Embedded Systems Design: Optimization Challenges

Paul Pop
Embedded Systems Lab (ESLAB)
Linköping University, Sweden

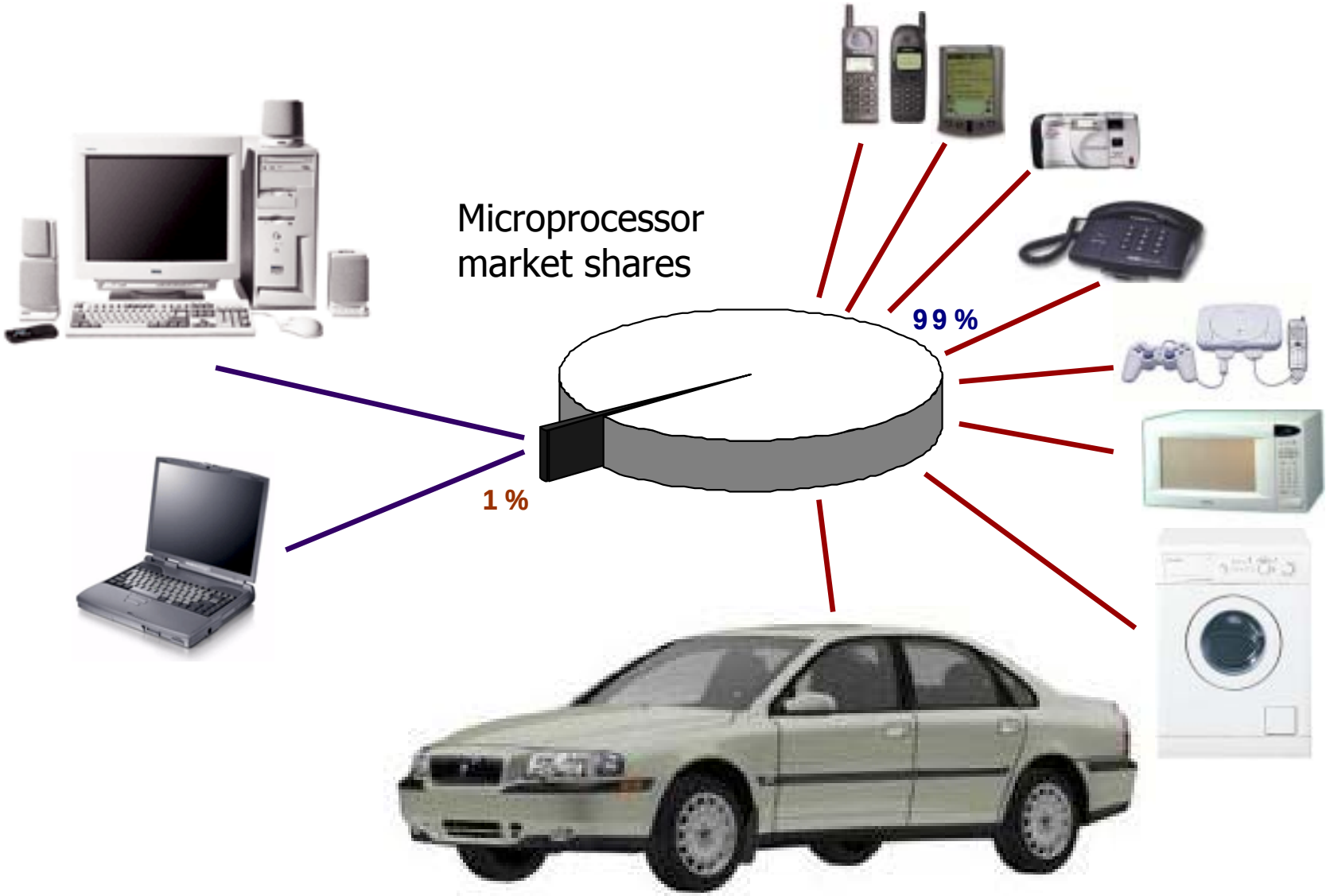
Outline

- → Embedded systems
  - Example area: automotive electronics
- Embedded systems design
- Optimization problems
  - Fault-tolerant mapping and scheduling
  - Voltage scaling
  - Communication delay analysis
- Assessment and message

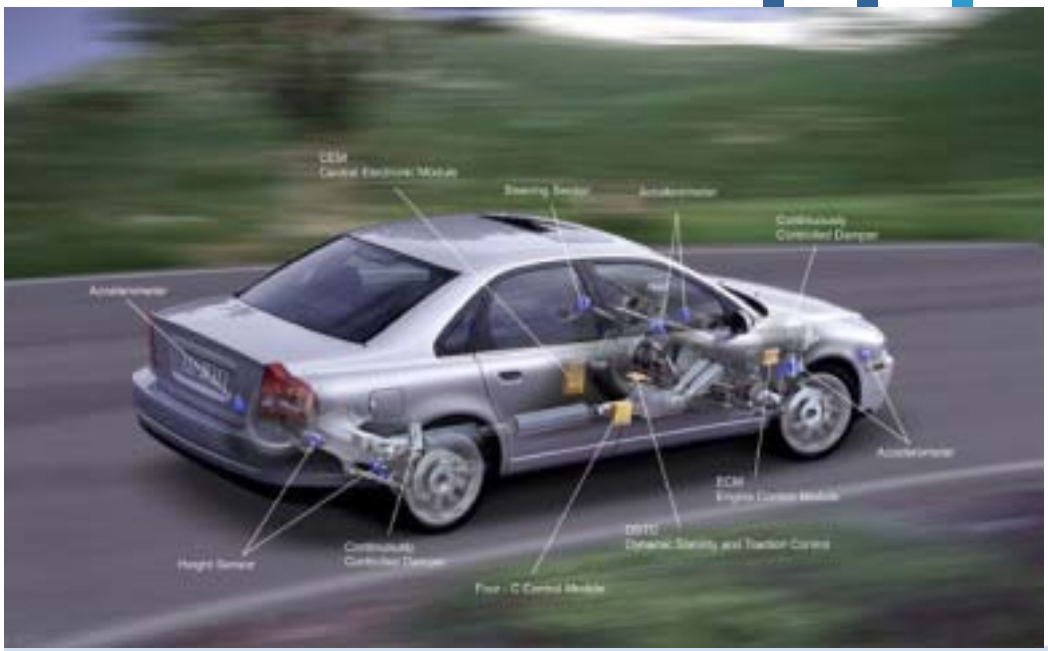
Embedded Systems

General purpose systems

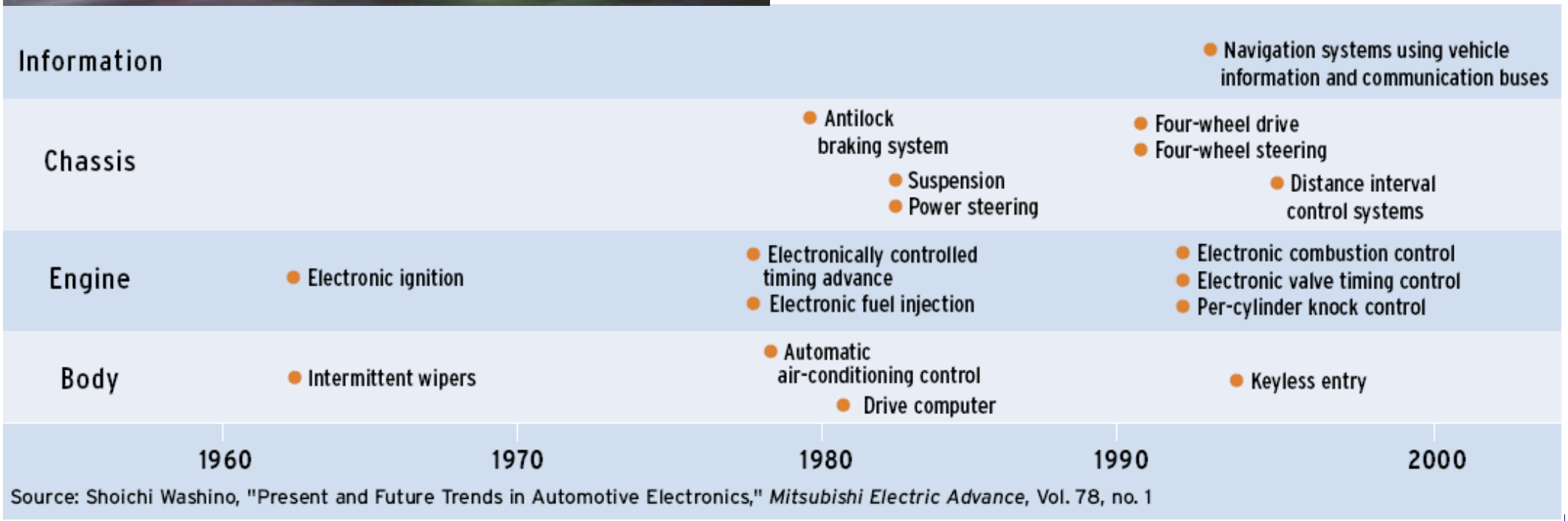
Embedded systems



Example Area: Automotive Electronics

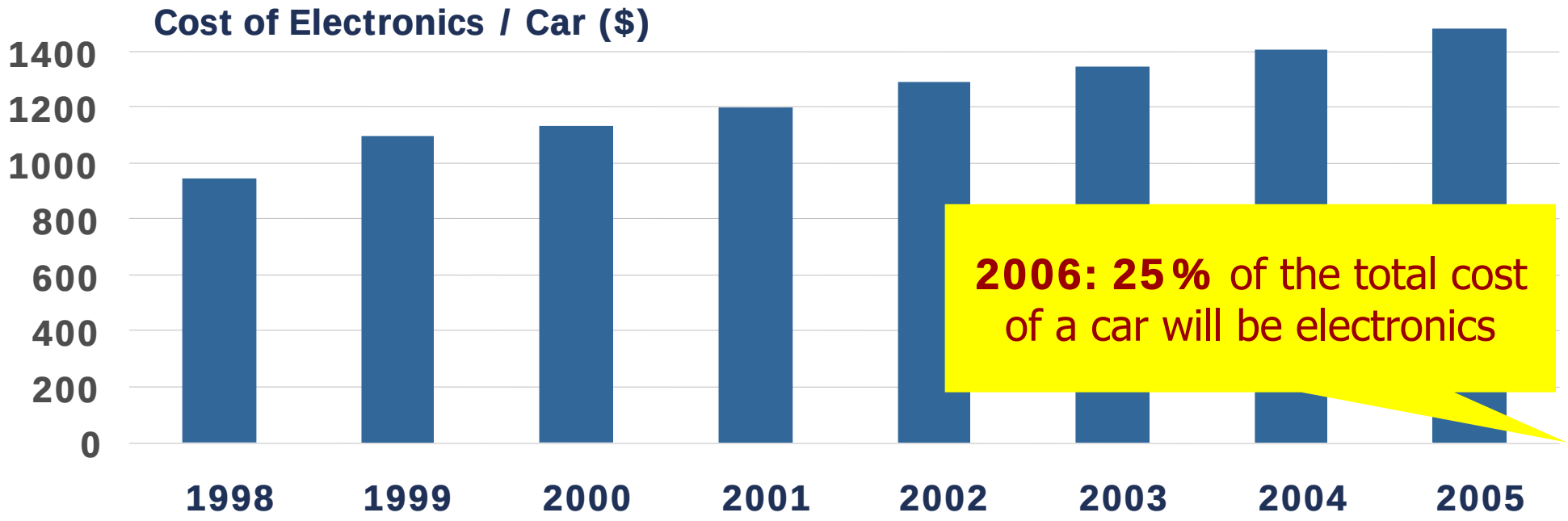


- What is "automotive electronics"?
  - Vehicle functions implemented with electronics
    - Body electronics
    - System electronics: chassis, engine
    - Information/entertainment



Source: Shoichi Washino, "Present and Future Trends in Automotive Electronics," *Mitsubishi Electric Advance*, Vol. 78, no. 1

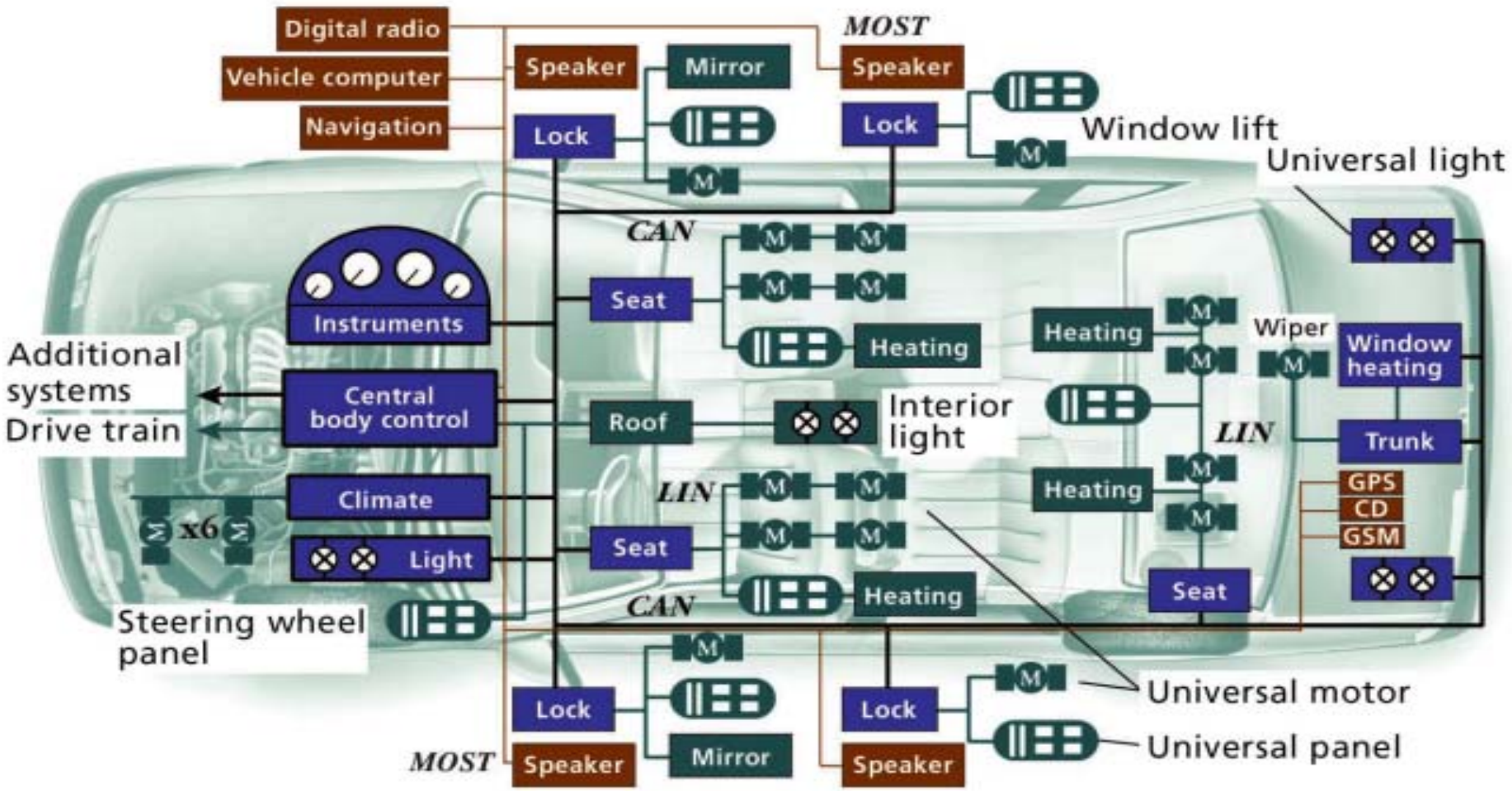
Automotive Electronics Market Size



Market ($billions)	**8.9**	**10.5**	**13.1**	**14.1**	**15.8**	**17.4**	**19.3**	**21.0**

90% of future innovations in vehicles:
based on electronic **embedded systems**

Automotive Electronics Platform Example



CAN Controller area network
GPS Global Positioning System
GSM Global System for Mobile Communications
LIN Local interconnect network
MOST Media-oriented systems transport

Source: Expanding automotive electronic systems, IEEE Computer, Jan. 2002

Outline

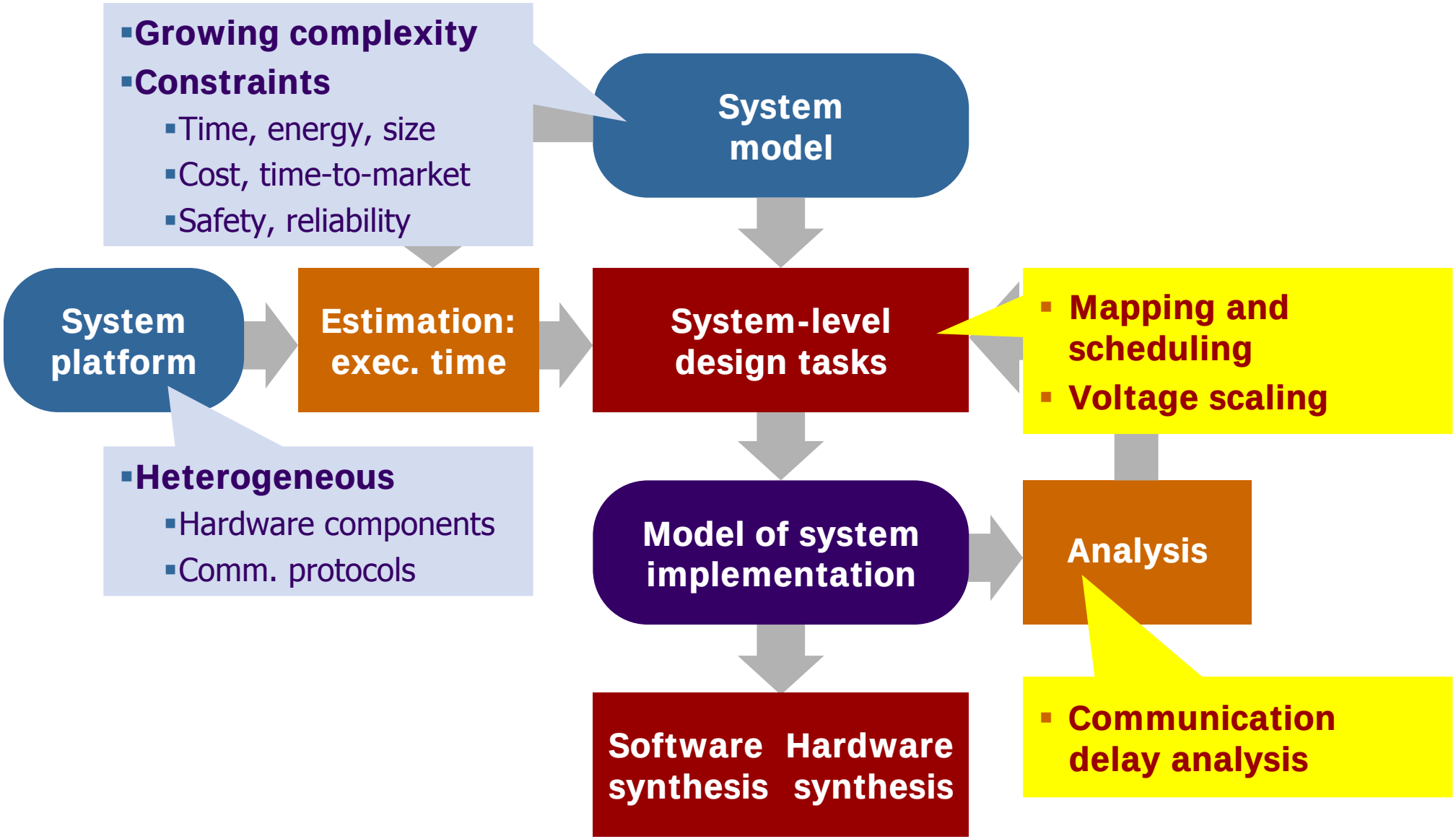
- Embedded systems
  - Example area: automotive electronics

→ Embedded systems design

- Optimization problems
  - Fault-tolerant mapping and scheduling
  - Voltage scaling
  - Communication delay analysis

- Assessment and message

Embedded Systems Design



Embedded System Design, Cont.

- Goal: automated **design optimization** techniques
  - Successfully manage the complexity of embedded systems
  - Meet the constraints imposed by the application domain
  - Shorten the time-to-market
  - Reduce development and manufacturing costs

Optimization: the key to successful design

Outline

- Embedded systems
  - Example area: automotive electronics
- Embedded systems design

→ Optimization problems
  - Fault-tolerant mapping and scheduling
  - Voltage scaling
  - Communication delay analysis

- Assessment and message

Optimization Problems

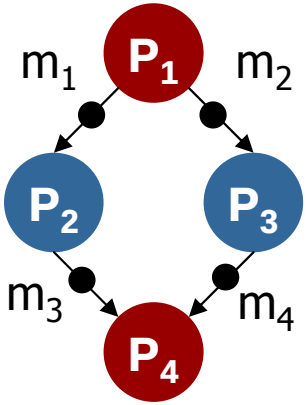
1. Mapping and scheduling
   - 1.1 Mapping to minimize communication
   - 1.2 Mapping and scheduling
   - 1.3 Fault-tolerant mapping and scheduling

2. Voltage scaling
   - 2.1 Continuous voltage scaling
   - 2.2 Discrete voltage scaling

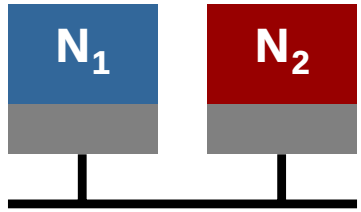
3. Communication delay analysis

Problem #1.1: Mapping

➔ Given
- Application: set of interacting processes
- Platform: set of nodes



➔ Determine
- Mapping of processes to nodes
  - Such that the communication is minimized

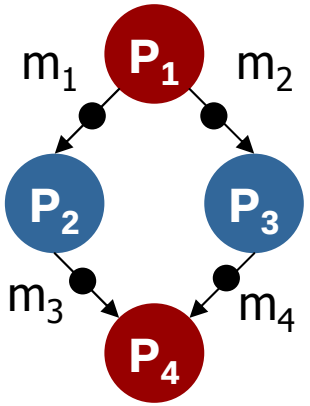


✱ Assessment
- Optimal solutions even for large problem sizes

Problem #1.2: Mapping and Scheduling

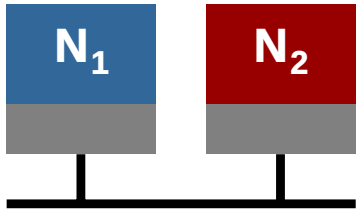
➔ Given

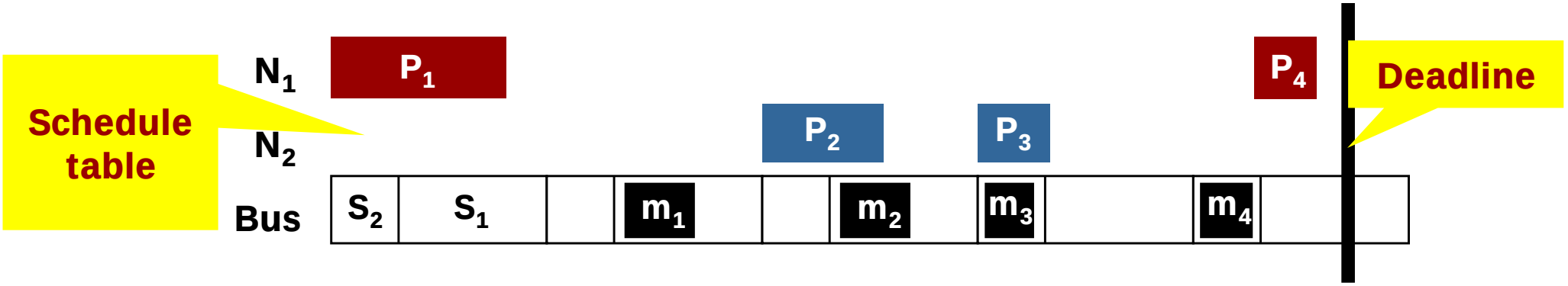
- Application: set of interacting processes
- Platform: set of nodes
- **Timing constraints:** deadlines



➔ Determine

- Mapping of processes and messages
- **Schedule tables** for processes and messages
  - Such that the timing constraints are satisfied

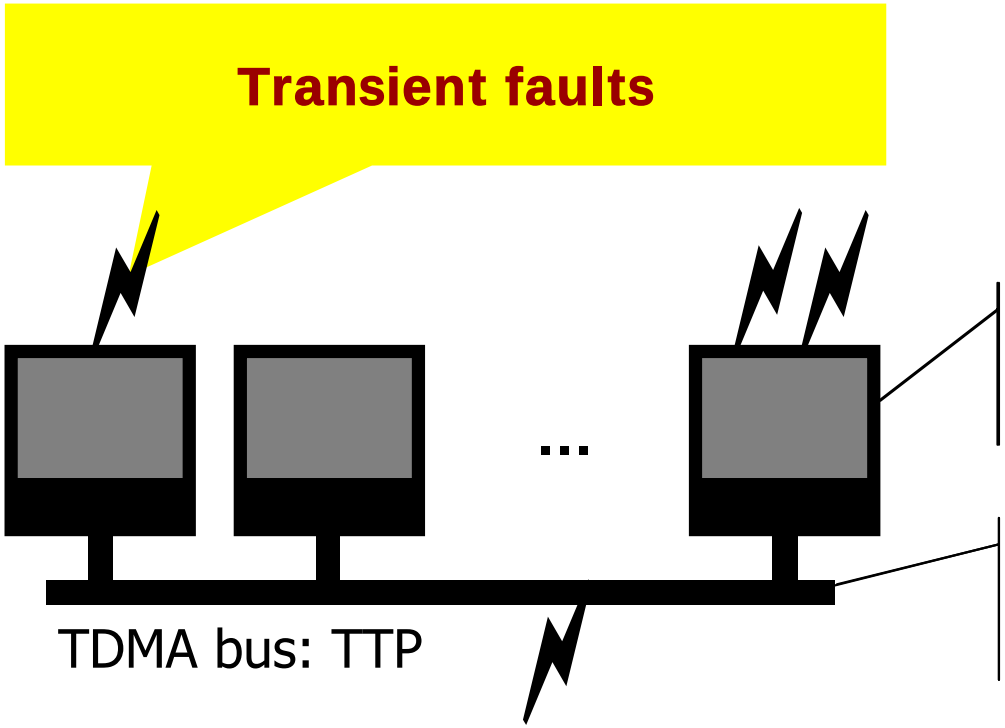




Problem #1.2: ∗Assessment

- Scheduling is NP-complete even in simpler context
  - D. Ullman, “NP-Complete Scheduling Problems”, Journal of Computer Systems Science, volume 10, pages 384–393, 1975.
- ILP formulation
  - Can’t obtain optimal solutions for large problem sizes
- Alternative: divide the problem
  - Scheduling
    - Heuristic: List scheduling
  - Mapping
    - Simulated annealing
    - Tabu-search
    - Problem-specific greedy algorithms

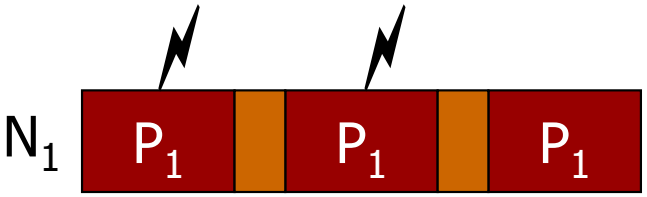
Fault-Tolerant Mapping and Scheduling

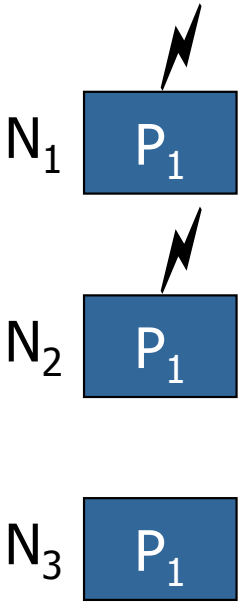


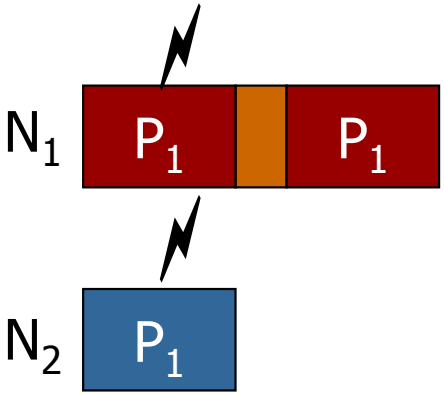
Processes:
Re-execution and replication

Messages:
Fault-tolerant protocol

Fault-Tolerance Techniques







Re-execution

Replication

Re-executed replicas

Problem #1.3: Formulation

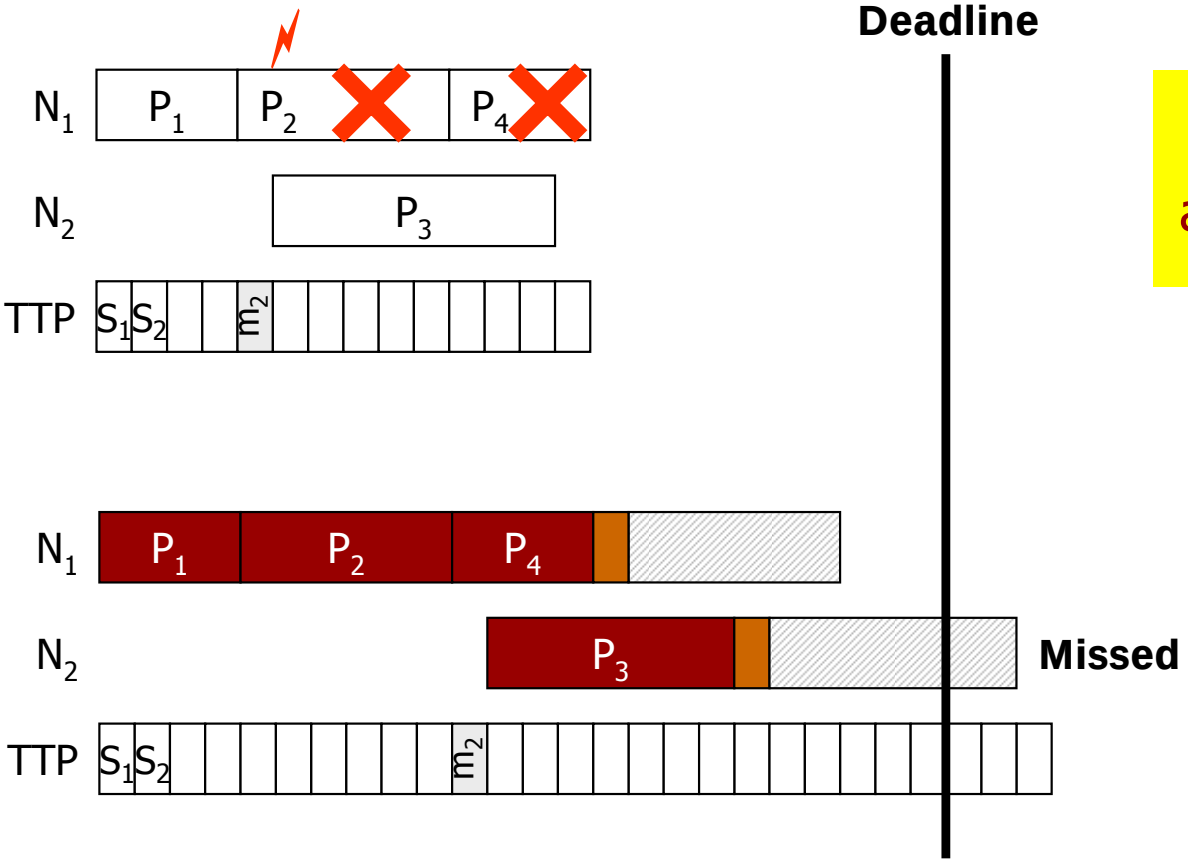
➔ Given

- Application: set of interacting processes
- Platform: set of nodes
- Timing constraints: deadlines
- **Fault model:** number of transient faults in the system period

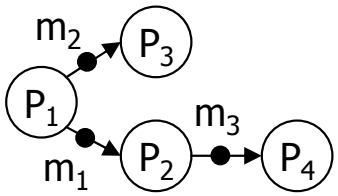
➔ Determine

- Mapping of processes and messages
- Schedule tables for processes and messages
- **Fault-tolerance policy assignment**
  - Such that the timing constraints are satisfied

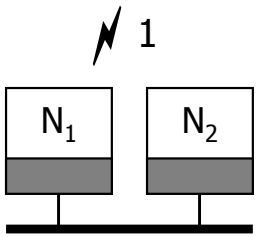
Fault-Tolerance Policy Assignment



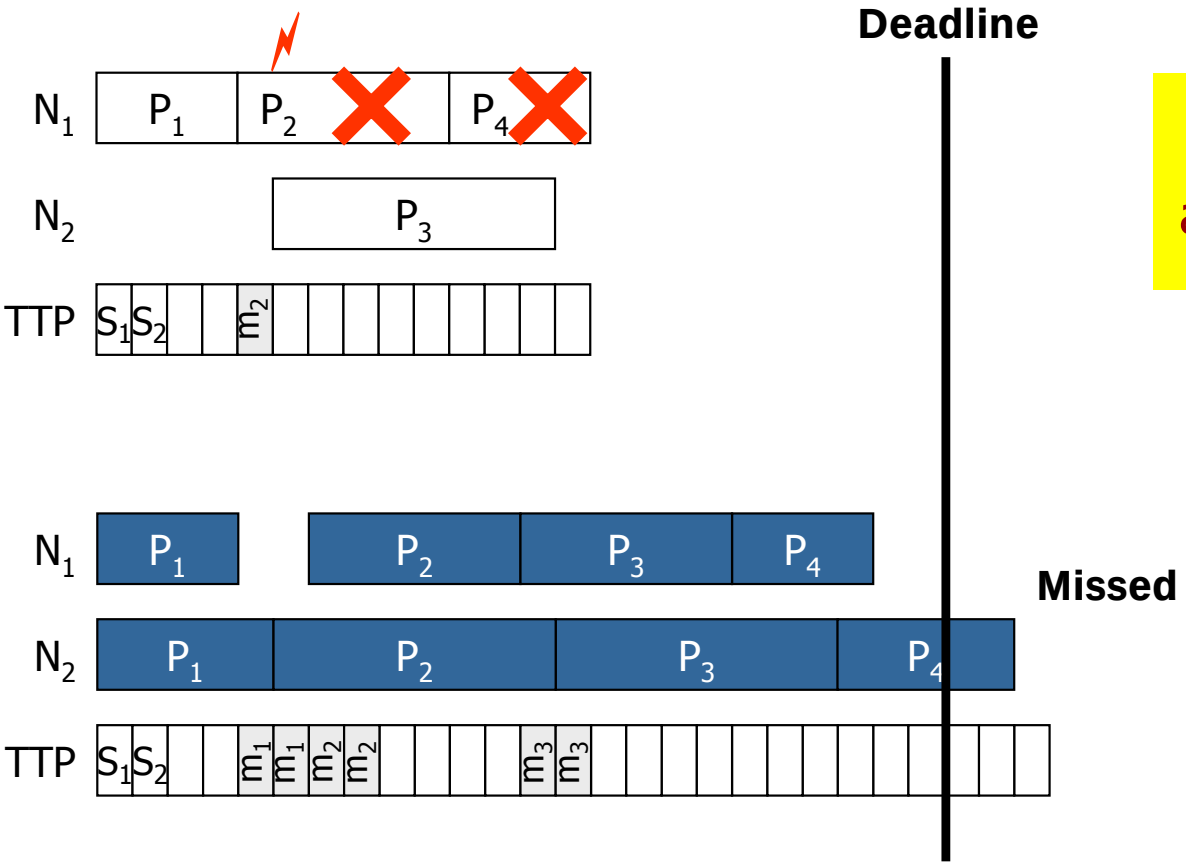
No fault-tolerance: application **crashes**



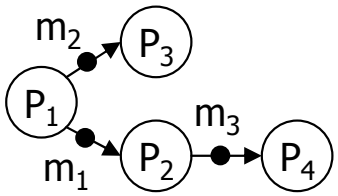
	N_1	N_2
P_1	40	50
P_2	60	80
P_3	60	80
P_4	40	50



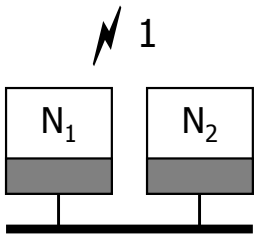
Fault-Tolerance Policy Assignment



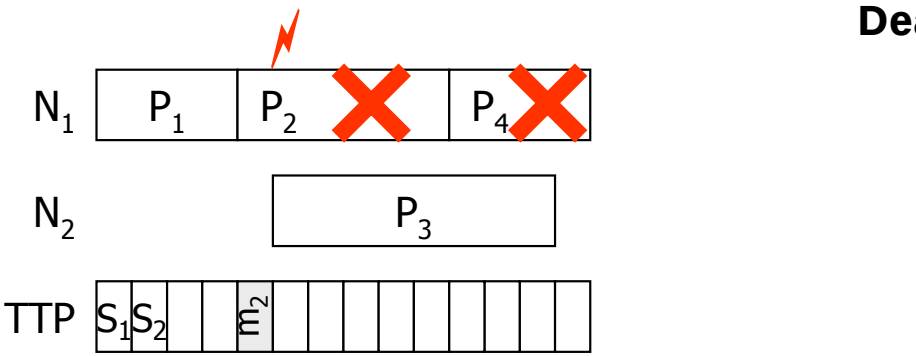
No fault-tolerance: application **crashes**



	N_1	N_2
P_1	40	50
P_2	60	80
P_3	60	80
P_4	40	50

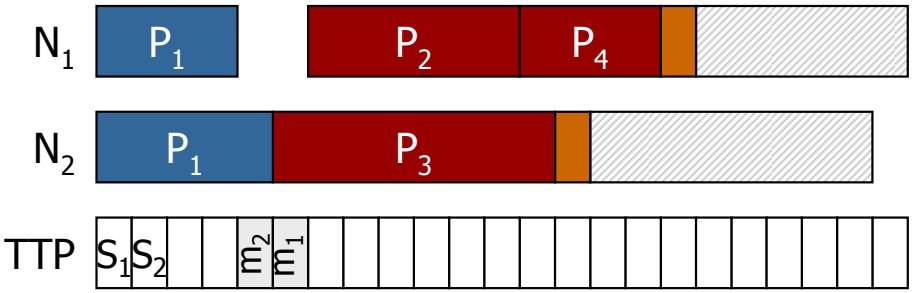


Fault-Tolerance Policy Assignment



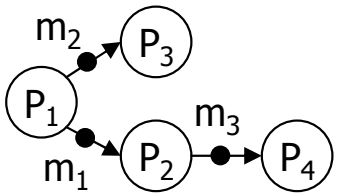
Deadline

No fault-tolerance: application **crashes**

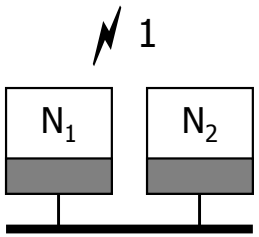


Met

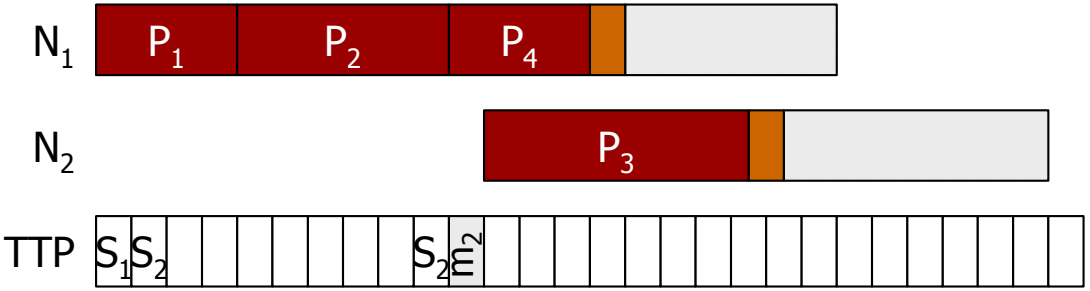
Optimization of fault-tolerance policy assignment



	N_1	N_2
P_1	40	50
P_2	60	80
P_3	60	80
P_4	40	50



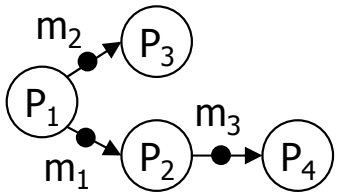
Tabu-Search: Policy Assignment & Mapping



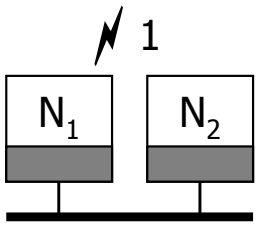
	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

Current solution

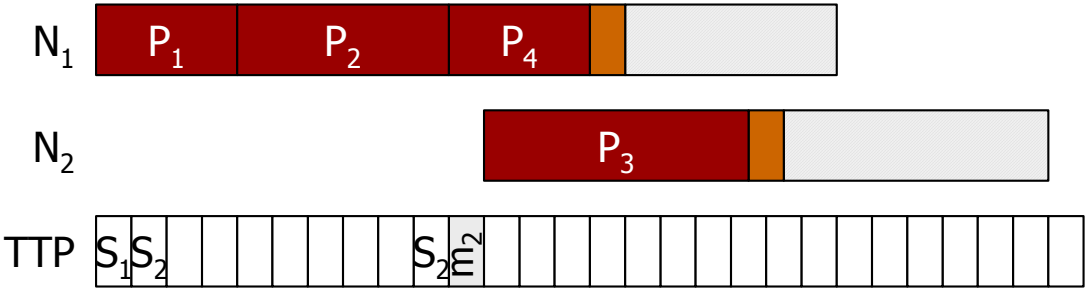
Design transformations



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



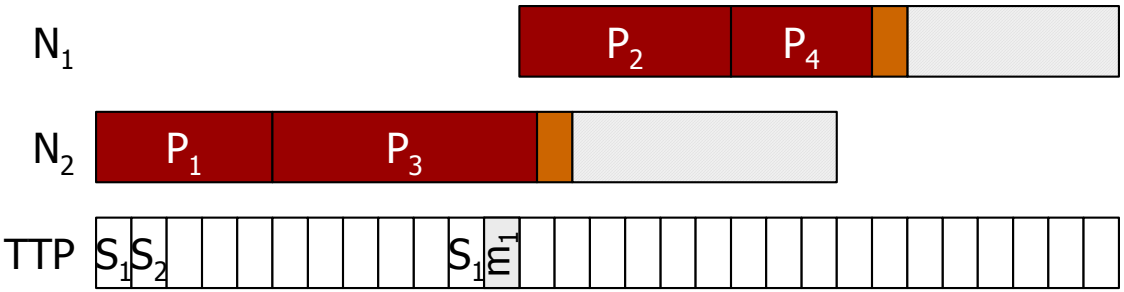
Tabu-Search: Policy Assignment & Mapping



	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

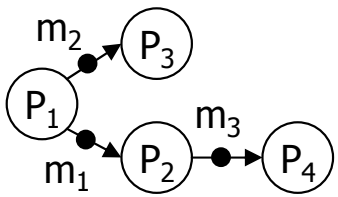
Current solution

Design transformations

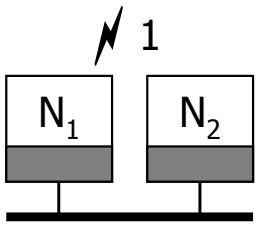


	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

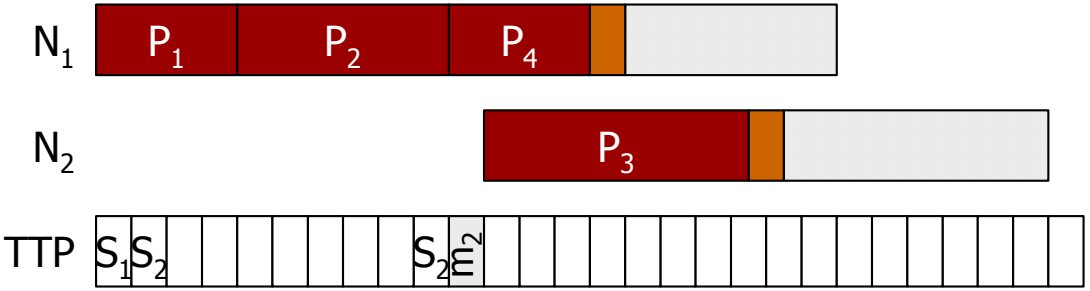
Tabu move & **worse** than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



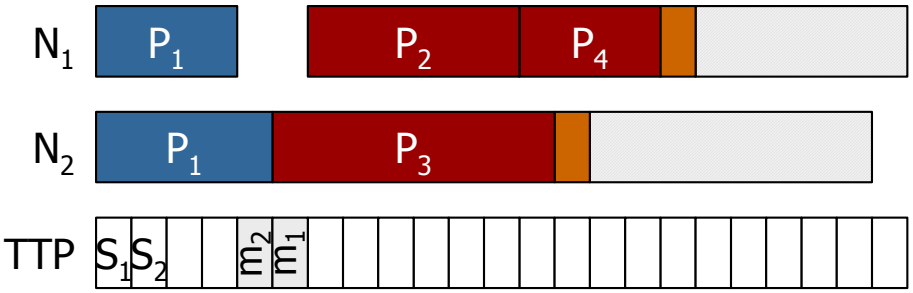
Tabu-Search: Policy Assignment & Mapping



	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

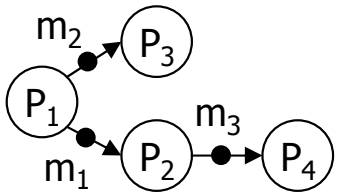
Current solution

Design transformations

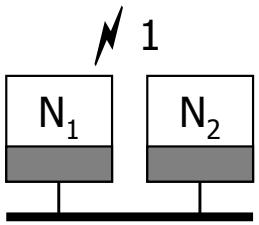


	P_1	P_2	P_3	P_4
Tabu	2	1	0	0
Wait	0	0	2	1

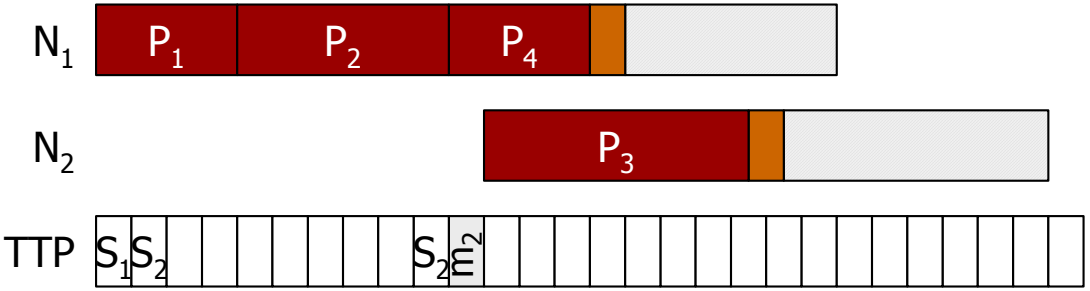
Tabu move & **better** than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



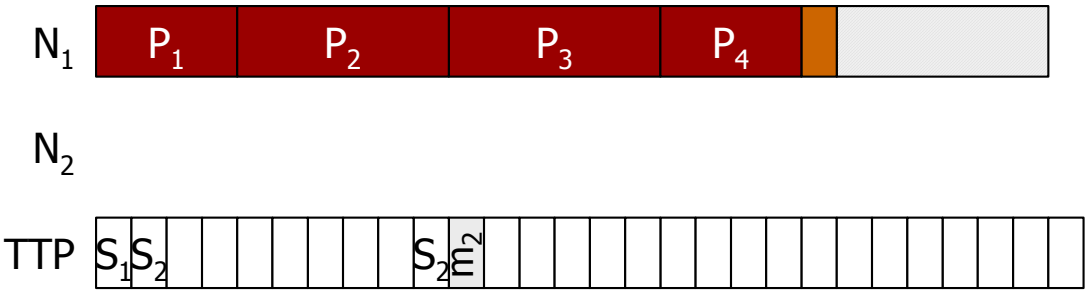
Tabu-Search: Policy Assignment & Mapping



	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

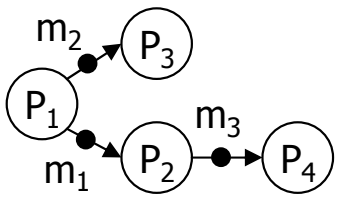
Current solution

Design transformations

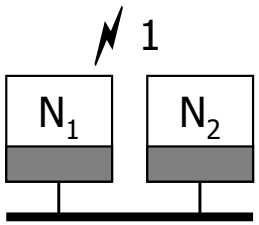


	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

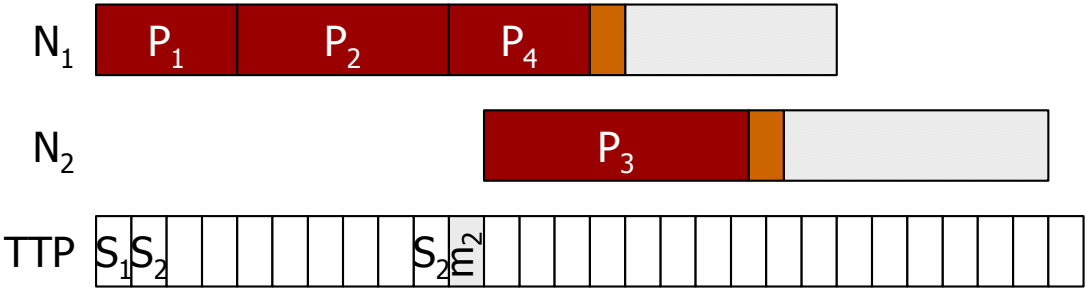
Non-tabu & **worse** than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



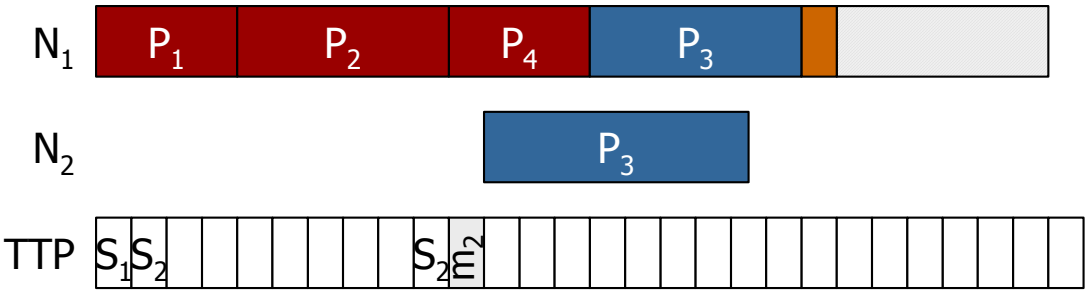
Tabu-Search: Policy Assignment & Mapping



	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

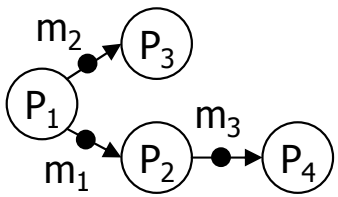
Current solution

Design transformations

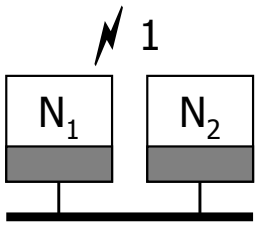


	P_1	P_2	P_3	P_4
Tabu	1	2	0	0
Wait	1	0	1	1

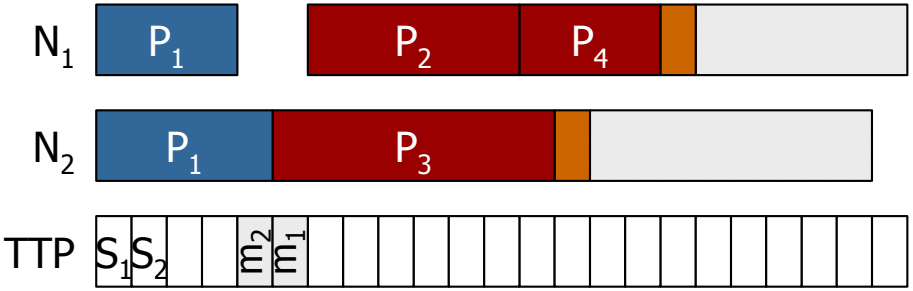
Non-tabu & **worse** than best-so-far



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50



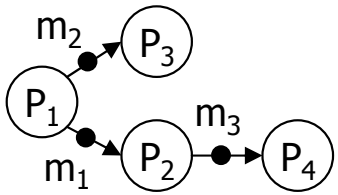
Tabu-Search: Policy Assignment & Mapping



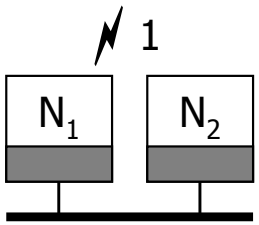
	P_1	P_2	P_3	P_4
Tabu	2	1	0	0
Wait	0	0	2	1

Current solution

Design transformations



	N_1	N_2
P_1	40	50
P_2	60	75
P_3	60	75
P_4	40	50

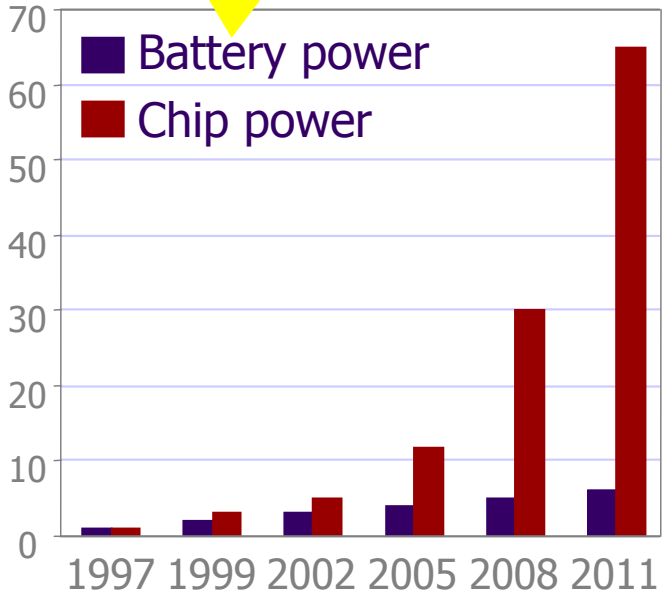


Problem #2: Voltage Scaling

- **GSM Phone:**
  - Search
  - Radio link control
  - Talking
- **MP3 Player**
- **Digital Camera:**
  - Take photo
  - Restore photo

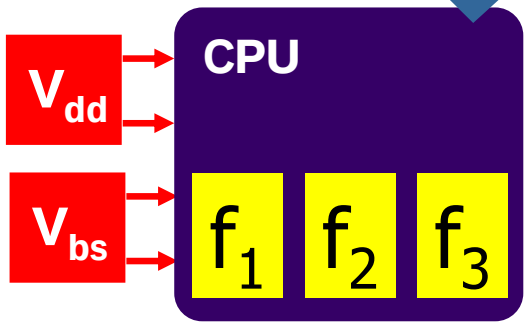
Timing constraints

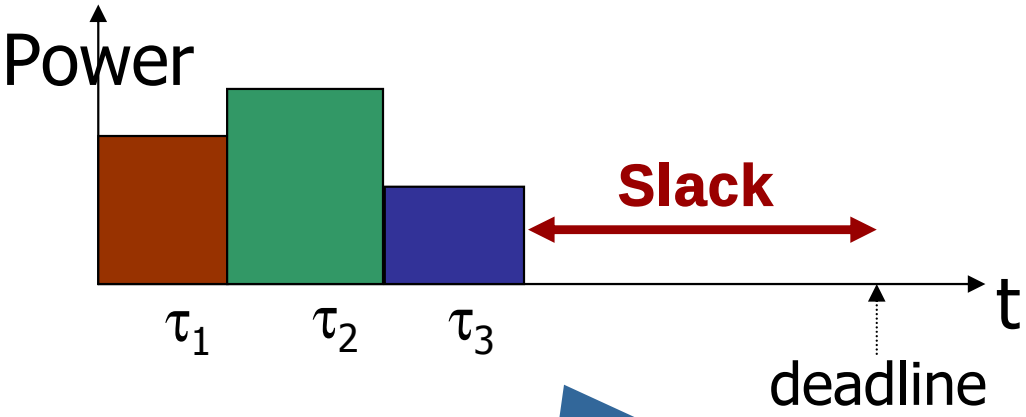
Power constraints



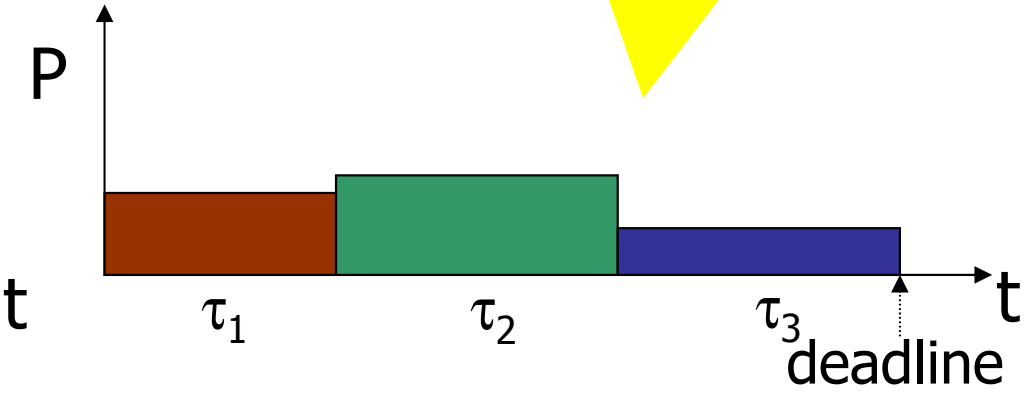
Problem #2: Voltage Scaling

Different voltages: different frequencies





Energy/speed trade-offs: varying the voltages

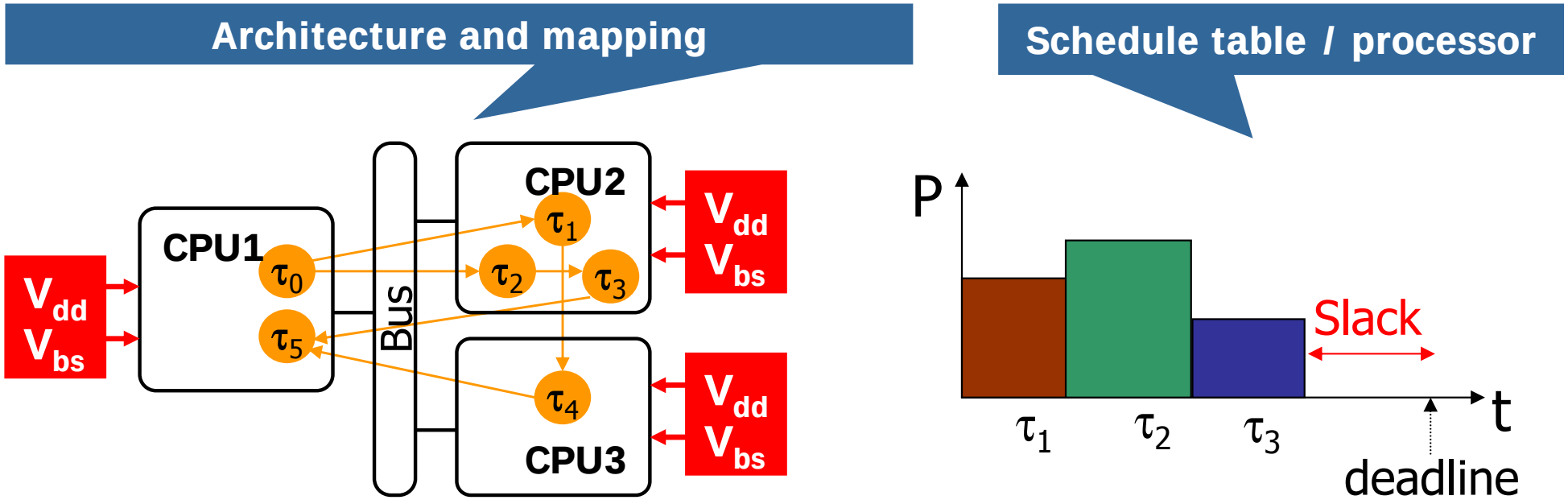


Mapping and scheduling: given (fastest freq.)

Problem #2.1: Continuous Voltage Scaling

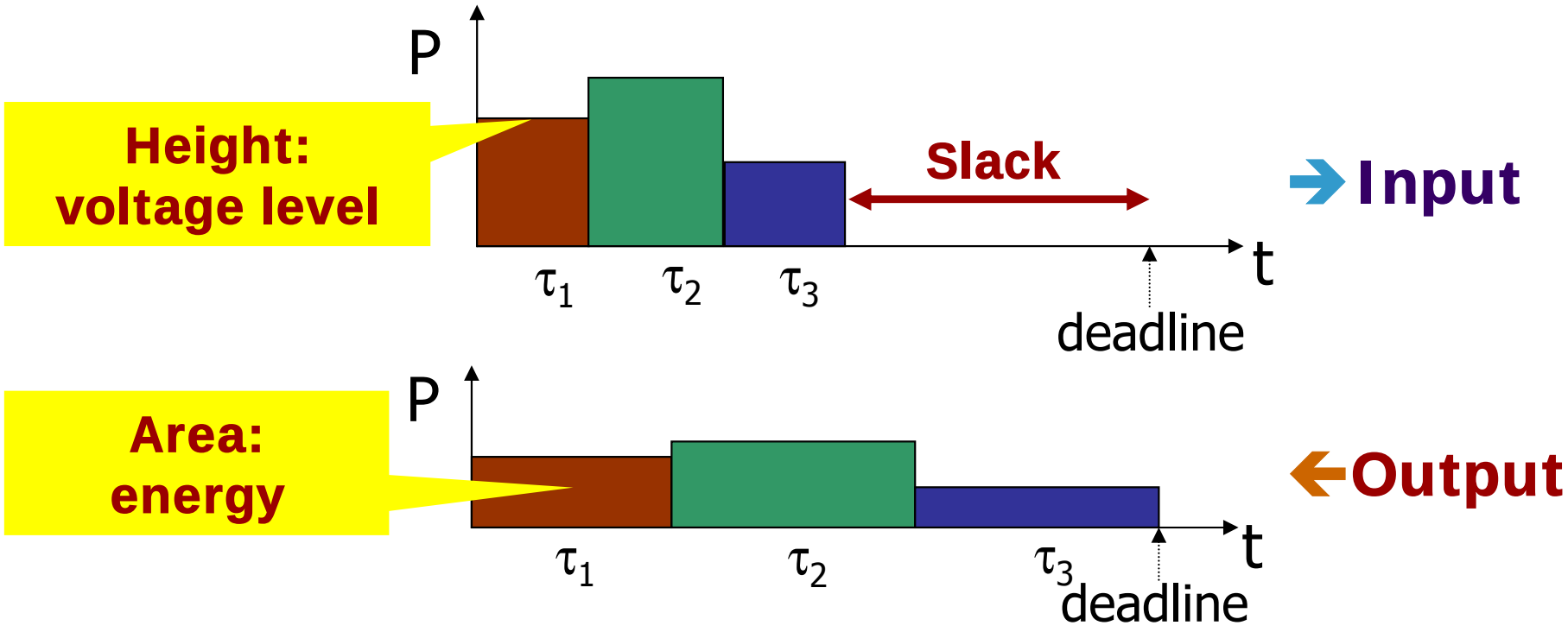
➔ Given

- Application: set of interacting processes
- Platform: set of nodes, each having **supply voltage (V_{dd}) and body bias voltage (V_{bs}) inputs**
- Mapping and schedule table (including timing constraints)



Problem #2.1: Continuous Voltage Scaling

- Determine
  - **Voltage levels** V_{dd} and V_{bs} for each process
    - Such that system **energy is minimized** and
    - Deadlines are satisfied



Problem #2.1: Continuous Voltage Scaling

- Determine
  - **Voltage levels** V_{dd} and V_{bs} for each process
    - Such that system **energy is minimized** and
    - Deadlines are satisfied

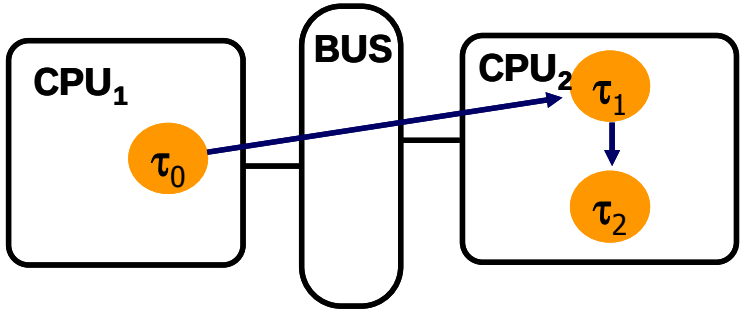
- Assessment
  - **Convex nonlinear problem**
    - Polynomial time solvable with an arbitrary good precision
    - A. Andrei, "Overhead-Conscious Voltage Selection for Dynamic and Leakage Energy Reduction of Time-Constrained Systems", technical report, Linköping University, 2004

Problem #2.1: Formulation

- Minimize energy
  - $E[\tau_0] + E[\tau_1] + E[\tau_2] + E\_OH[\tau_1\text{-}\tau_2]$

Energy due to processes

Overhead due to voltage changes



- Such that
  - $T_{start}[\tau_0] + T_{exe}[\tau_0] \leq T_{start}[\tau_1]$
  - $T_{start}[\tau_1] + T_{exe}[\tau_1] + T_{oh}[\tau_1\text{-}\tau_2] \leq T_{start}[\tau_2]$

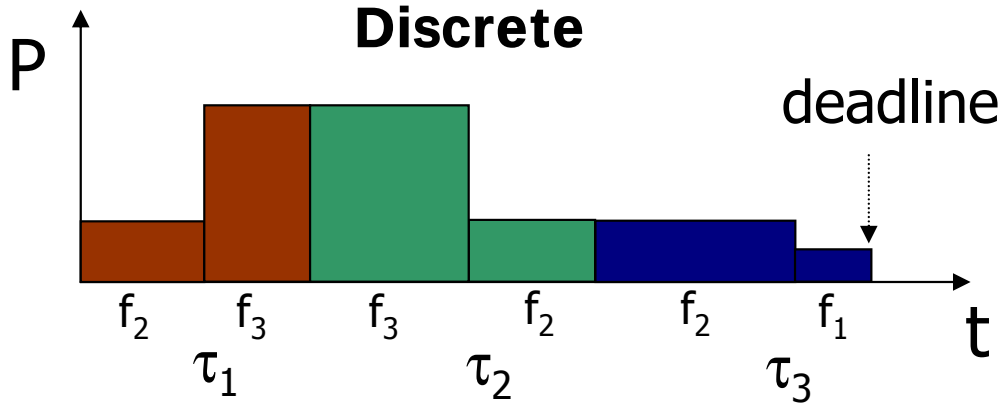
Precedence relationships

  - $T_{start}[\tau_2] + T_{exe}[\tau_2] \leq DL[\tau_2]$

Deadlines

Problem #2.2: Discrete Voltage Scaling

- Problem formulation
  - ➔ Given **discrete execution frequencies**
    Processors can operate using a frequency from a fixed discrete set
    Changing the frequency incurs a delay and an energy penalty
  - ➔ Determine the **set of frequencies** for each task
    - Such that system energy is minimized and
    - Deadlines are satisfied



Problem #2.2: Example

- Given
  - 1 processor: f∈{50, 100, 150} MHz
  - 3 processes
    - τ_1: P={10, 20, 30} mW, dl=1ms, NC=100 cycles
    - τ_2: P={12, 22, 32} mW, dl=1.5ms , NC=100 cycles
    - τ_3: P={15, 25, 35} mW, dl=2ms , NC=100 cycles
  - Schedule: execution order is τ_1, τ_2, τ_3
- Determine
  - For each process, number of clock cycles to be executed at each frequency

$$(c_1^1, c_1^2, c_1^3), (c_2^1, c_2^2, c_2^3), (c_3^1, c_3^2, c_3^3)$$

    - such that the energy is minimized

Problem #2.2: Example, Cont.

Assessment:

- **Strongly NP-hard problem**
  - The frequencies are now a set of integers; identical to:
  - P. De, "Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks", Operations Research, 45(2):302–306, March 1997.
- MILP formulation for the optimal solution

$$c_i^1 + c_i^2 + c_i^3 = NC_i$$

Each task has to execute the given number of cycles

$$\frac{c_i^1}{f^1} + \frac{c_i^2}{f^2} + \frac{c_i^3}{f^3} = t_i$$

Task execution time

$$start_i + t_i \leq t_{i+1}$$

Precedence constraints

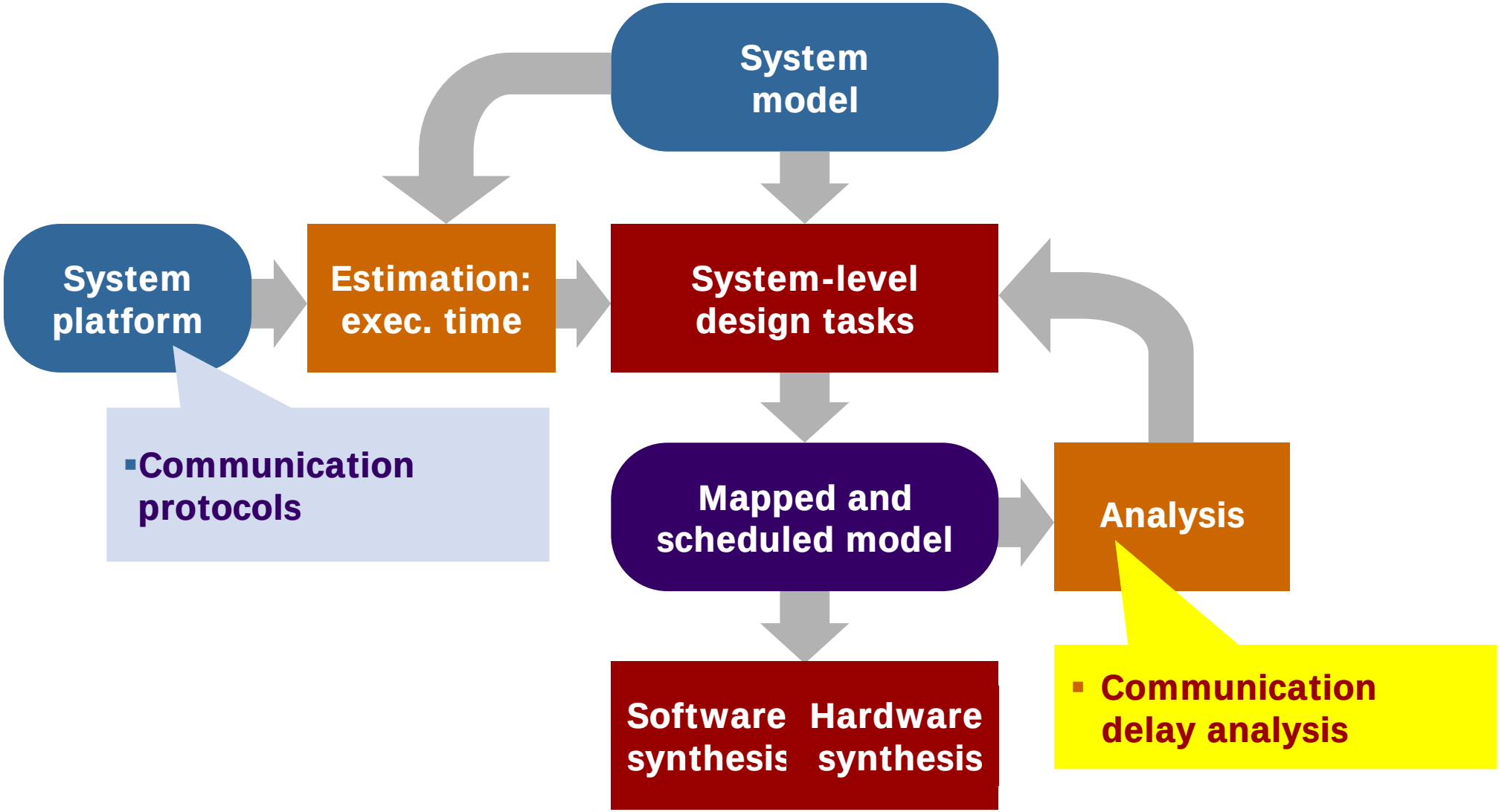
$$start_i + t_i \leq dl_i$$

Deadline constraints

$$\sum_i \frac{c_i^1}{f^1} \cdot P_i^1 + \frac{c_i^2}{f^2} \cdot P_i^2 + \frac{c_i^3}{f^3} \cdot P_i^3$$

Minimize energy

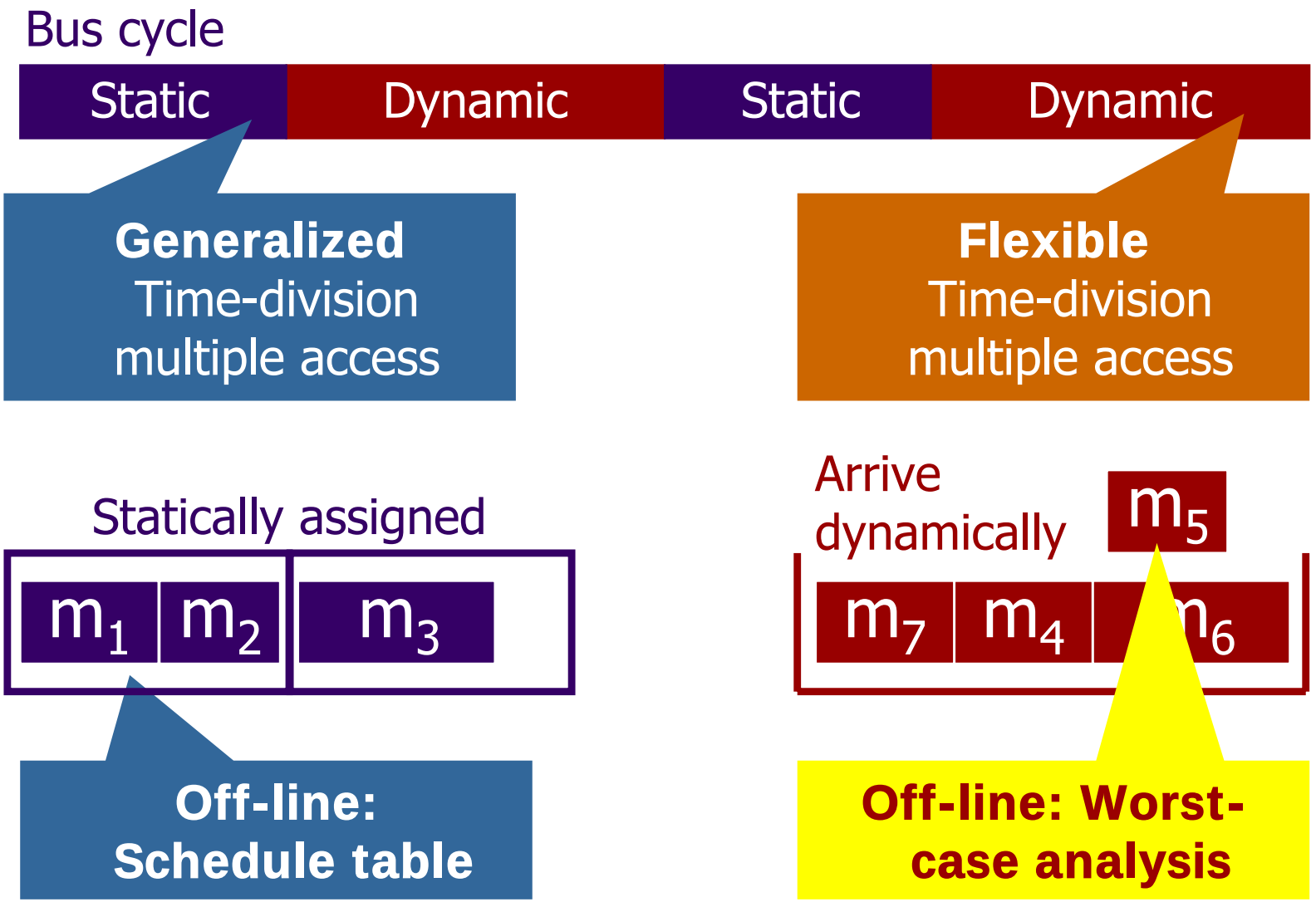
Embedded Systems Design



Problem #3: FlexRay Analysis

- FlexRay communication protocol
  - Becoming **de-facto standard** in automotive electronics
    - BMW, DaimlerChrysler, General Motors, Volkswagen, Bosch, Motorola, Philips
  - Deterministic data transmission, fault-tolerant, high data-rate
- Problem
  - ➔ Given
    - Application: set of interacting processes
    - Platform: set of nodes **connected by FlexRay**
    - **Implementation:** Mapping and scheduling
  - ➔ Determine
    - **Worst-case** communication delays for messages

Problem #3: FlexRay Analysis, Cont.



Problem #3: Formulation and Example

- Given
  - FlexRay bus
    - Length of the static phase
    - Length of the dynamic phase
  - Dynamically arriving messages
    - Priorities
- Determine for each message
  - Worst-case communication delay

Priority

m_4

m_7

m_6

m_5

Analyze this!

Problem #3: ∗Assessment

- ”Classic” bin covering problem
  - ➔ Given
    - Set of bins of fixed integer size
    - Set of items of integer size
  - ➔ Determine
    - **Maximum** number of bins that can be filled with the items
  - ∗ Assessment
    - Asymptotic fully polynomial time approximation

- FlexRay dynamic phase analysis ≠ ”classic” bin covering
  - Bins have an **upper limit**: size of the dynamic phase
  - Assessment
    - Approximation algorithm does not exist
    - MILP formulation feasible up to 60 messages

Wanted:
better solution

Outline

- Embedded systems
  - Example area: automotive electronics
- Embedded systems design
- Optimization problems
  - Fault-tolerant mapping and scheduling
  - Voltage scaling
  - Communication delay analysis

→ Assessment and message

Message

- Optimization
  - Key to successful embedded systems design

- **Challenges**
  - Classify the problems
  - Divide the problem into sub-problems
  - Formulate the problems
  - Solve the problems optimally
  - Fast and accurate heuristics for specific problems