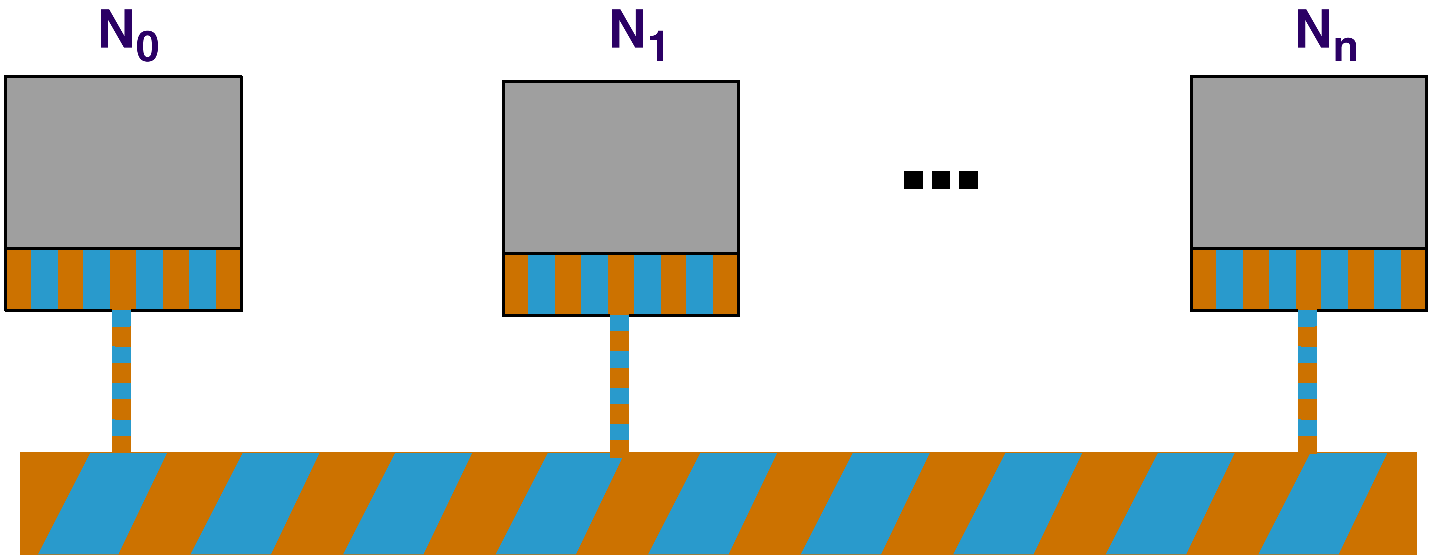


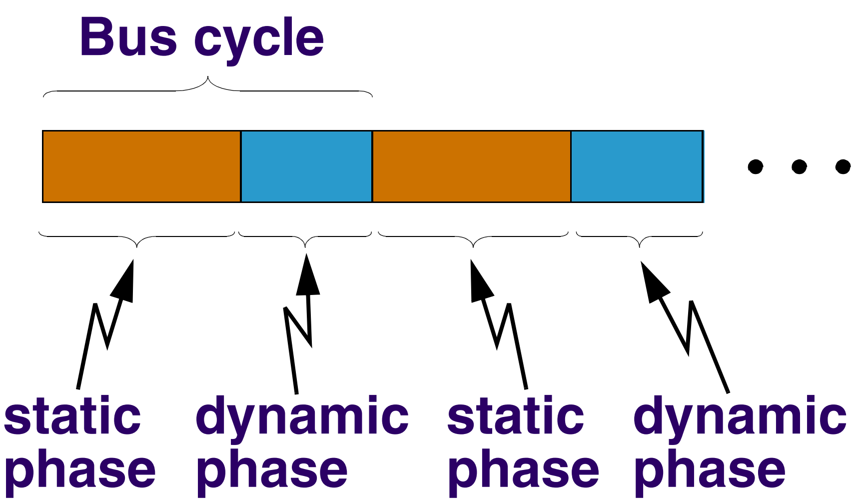


☞ **Universal Communication Model**

- **Static phase: TDMA**
- **Dynamic phase: CAN**

Mixed Communication: FlexRay

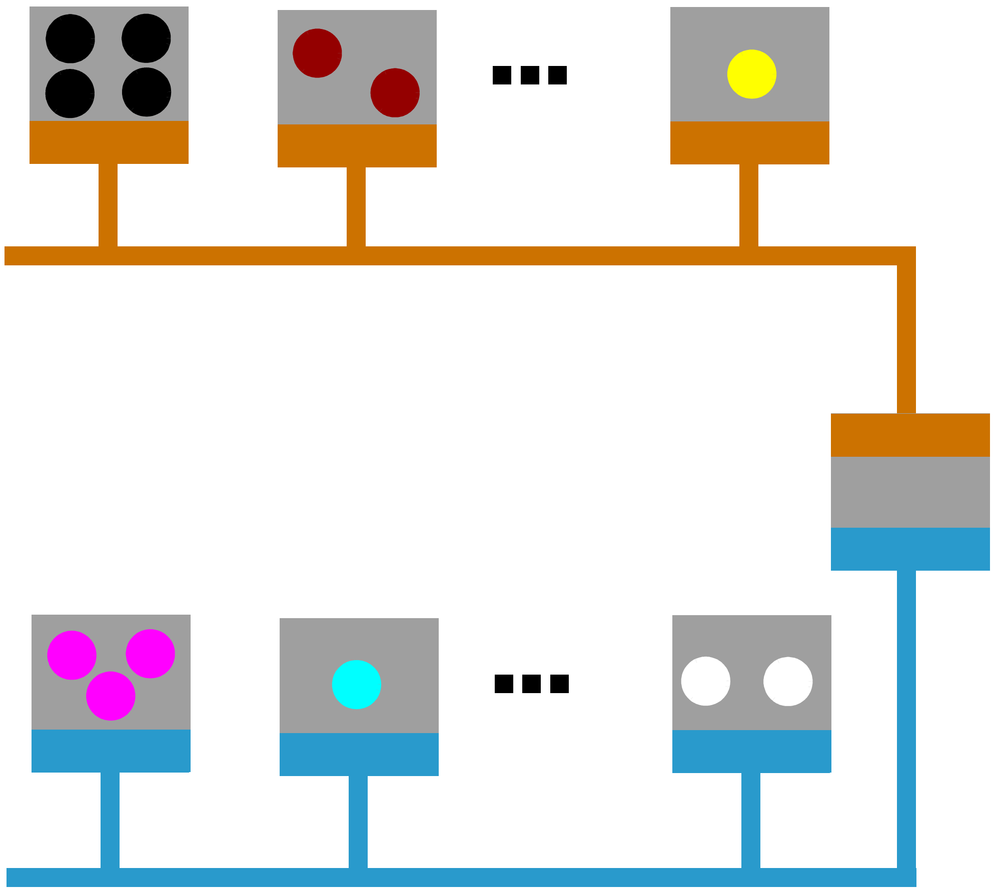




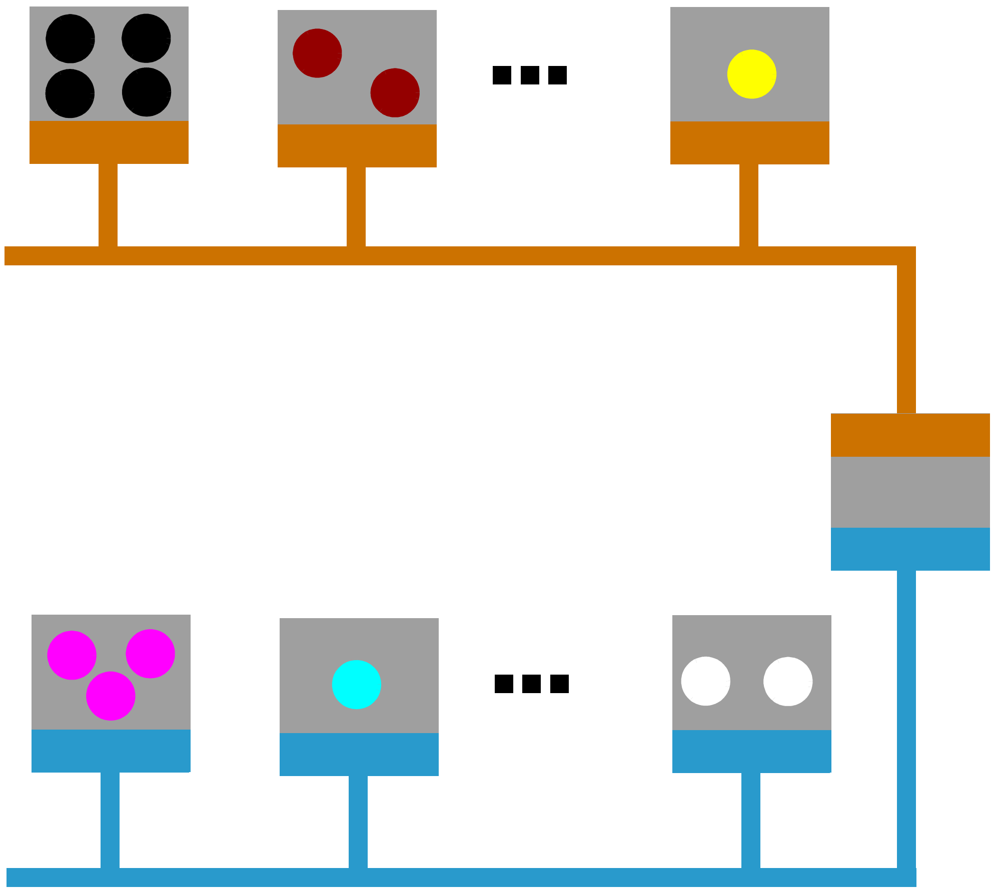
- FlexRay
  - Static phase: TDMA
  - Dynamic phase: Flexible TDMA

The Traditional Approach

One node ⟷ One function



The Traditional Approach



One node ⟷ One function

- Purchase node&function and integrate into system
- Sophistication grew quickly

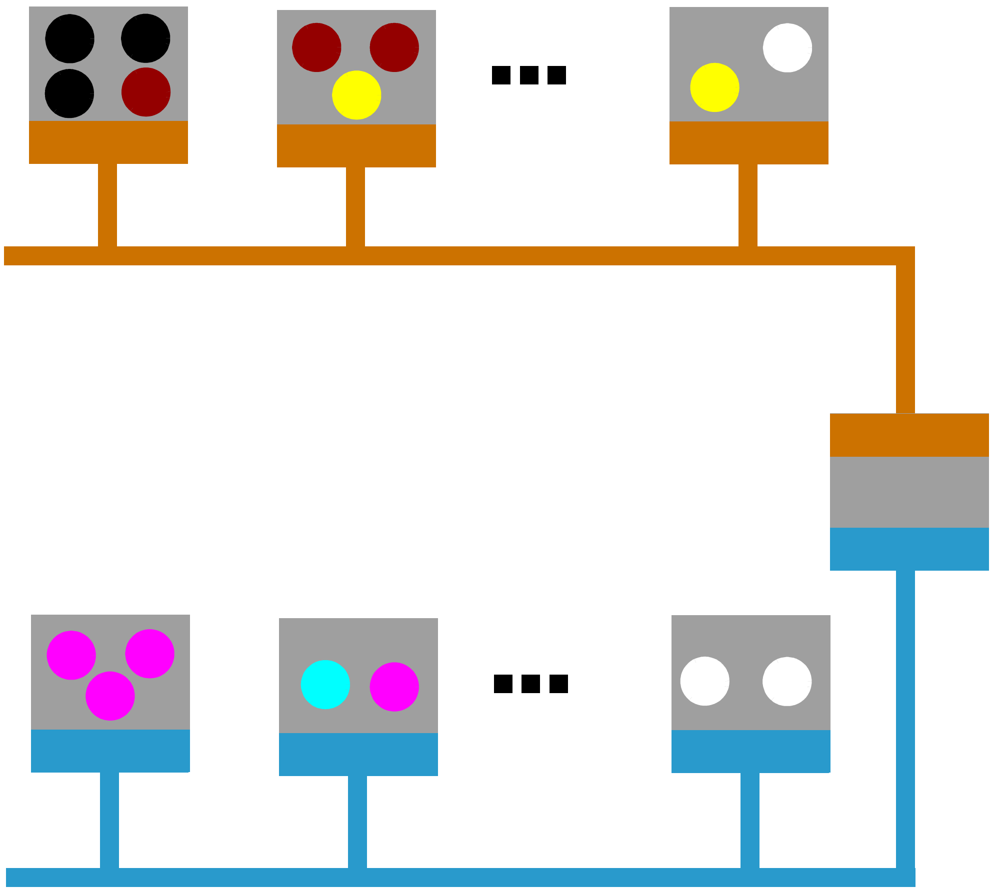
↓

More then 100 nodes

↓

Resources have to be used more efficiently

What Comes



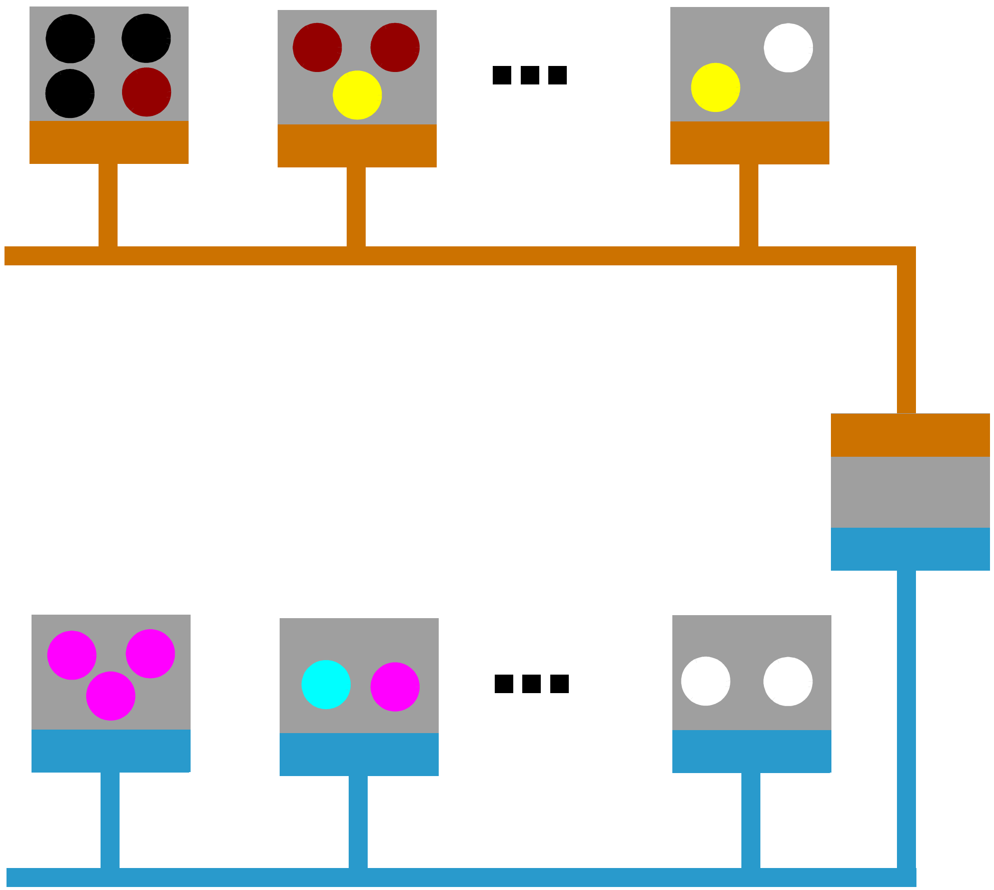
One node ⟷ Several functions

Several nodes ⟷ One function

Flexibility!

- Reduce cost
- Improve resource usage
- Function close to sensor

What Comes



One node ⟷ Several functions

Several nodes ⟷ One function

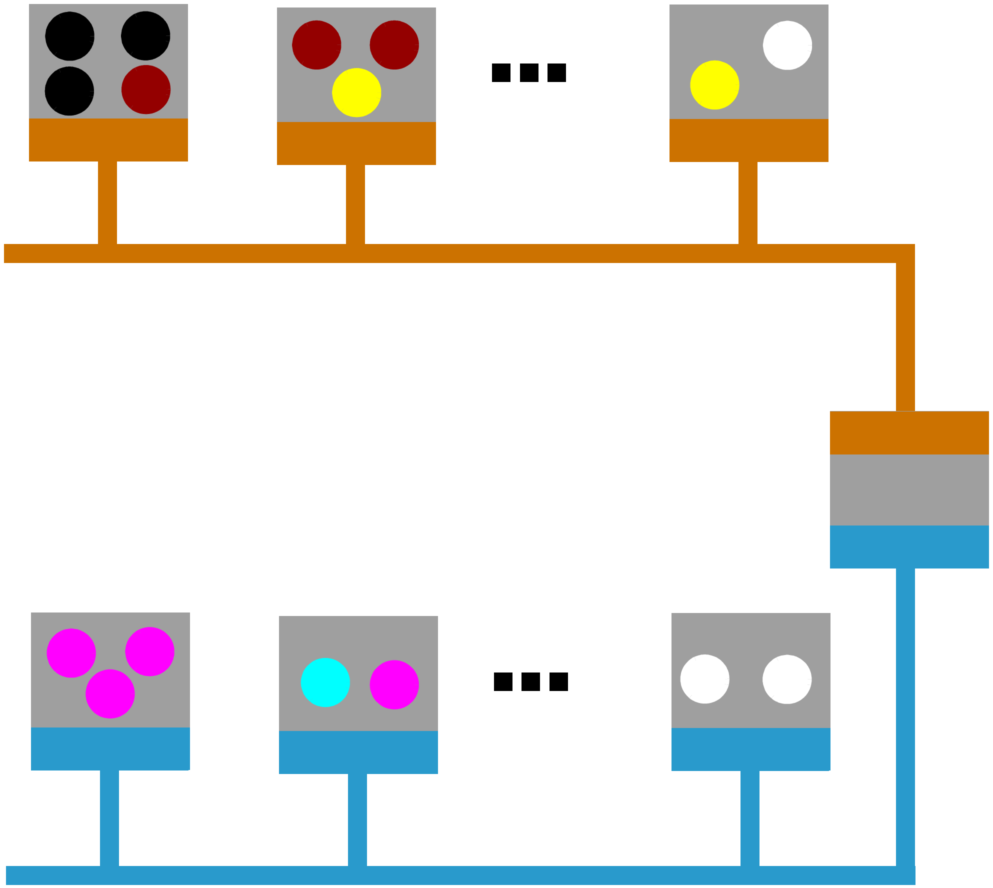
Flexibility!

- Reduce cost
- Improve resource usage
- Function close to sensor

☞ Needed:

- Middleware software
- New analysis
- New system optimization

What Comes

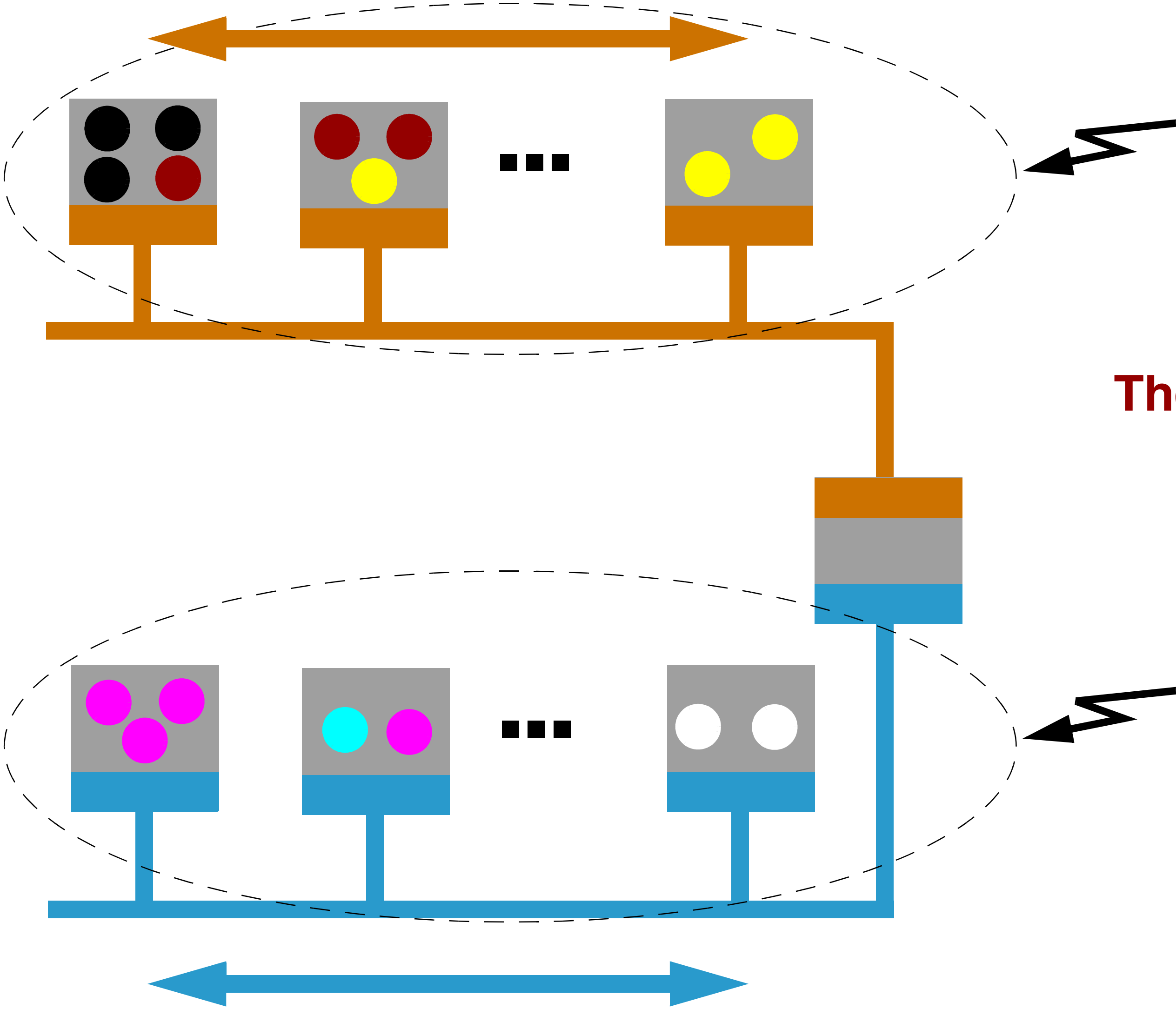


Navet et al.

Trends in Automotve Communication Systems

Proceedings of the IEEE, June 2005.

Isolated Domains



Time triggered cluster:

- TT tasks
- Static communication

The different domains are isolated.

Event triggered cluster:

- ET tasks
- Dynamic communication

Interacting Domains

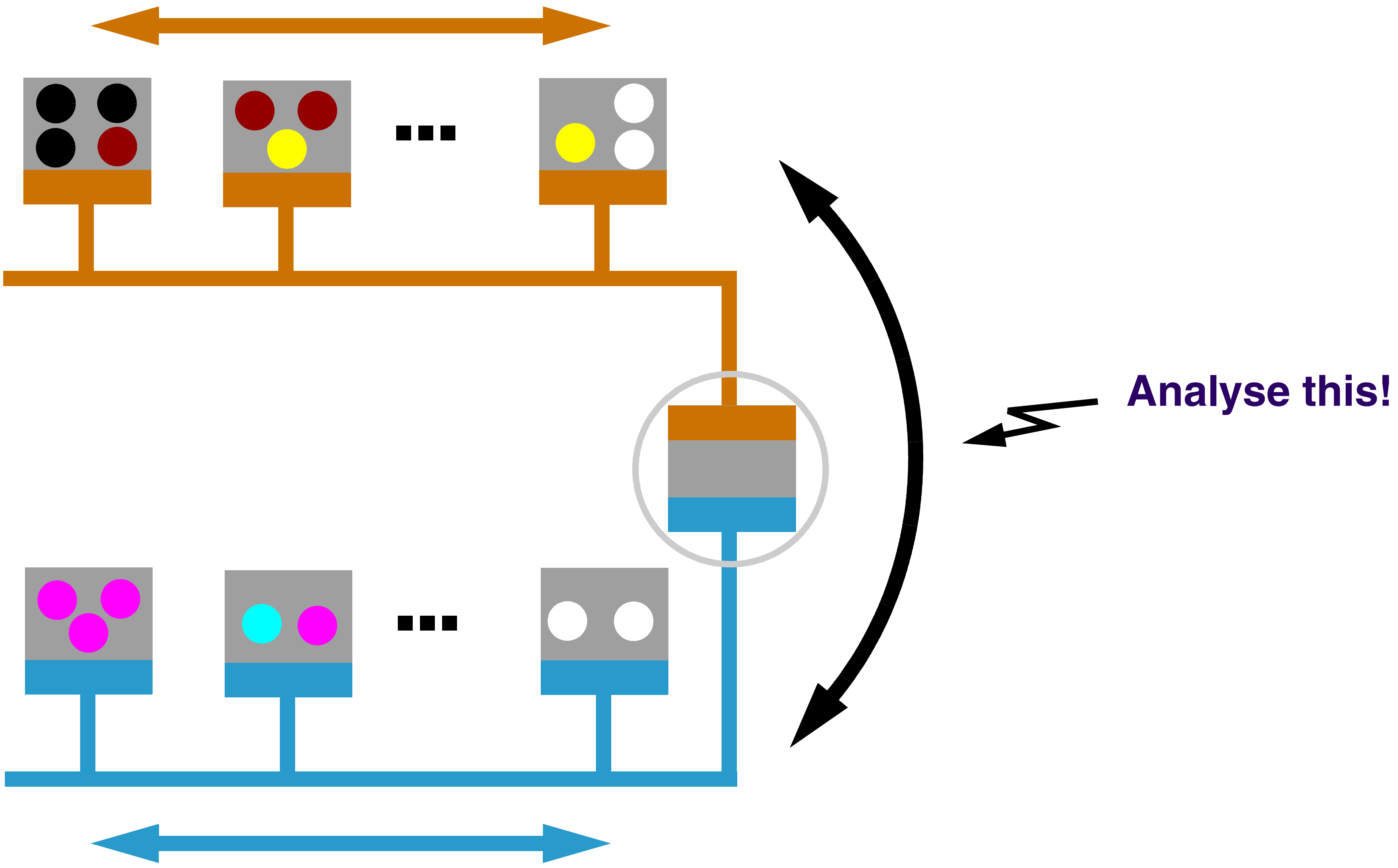
The TT & ET, Static & Dynamic domains are interacting because:

- **They share resources (node, bus)**

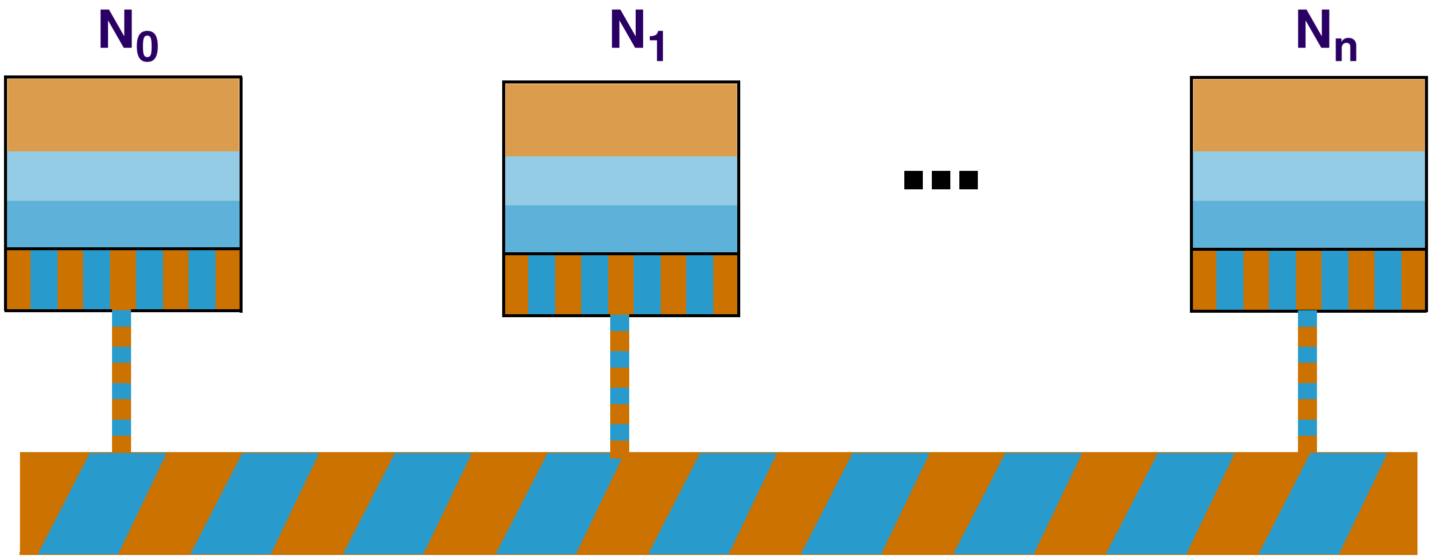
and/or

- **They communicate**

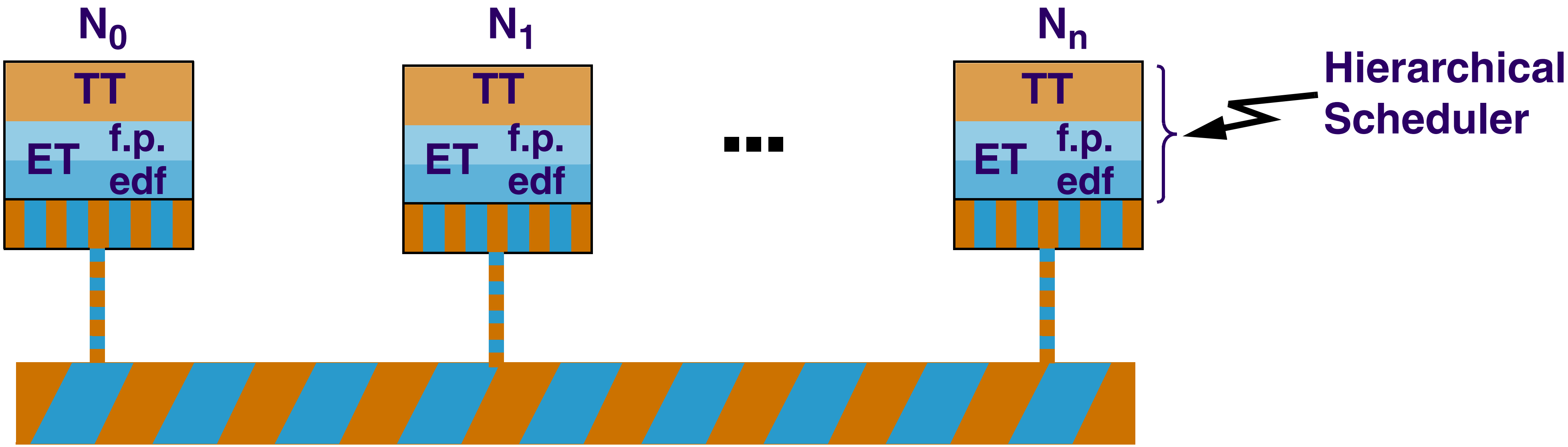
Multi-cluster Heterogeneous Distributed Architecture



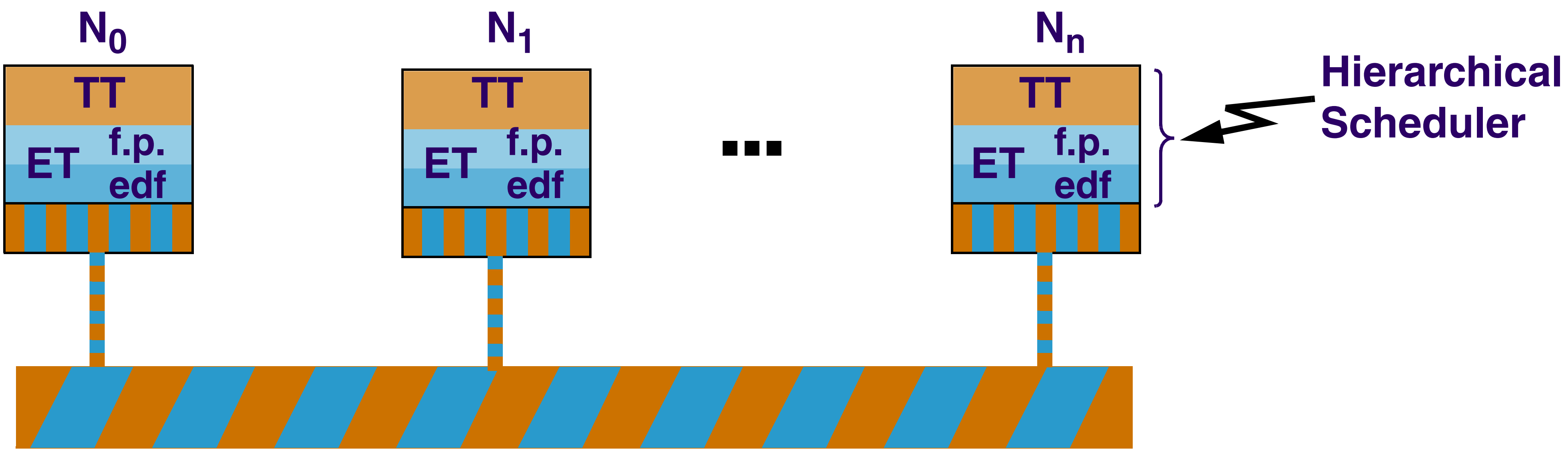
Single-cluster Heterogeneous Distributed Architecture

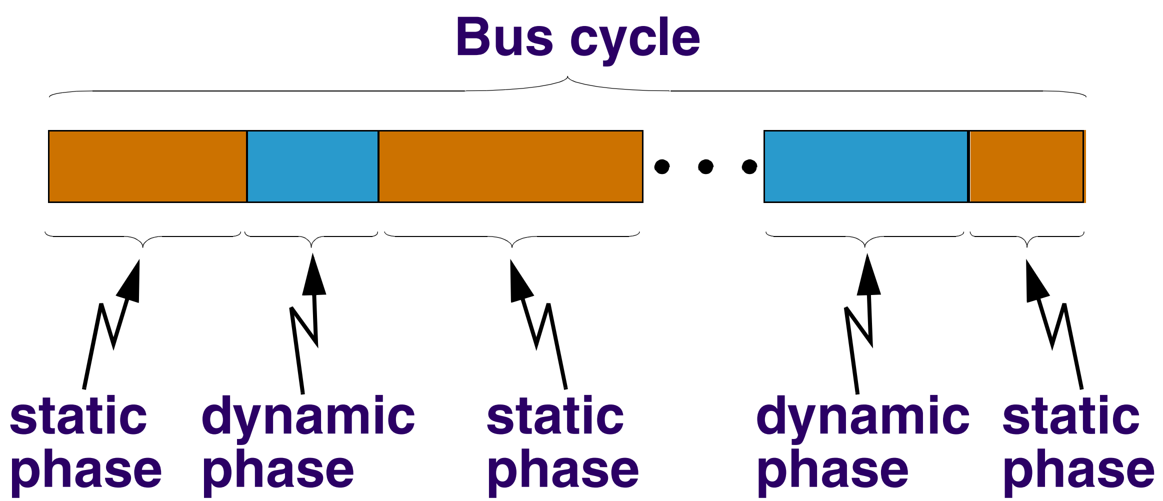


Single-cluster Heterogeneous Distributed Architecture



Single-cluster Heterogeneous Distributed Architecture



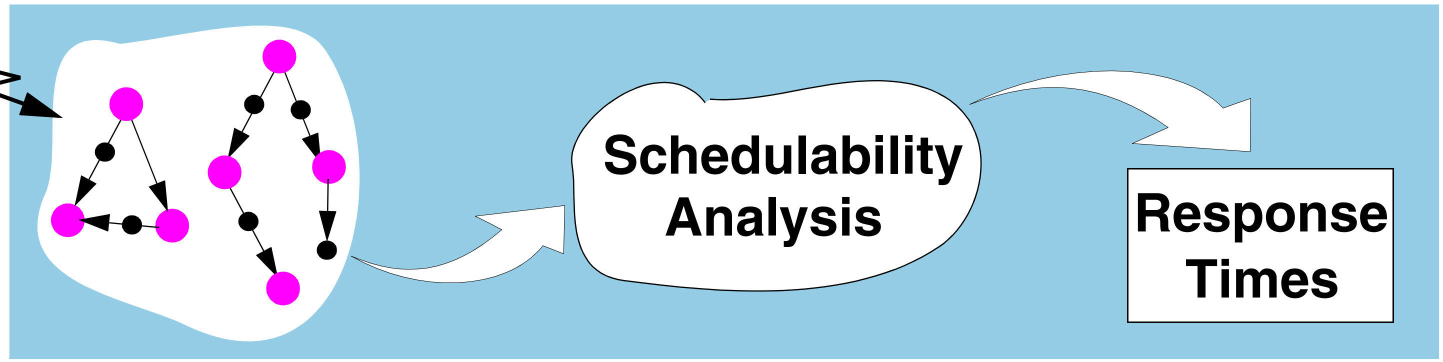


- UCM (TTP&CAN)
- FlexRay

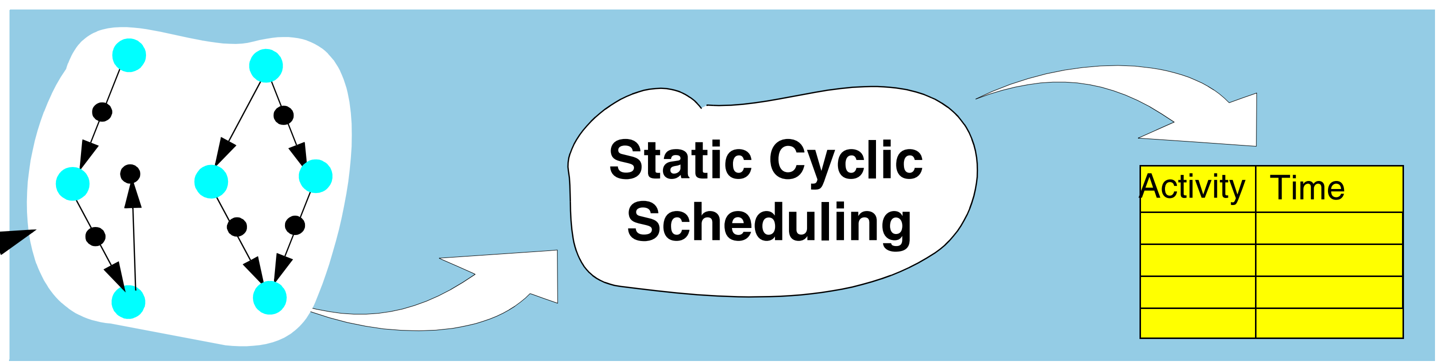
Analysis

☞ **Isolated Domains**

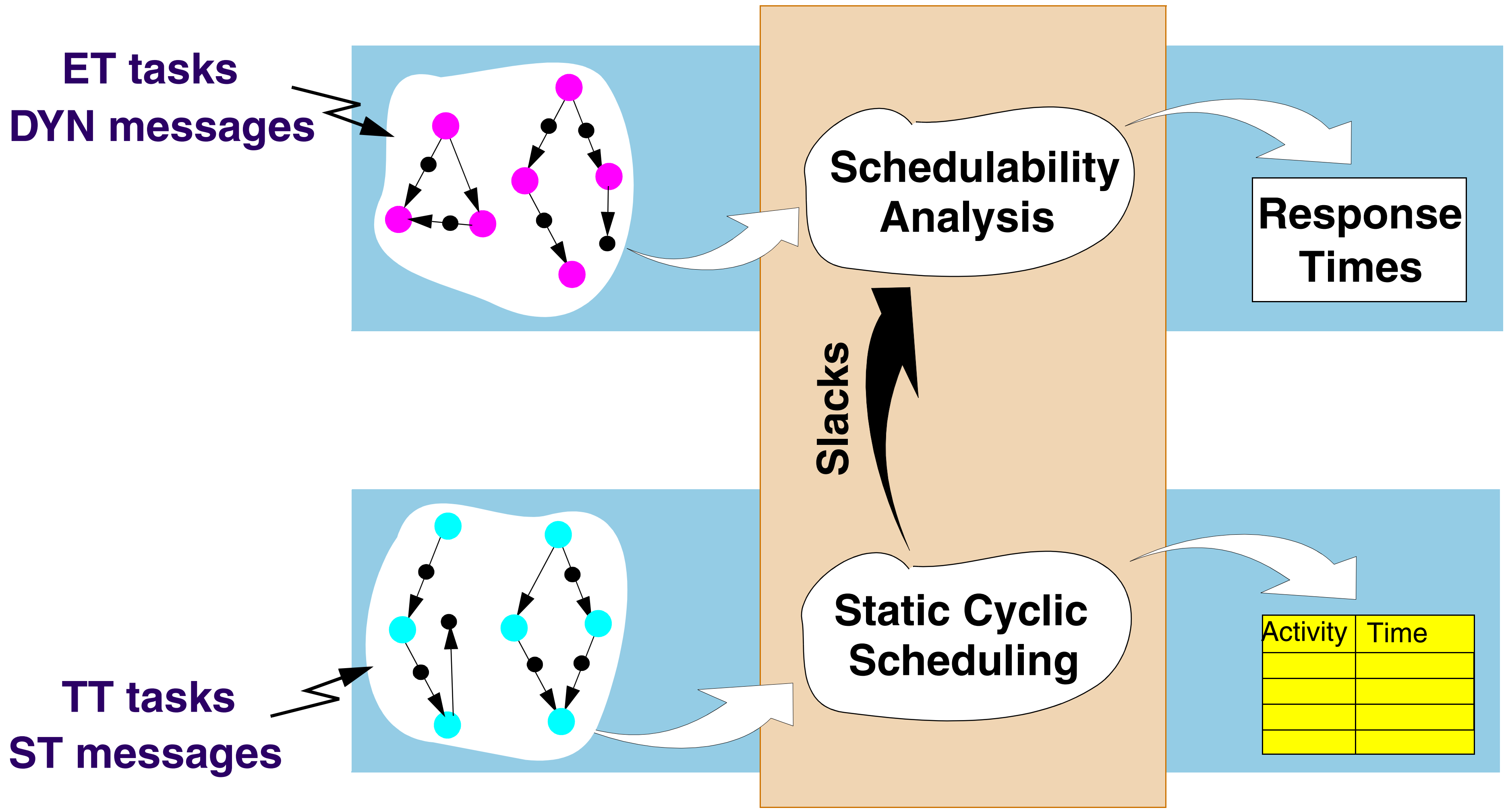
ET tasks
DYN messages



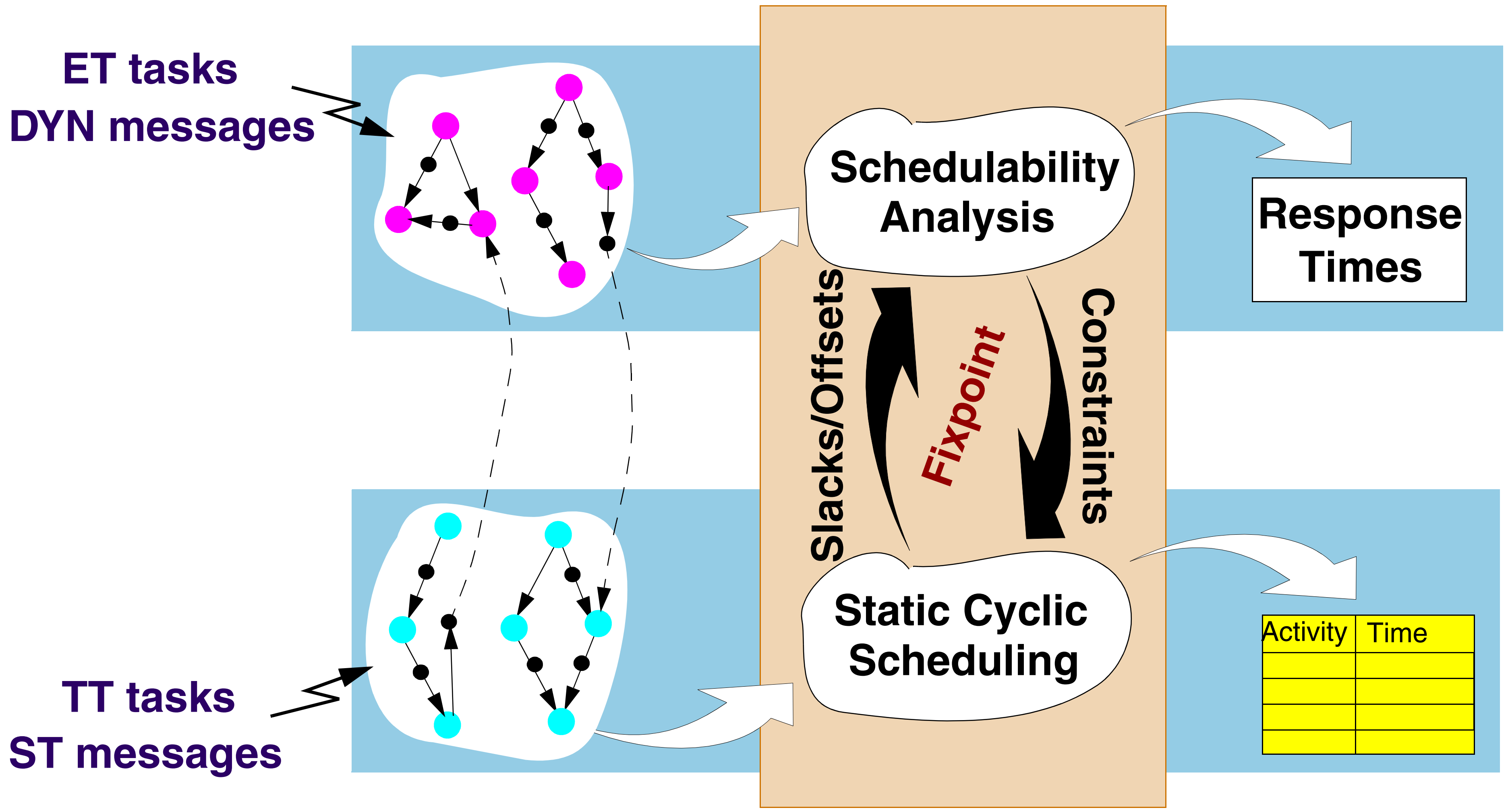
TT tasks
ST messages

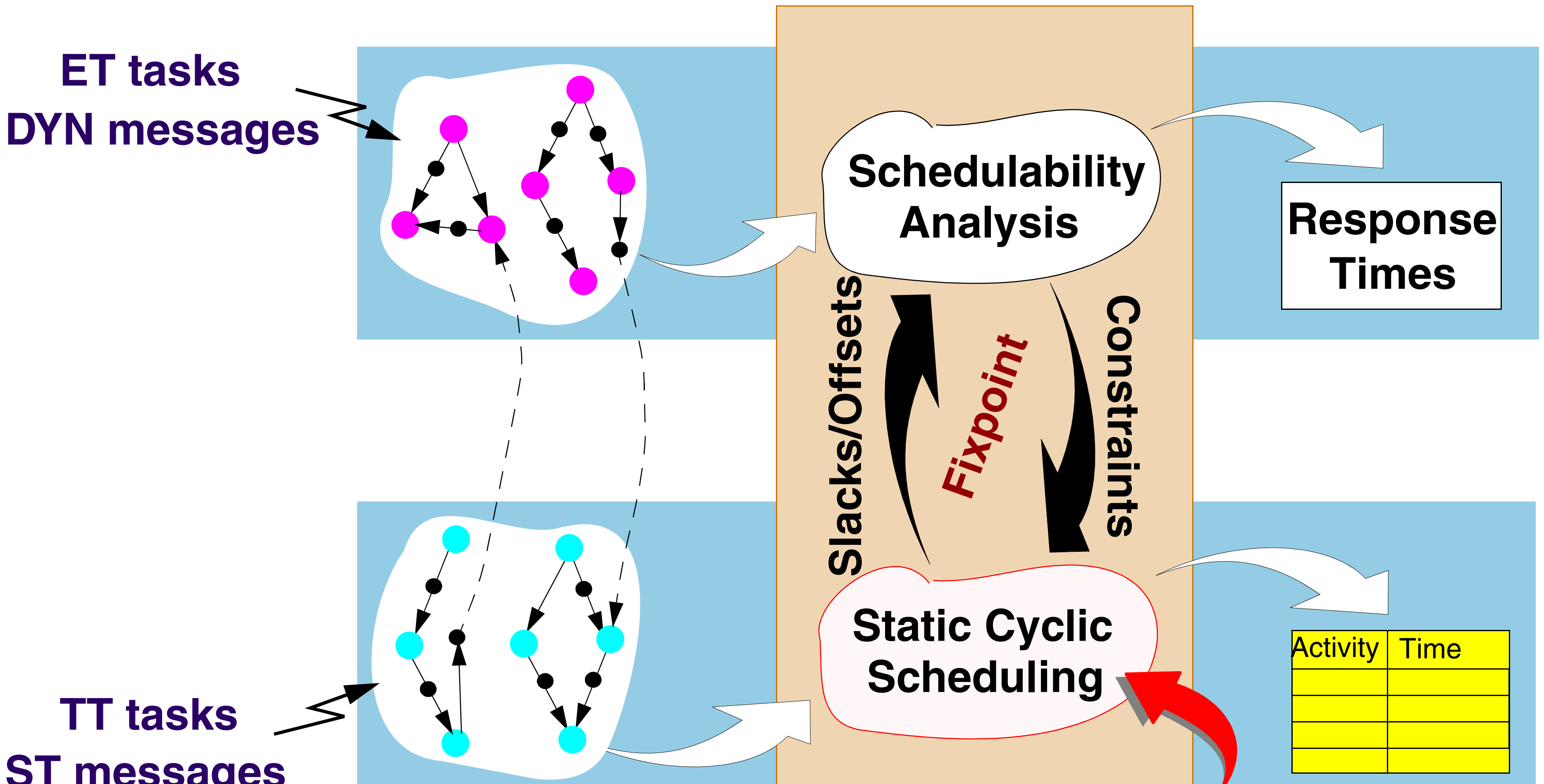


☞ **Resource sharing, no communication**



☞ **Resource sharing and communication**





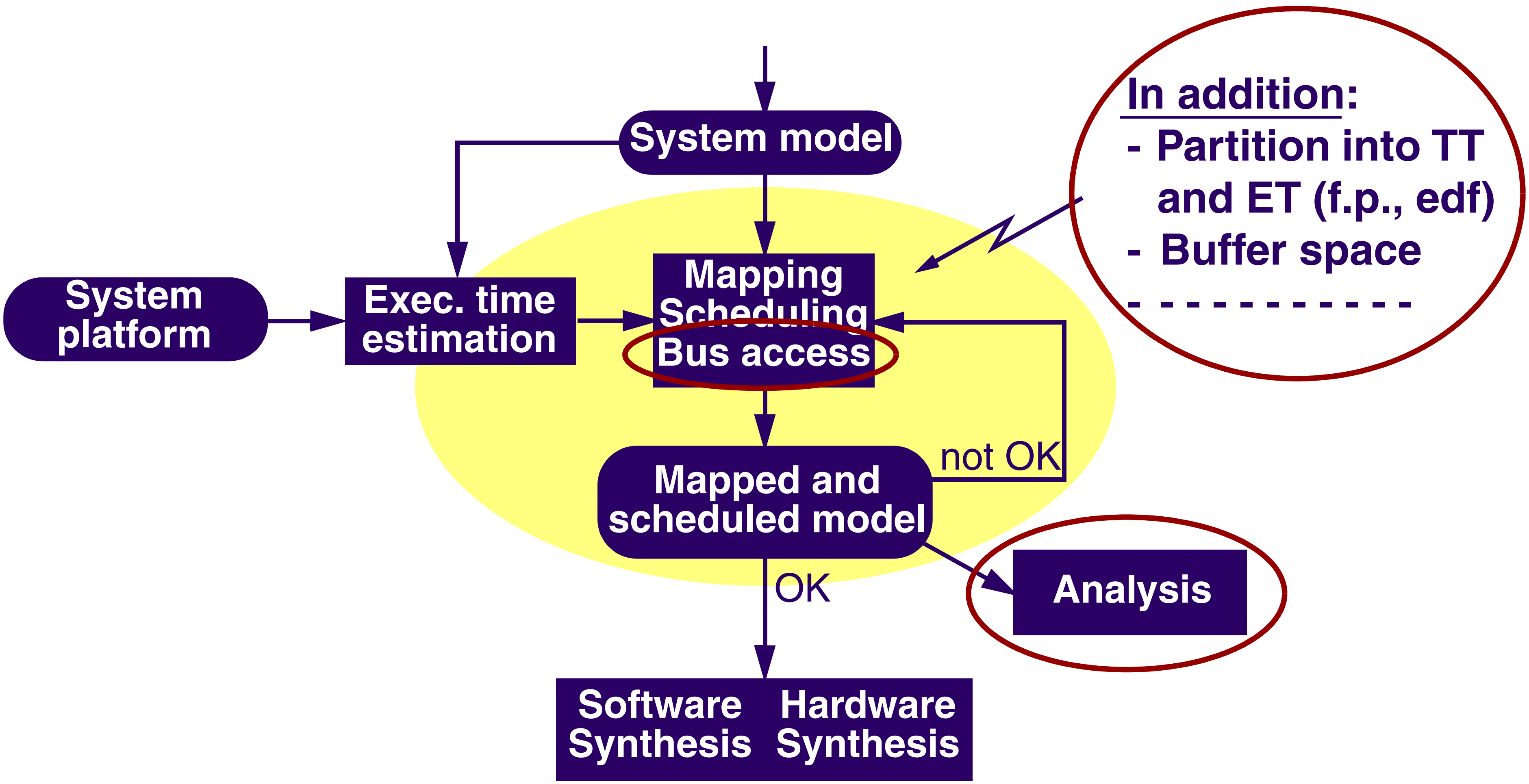
Find a schedule as “friendly” as possible towards the ET tasks!

Optimization

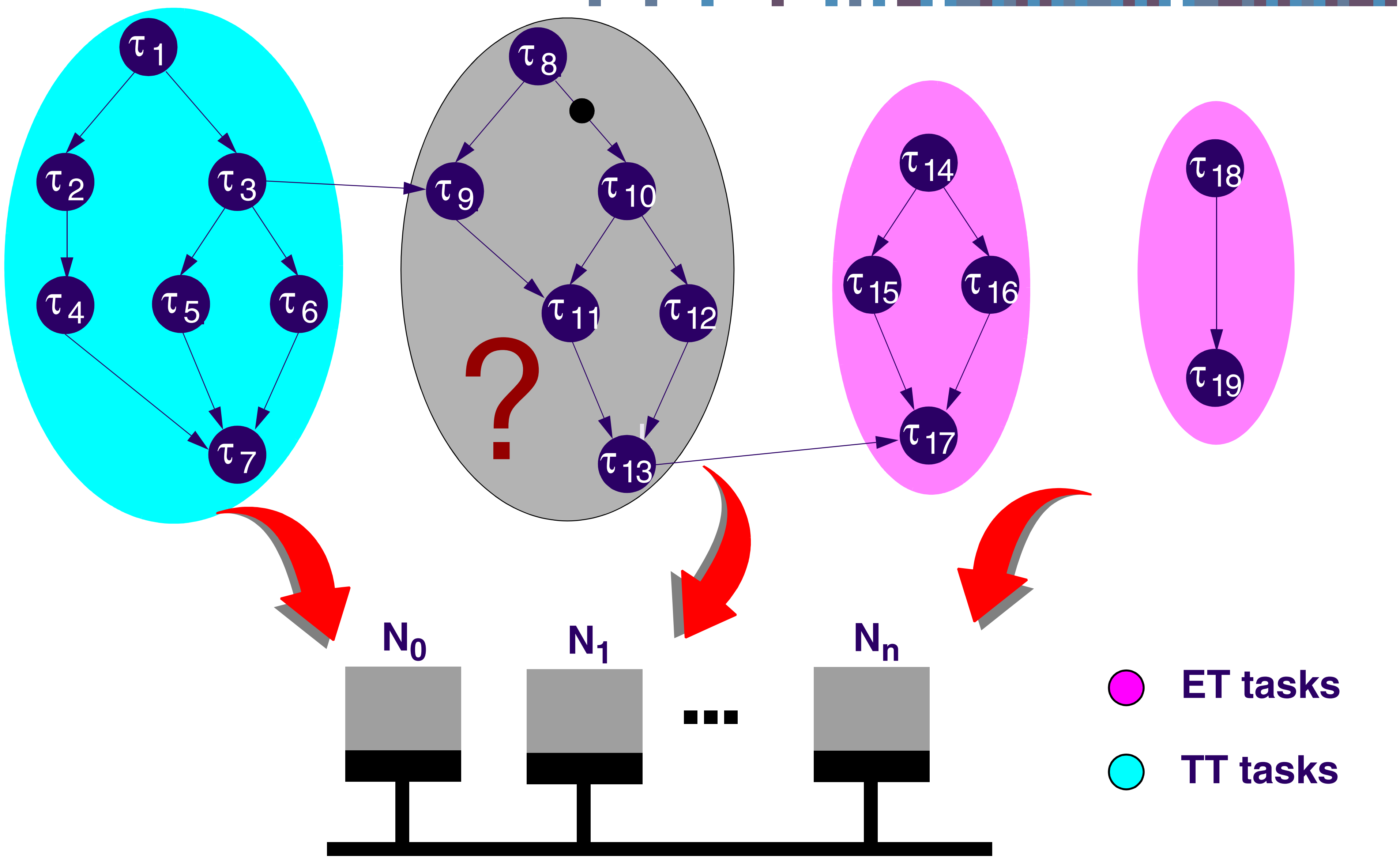
☞ **Once the analysis approach is in place, several new, specific optimization tasks can be performed (in addition to "classical" once, like task mapping):**

- **Task partitioning into TT, ET**
- **Cluster mapping**
- **Bus access optimization (static, dynamic phases)**
- **Buffer minimisation**

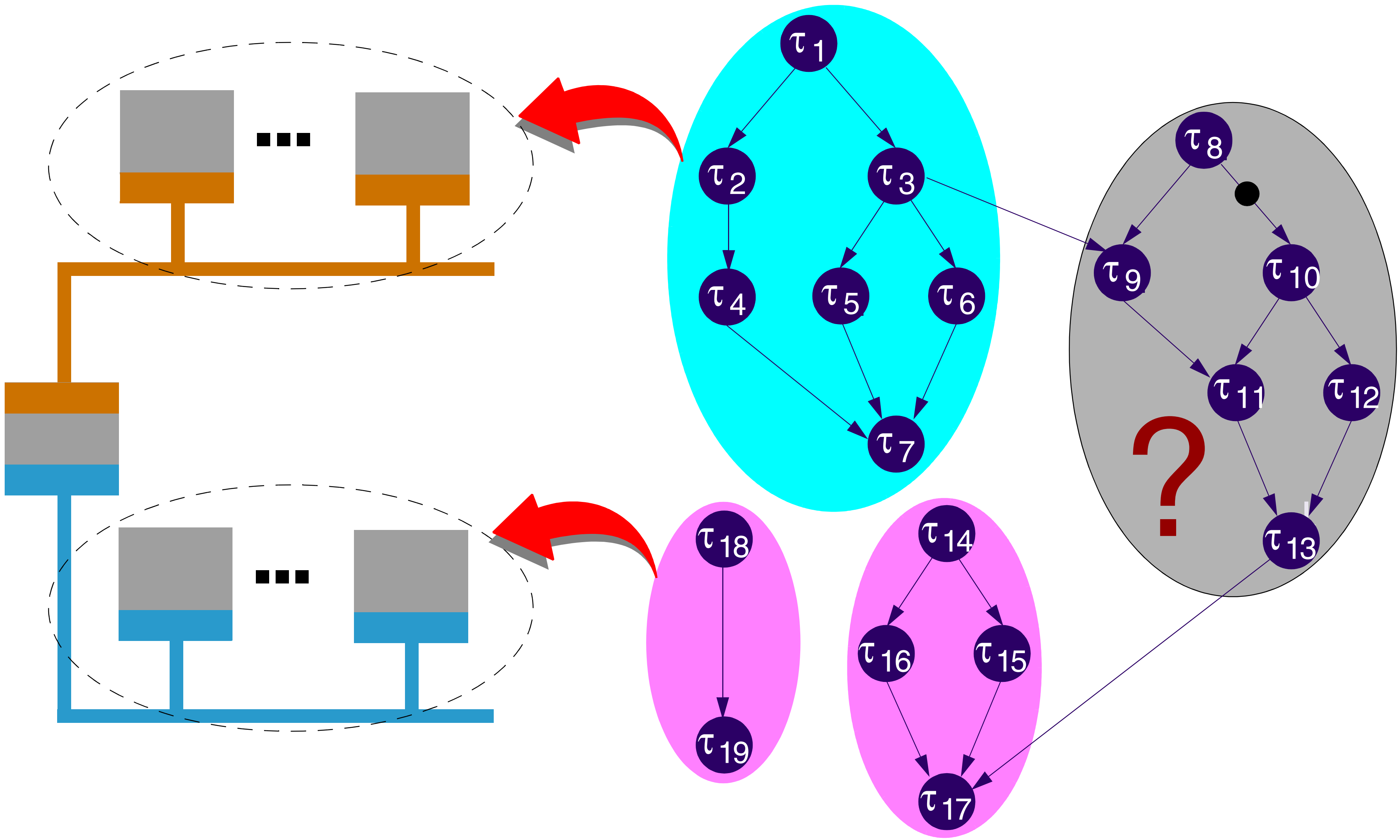
Optimization



Mapping & Task Partitioning



Mapping & Task Partitioning



Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization

Another Issue: Fault Tolerance

☞ **Transient faults**

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

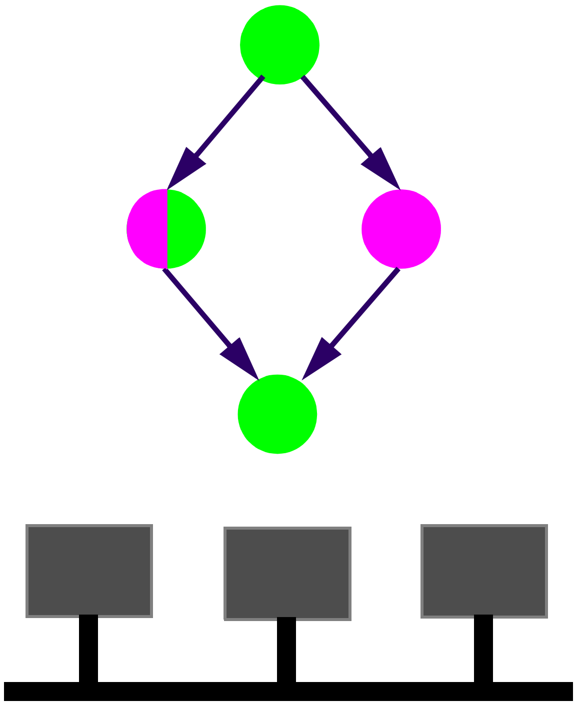
■ **Find cost-effective implementations that are fault tolerant and satisfy time constraints.**

Some Interesting trade-offs!

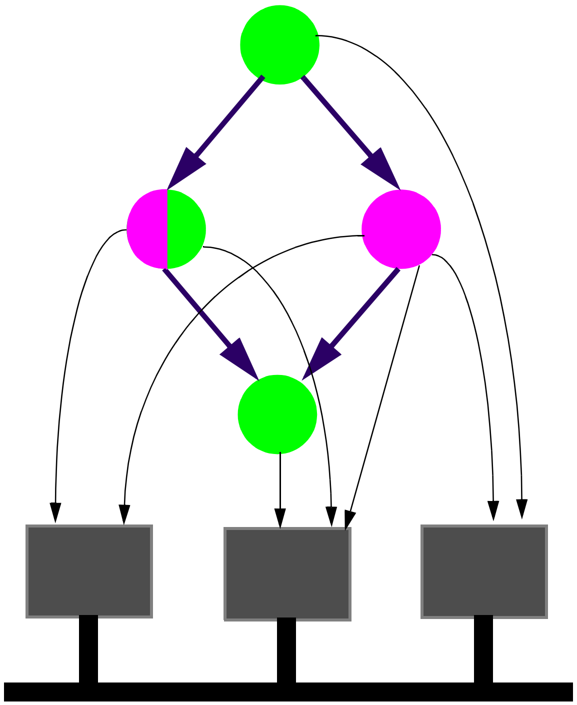
Fault Tolerance

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

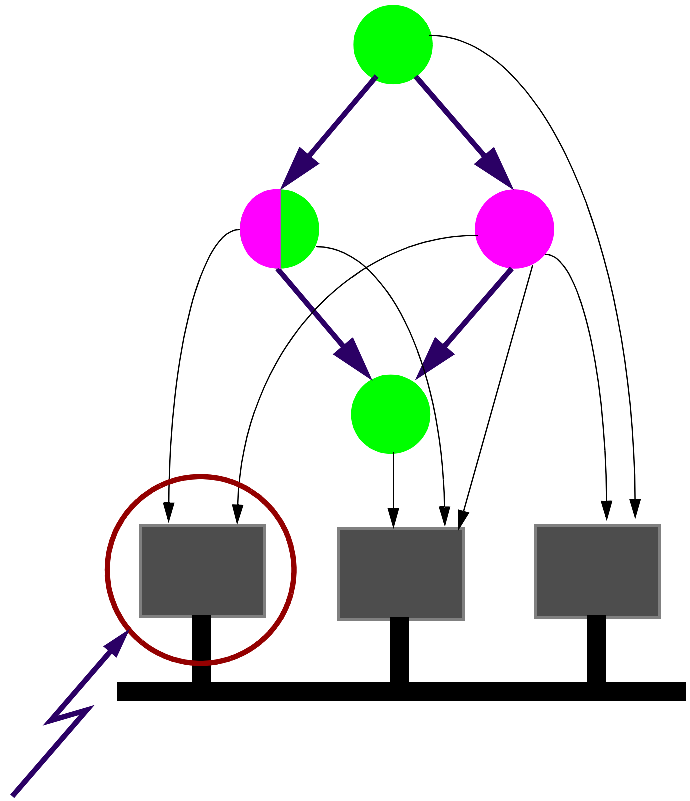
different techniques can be applied to different tasks



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



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

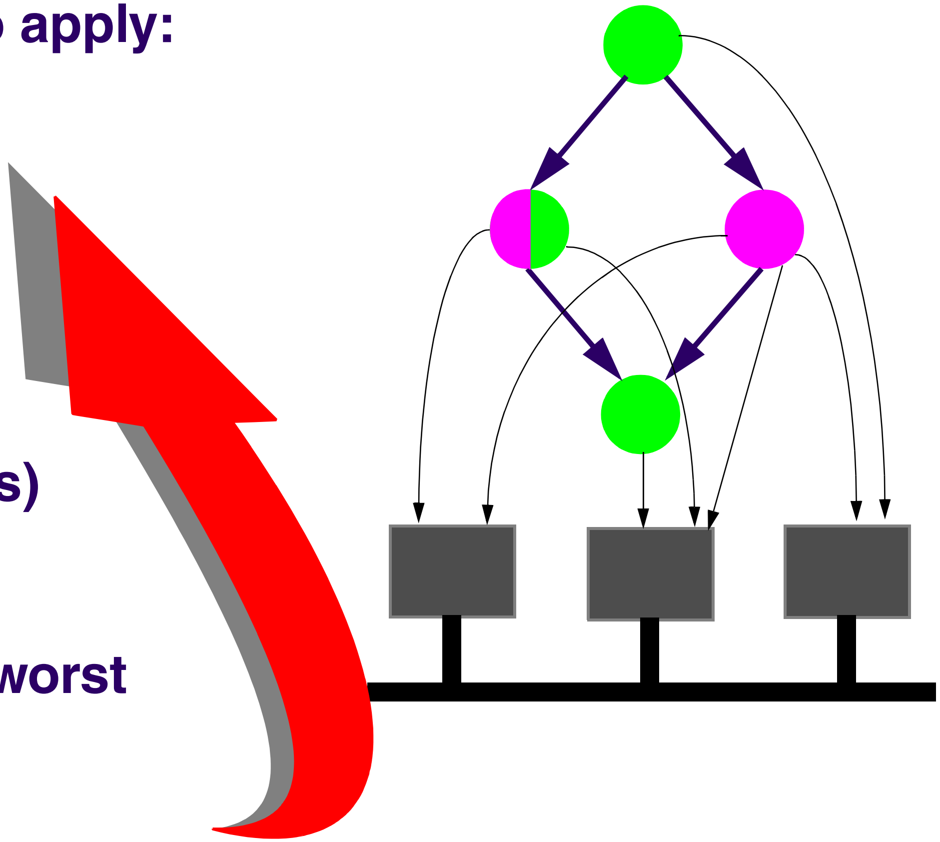


<u>Transparent</u>: The schedule of outgoing messages does not depend on occurrence of faults (faults are not visible to the outside).

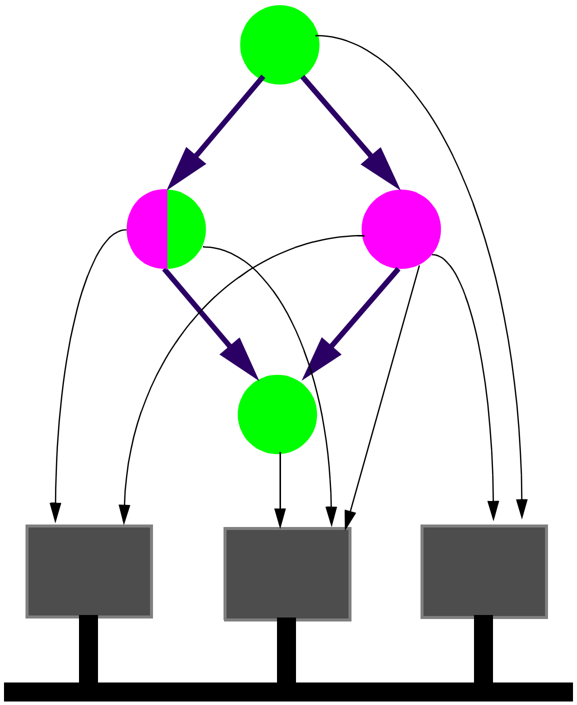
Fault Tolerance

- 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!



- 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?

Outline

- Distributed Embedded Systems
- System-level Design Flow
- Application Model
- Distributed Hard Real-Time Systems
  - Heterogeneous Distributed Systems
  - Static and Dynamic Communication
  - Analysis & Optimization
  - Fault Tolerance
- Distributed Soft Real-Time Systems
  - Analysis Approaches and Optimization
  - Optimization

Distributed Soft Real-Time Systems

- **Given:**
  - ~~WCET and BCET~~ of each task
  - Dimension of messages
  - Deadlines for tasks/task graphs

☞ ~~All~~ deadlines have to be satisfied!

Distributed Soft Real-Time Systems

- **Given:**
  - Execution Time Probability Distribution Function (ETPDF) of each task
  - Dimension of messages
  - Deadlines for tasks/task graphs

☞ **A certain percentage of deadlines have to be satisfied!**

Distributed Soft Real-Time Systems

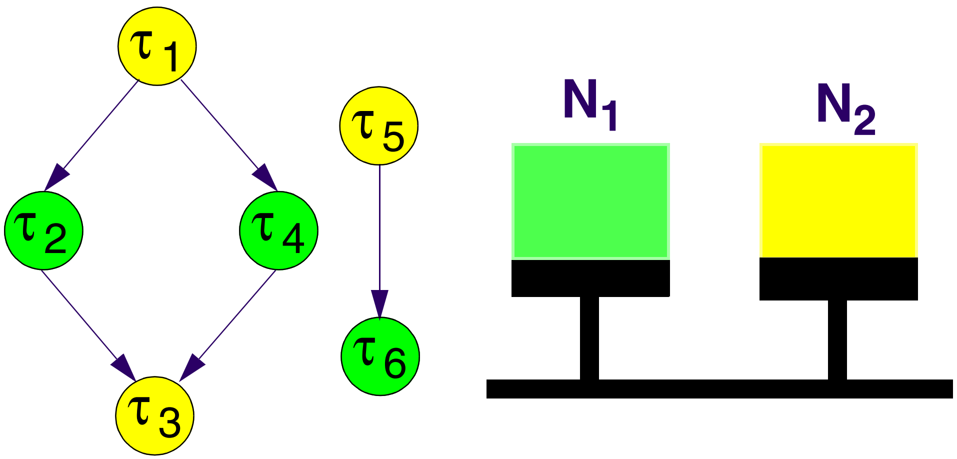
- **Given:**
  - Execution Time Probability Distribution Function (ETPDF) of each task
  - Dimension of messages
  - Deadlines for tasks/task graphs

☞ **A certain percentage of deadlines have to be satisfied!**

- **Problems:**
  - Determine the percentage of missed deadlines for each task/task graph.
  - Optimize task mapping and priority assignment, such that a required percentage of satisfied deadlines is achieved.

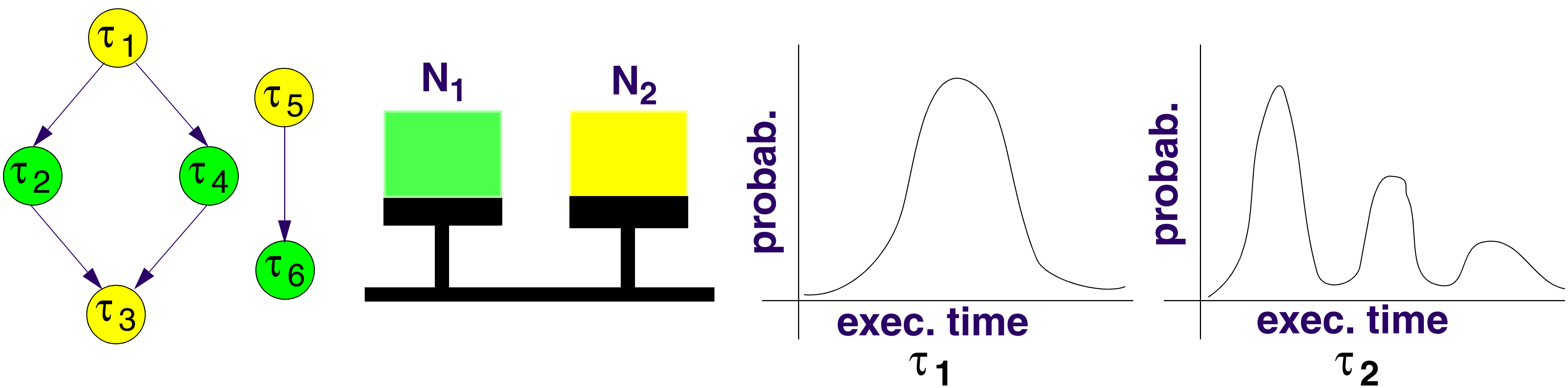
Input:

- **Set of mapped task graphs, periods, deadlines, priorities.**



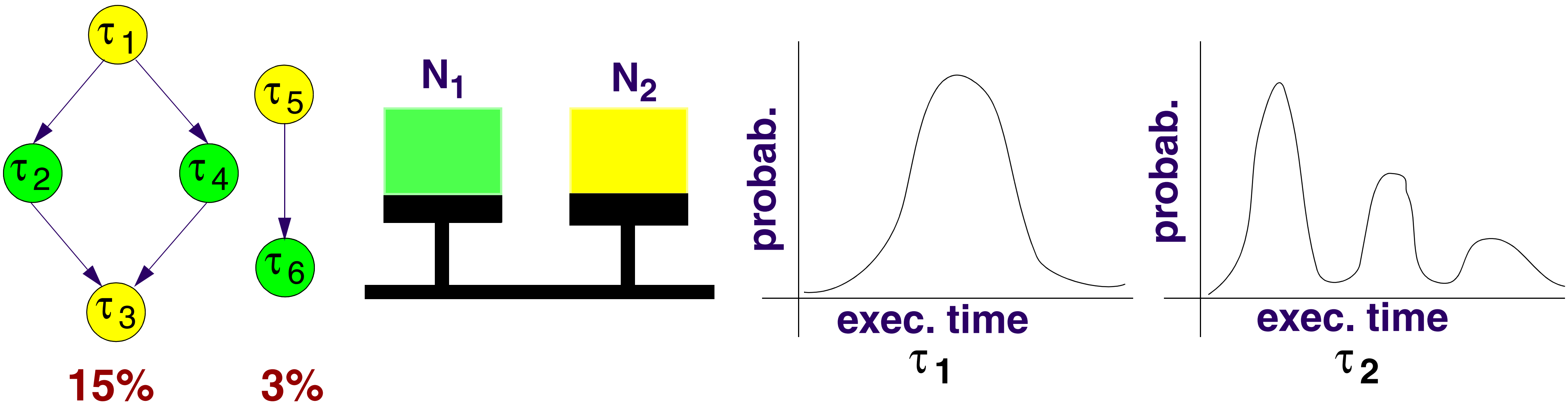
Input:

- Set of mapped task graphs, periods, deadlines, priorities.
- Set of execution times probability density functions.
- Scheduling policy.



Output:

- Ratio of missed deadlines per task/task graph



Main Problem

- The main problem is **Complexity** (in terms of analysis time and memory).

    - Applying straight-forward solutions is possible

        - only for very(!) small applications.

        and/or

        - for very particular cases (e.g. exponential distributions).

Approaches

- An exact method for schedulability analysis, efficiently applicable (but not limited) to monoprocessor systems.

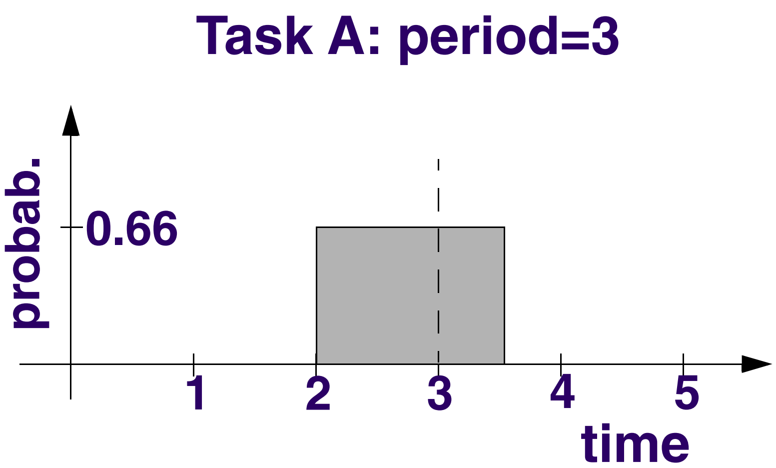
- An method for schedulability analysis, trading analysis efficiency for result accuracy; efficiently applicable to multiprocessors.

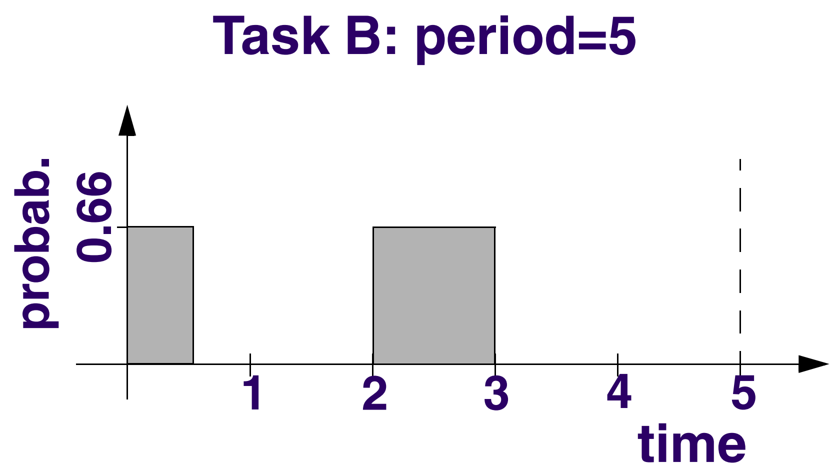
- A very fast, approximate analysis, to be used inside an optimization loop.

Stochastic Process Construction

- Analyse of the underlying stochastic process.
  - A state of the process should capture enough information to be able to generate the next states and to compute the corresponding transition probabilities

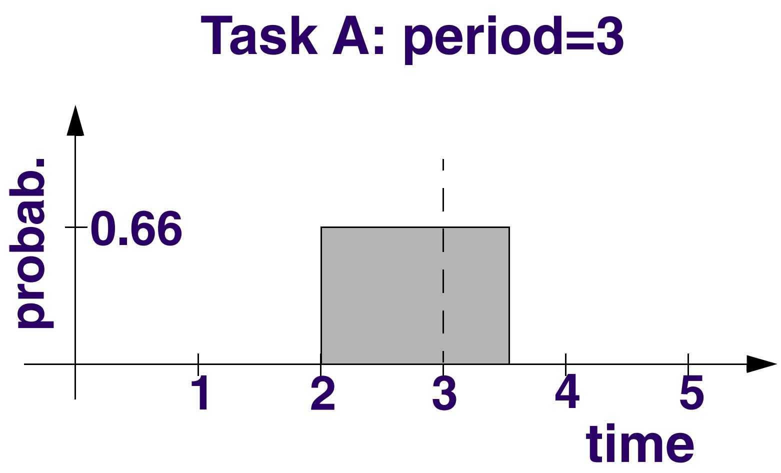
Stochastic Process Construction

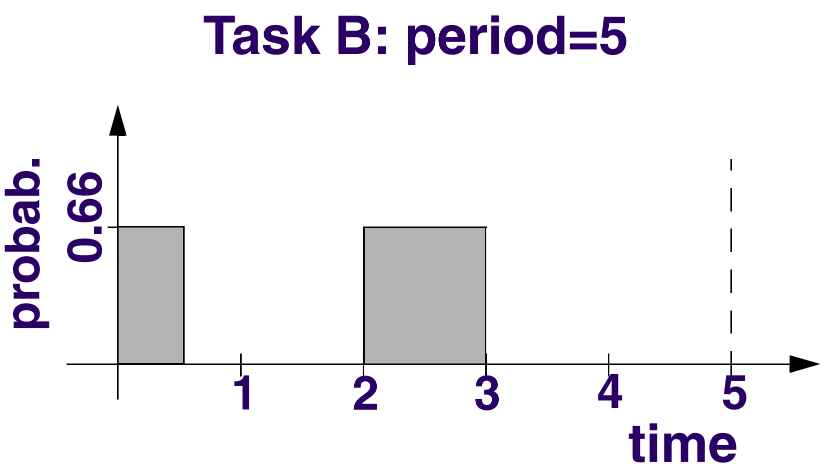


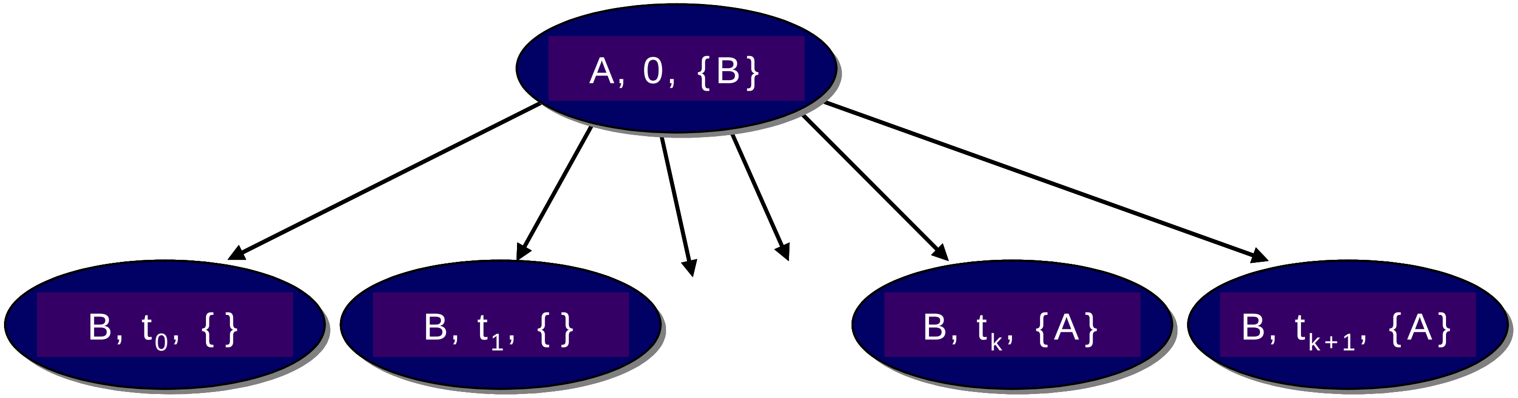


☞ EDF scheduling

Stochastic Process Construction

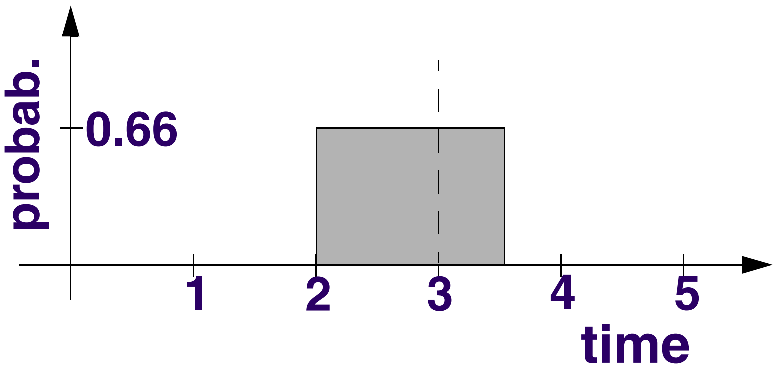


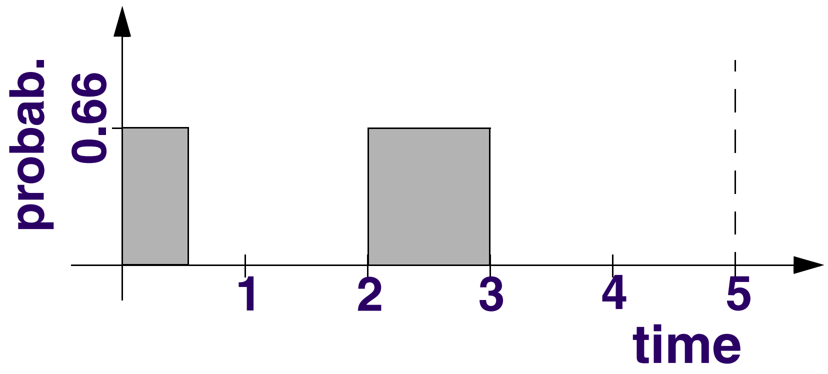


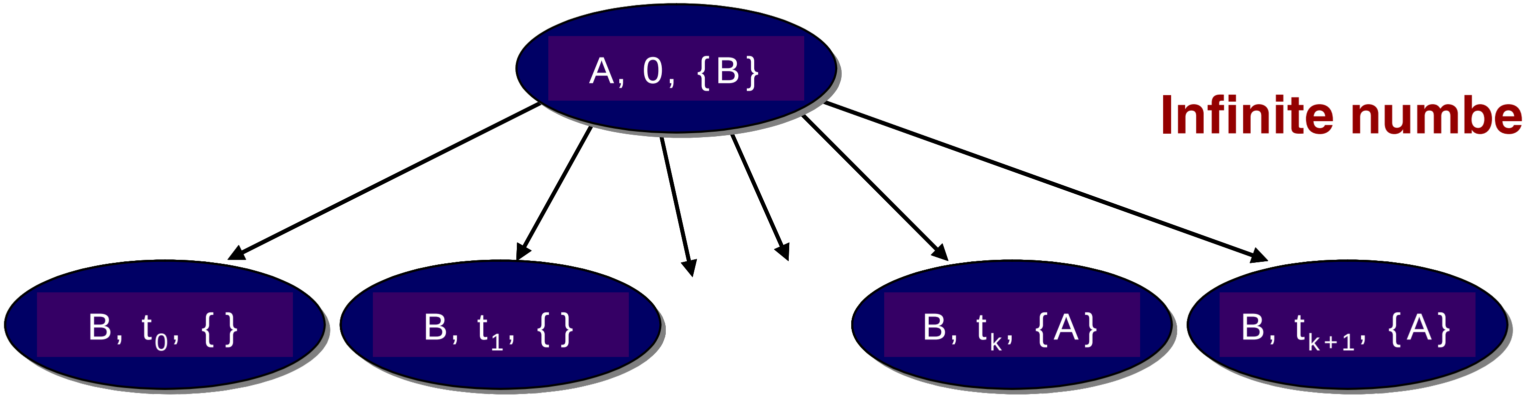


Stochastic Process Construction

Task A: period=3 **Task B: period=5**



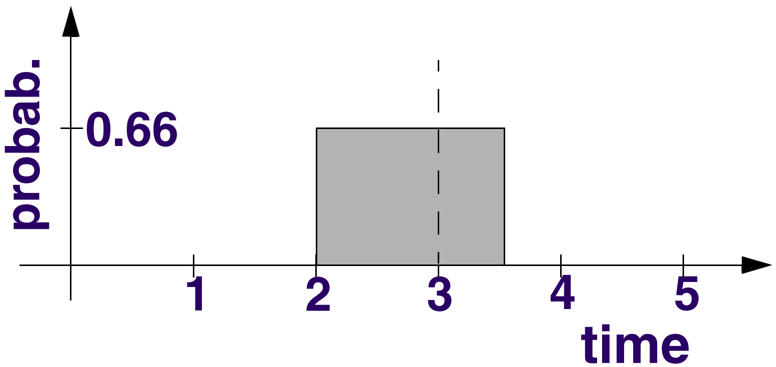




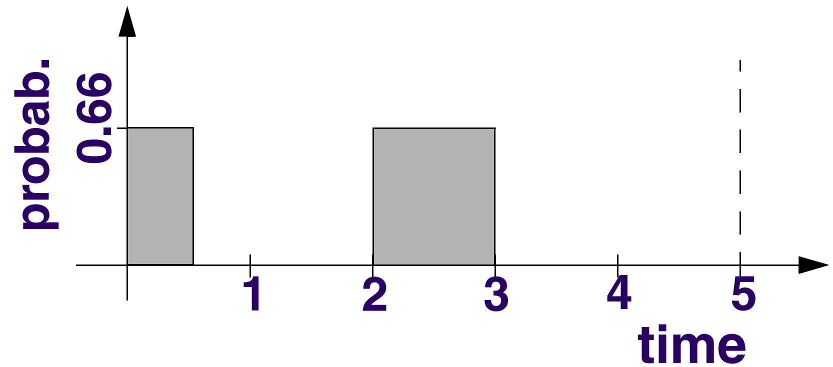
Infinite number of states!

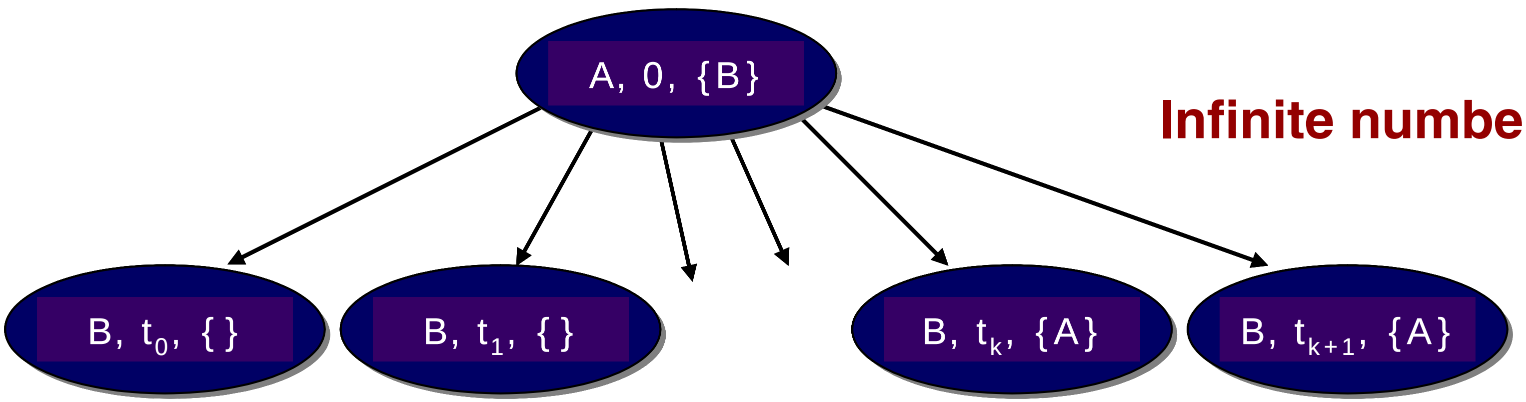
Stochastic Process Construction

Task A: period=3

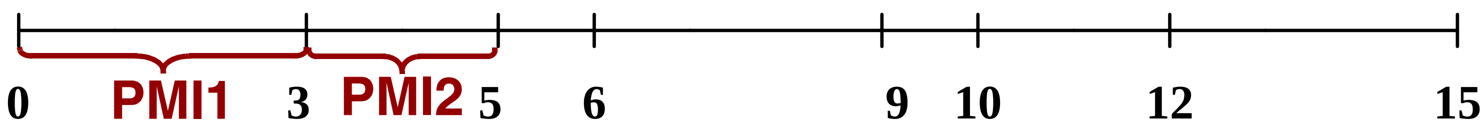


Task B: period=5

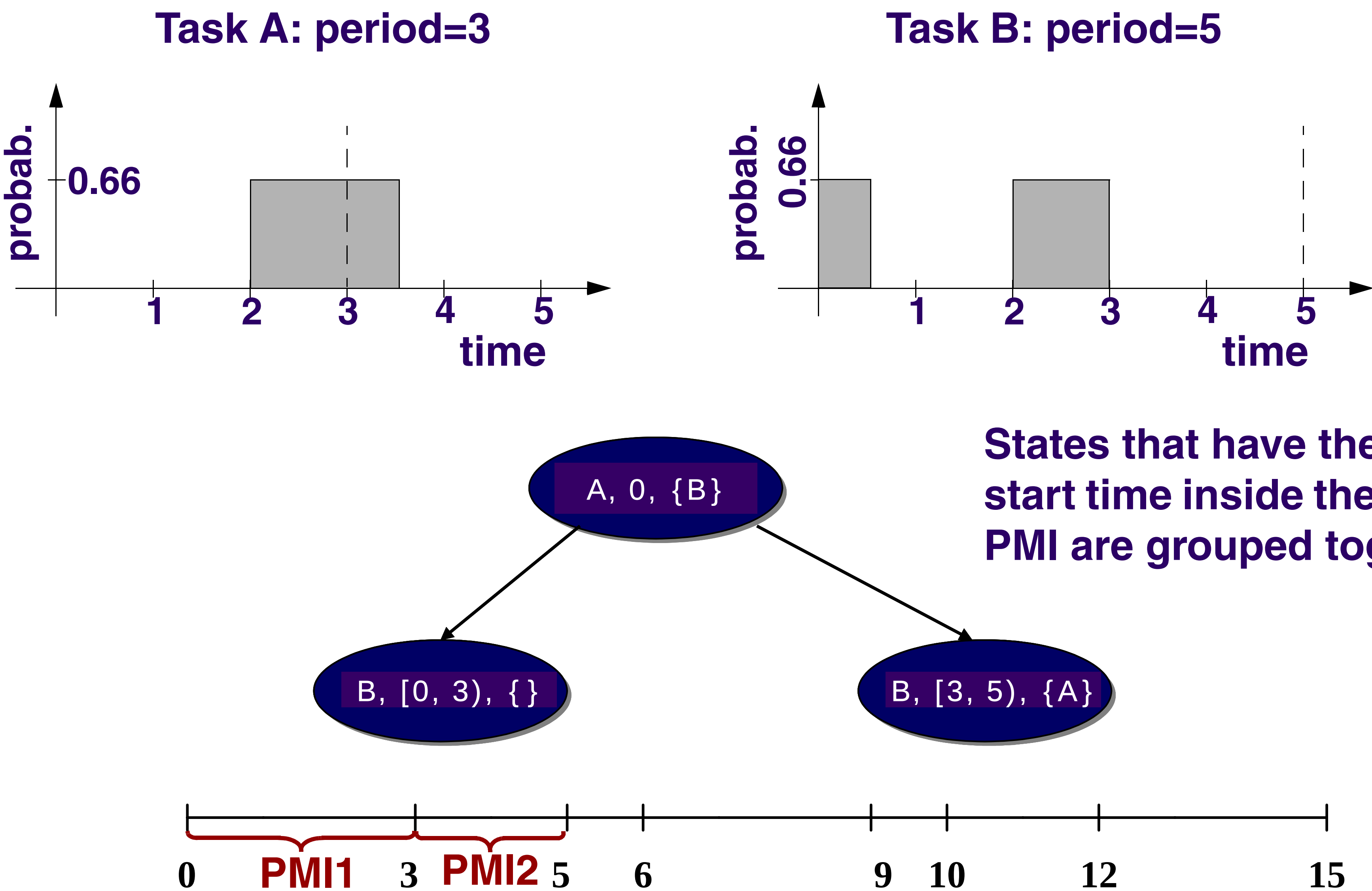




Infinite number of states!



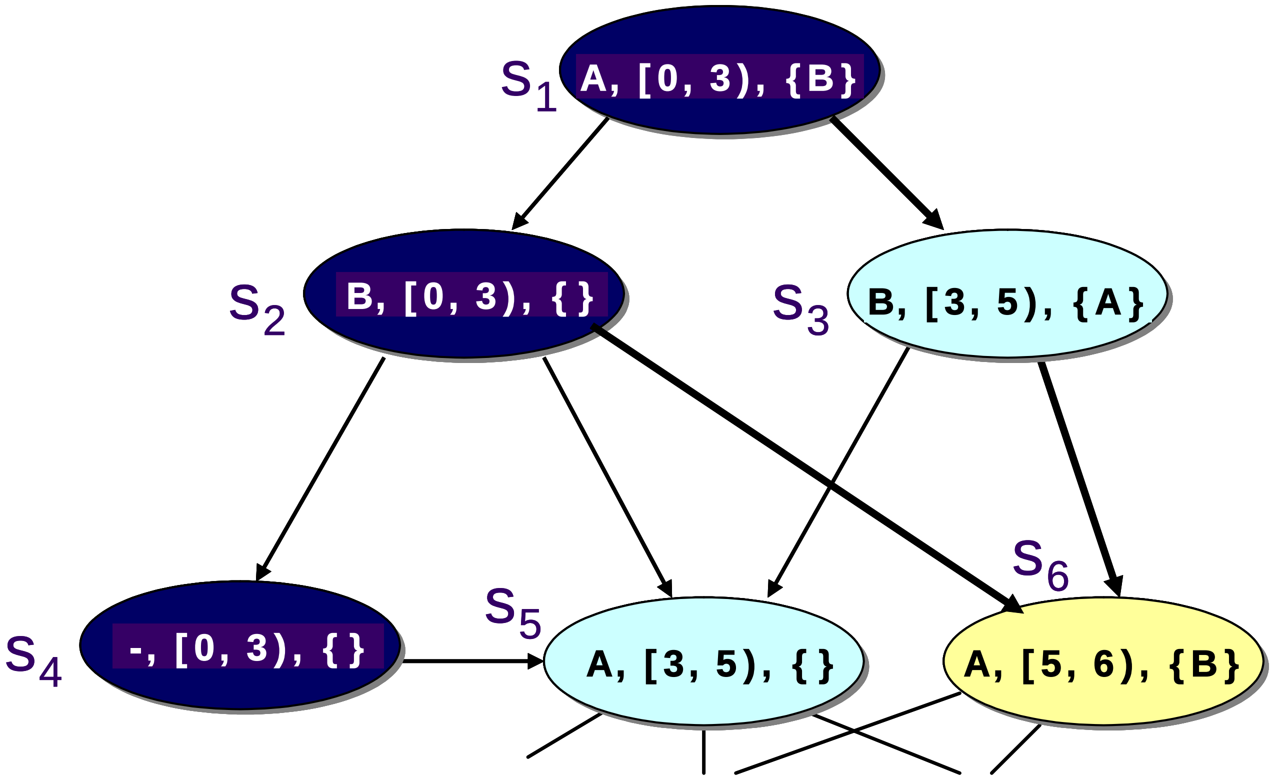
Stochastic Process Construction



States that have their start time inside the same PMI are grouped together.

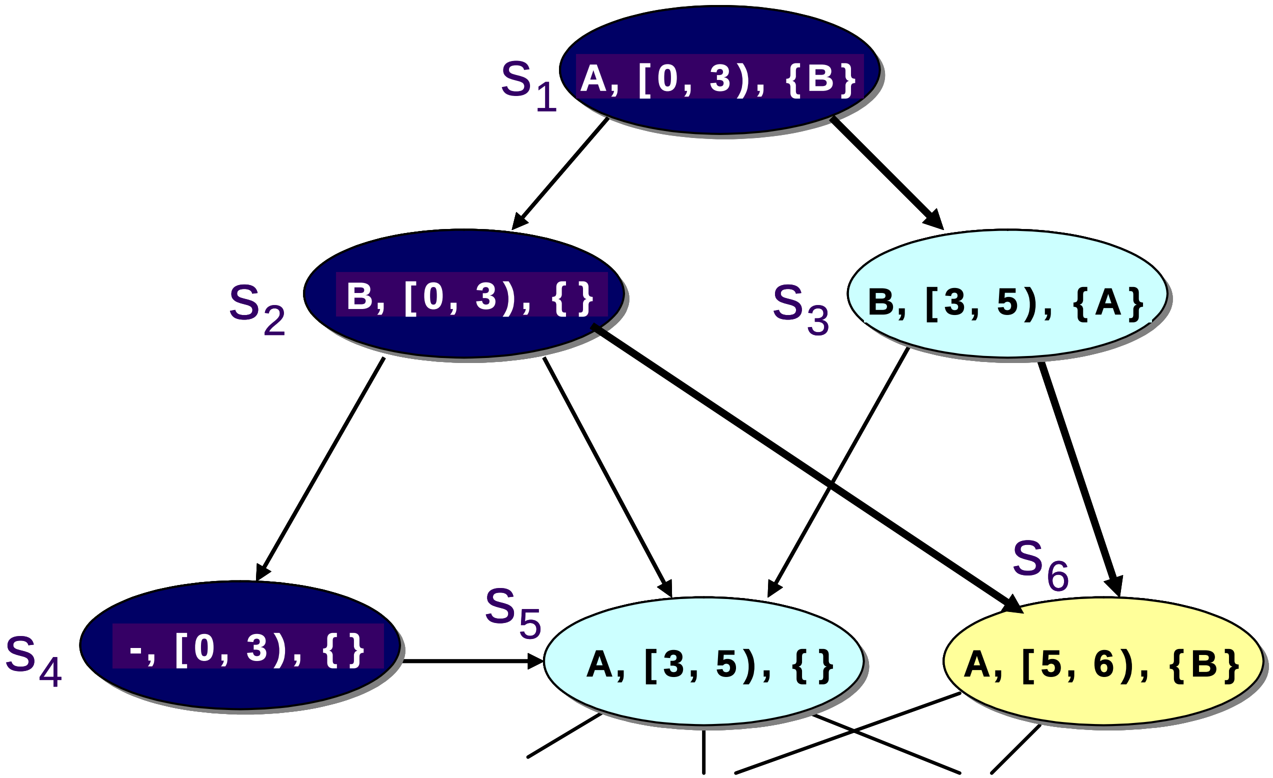
Stochastic Process Construction

☞ **The Stochastic process is constructed and probabilities corresponding to individual states are calculated by convolution.**



Stochastic Process Construction

- The Stochastic process is constructed and probabilities corresponding to individual states are calculated by convolution.
- **Good News:** Process construction and analysis are performed simultaneously. Only a (very small) sliding window of the states has to be stored in memory at any time.



Limitation

- The number of states increases very much in the case of *interacting tasks implemented on multiprocessor systems*.

- We need to avoid
  - storing explicitly the distribution of residual exec. time in each state;
  - calculation of the convolutions.

Multiprocessor Approach

The generalised ETPDFs are approximated by weighted sums of exponential functions (Coxian approximation).

Multiprocessor Approach

The generalised ETPDFs are approximated by weighted sums of exponential functions (Coxian approximation).

The initial generalised semi-Markov process is transformed into a continuous time Markov chain.

Larger, but much easier to analyse

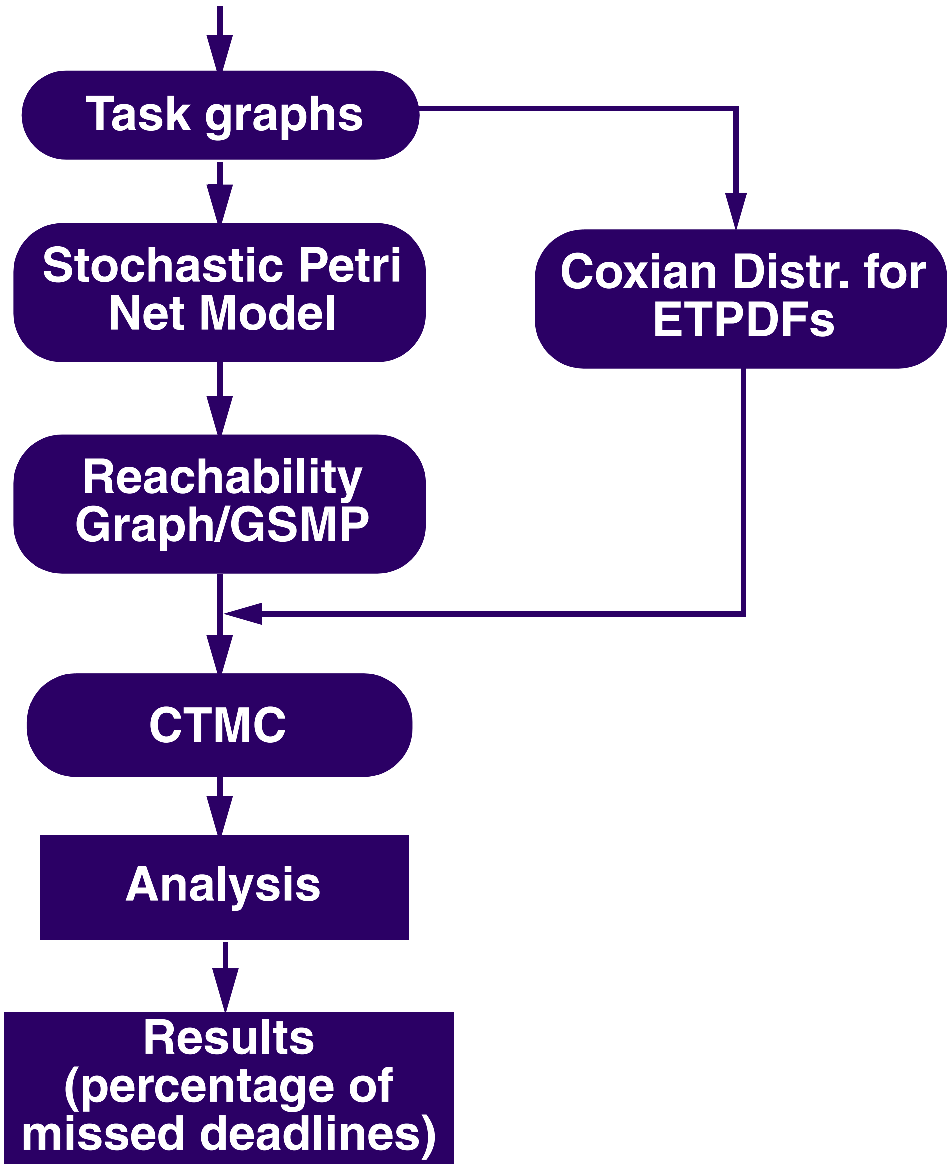
The generalised ETPDFs are approximated by weighted sums of exponential functions (Coxian approximation)

↓

The initial generalised semi-Markov process is transformed into a continuous time Markov chain.

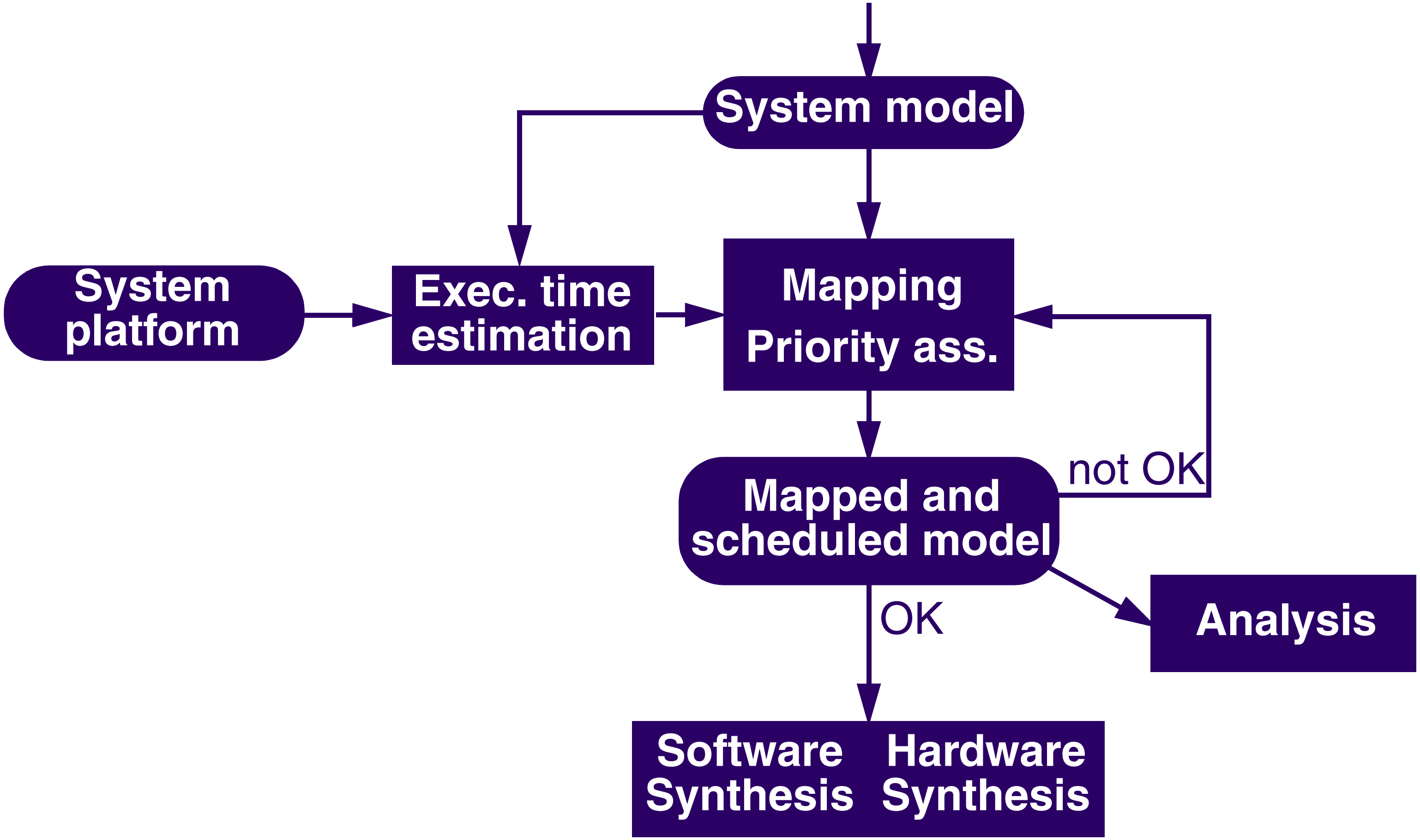
- ☞ **Good News**: **The whole matrix, corresponding to the complete model, has *not* to be stored in memory; it is generated on the fly, from terms expressed as small matrices.**
- ☞ **Accuracy can be traded for analysis time/memory.**

Multiprocessor Approach



- **Problem:**
  - **Optimize task mapping and priority assignment, such that a required percentage of satisfied deadlines is achieved.**

- **Problem:**
  - **Optimize task mapping and priority assignment, such that a required percentage of satisfied deadlines is achieved.**

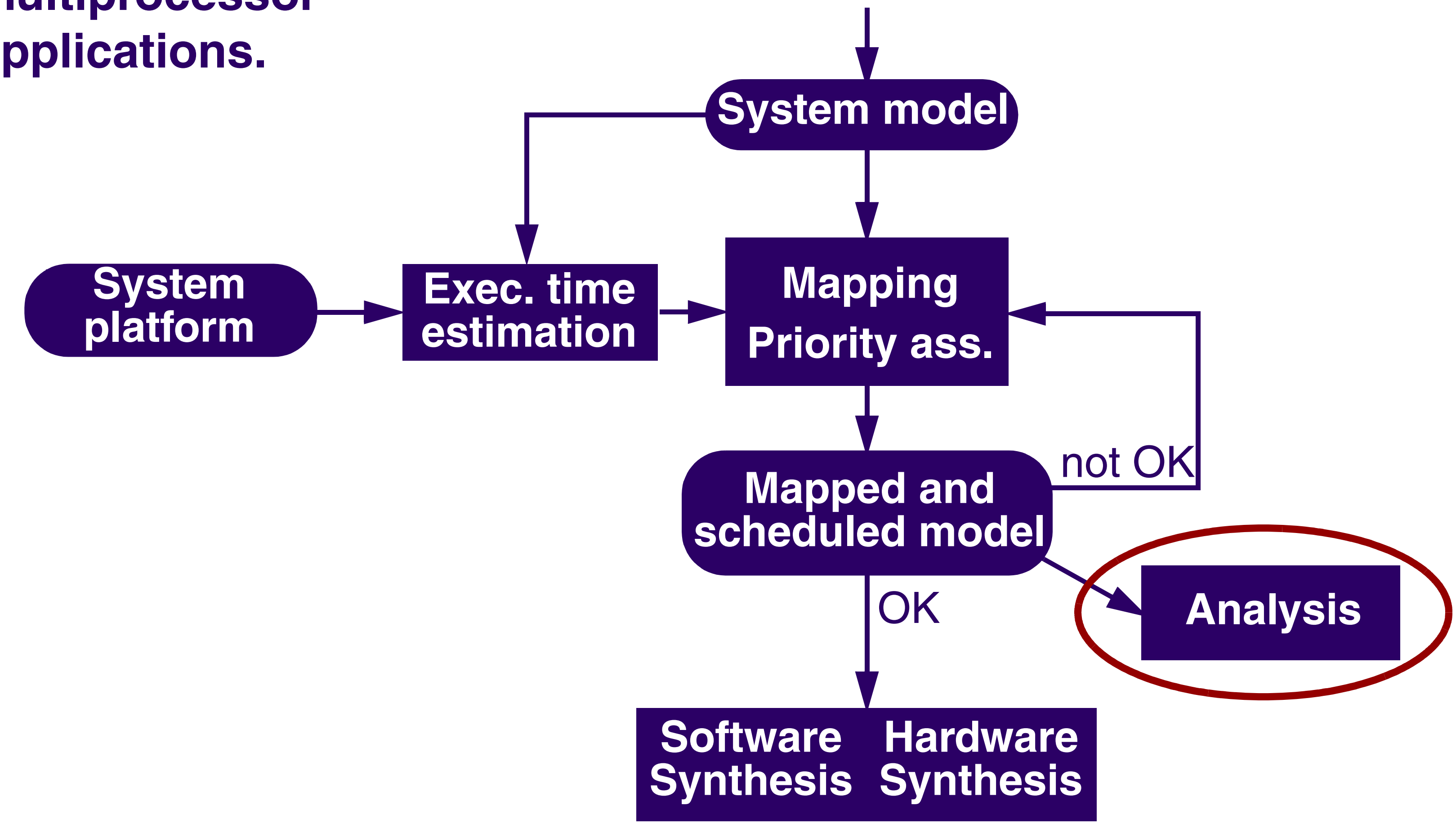


System Optimization

☞ The presented analysis approach works for large multiprocessor applications.

But

☞ It is too slow to be used inside an optimization loop.



- ☞ An approximate analysis method of polynomial complexity has been developed.

- **Basic idea:**
  - Weak dependencies among the random variables have been neglected.

↓

Very fast analysis, sufficiently accurate to guide the design space exploration.

System Optimization

If you optimize your system to minimise average or worst case execution time

If you optimize your system to minimise average or worst case execution time and you hope that the resulted design is close to optimal with regard to the percentage of missed deadlines

If you optimize your system to minimise average or worst case execution time and you hope that the resulted design is close to optimal with regard to percentage of missed deadlines

then

YOU ARE VERY WRONG!!!

Conclusions

- Distributed embedded systems are becoming common.

- They are communication dominated and heterogeneous.
- They have to be safe (fault tolerance).

- Hard distributed real-time systems.
- Soft distributed real-time systems.

} How to guarantee timing constr.?

- Analysis (in particular timing) is difficult, but possible.

  Simulation is not the right solution!

- It is not sufficient to analyse! Help the designer to implement a cost effective system that satisfies the imposed constraints!

  Efficient optimization tools are needed.