Scheduling and Voltage Scaling for Energy/Reliability Trade-offs in Fault-Tolerant Time-Triggered Embedded Systems

Kåre Harbo Poulsen

August 23, 2007

Outline

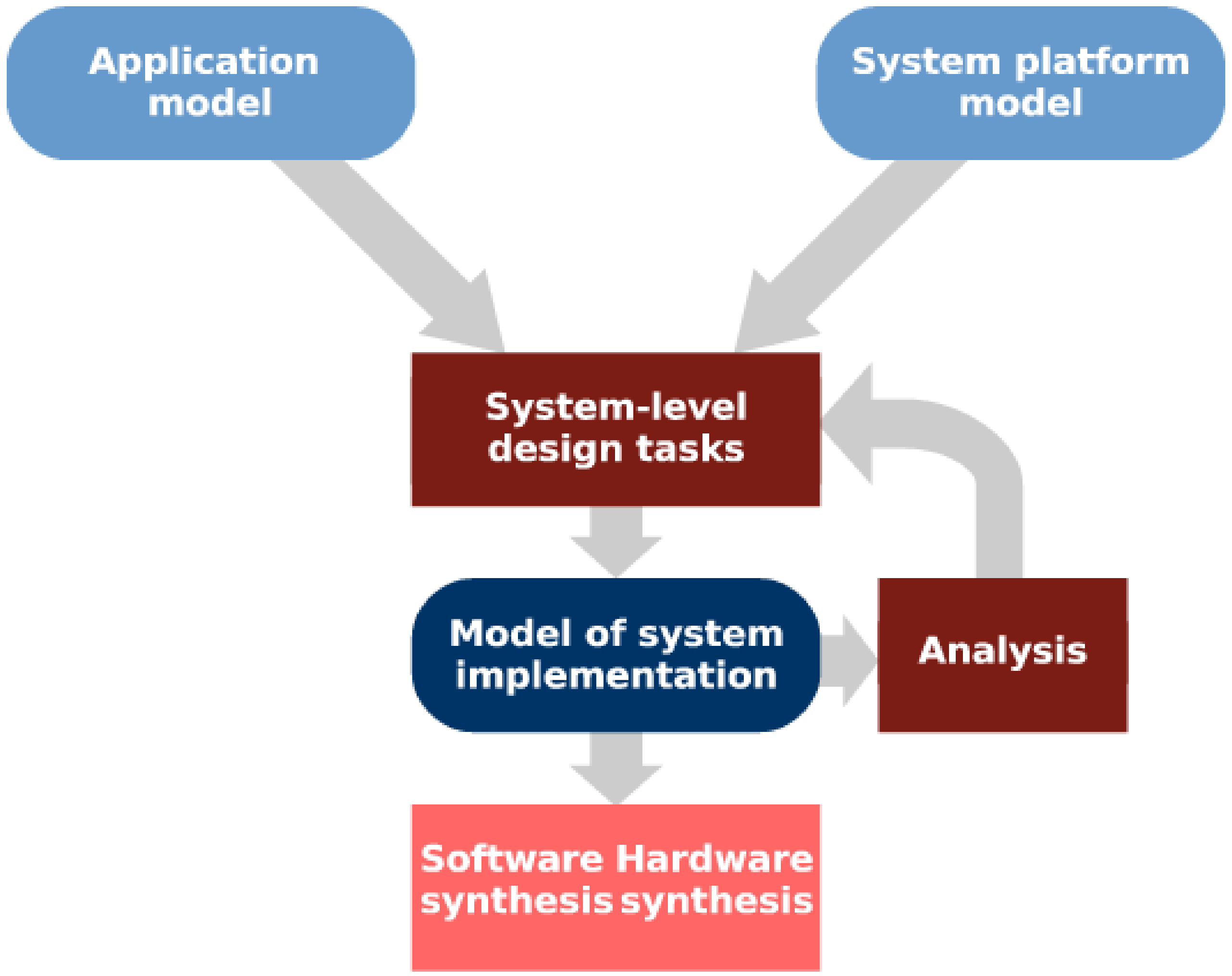
- *Introduction*
- Motivation
- Problem Formulation
- Implementation
- Experiments and Results
- Conclusions
- Q&A

Embedded Systems

Embedded Systems

- Embedded MP-SoCs
  - Single purpose
  - Real-time
  - Reliable
  - Low power

Design of Embedded Systems

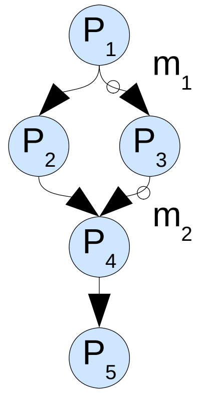


Summary

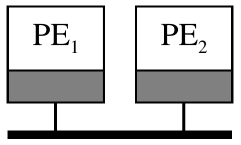
- Design tool for Embedded MP-SoCs
  - Schedule and Mapping
  - Timing constraints
  - Fault tolerant / reliable
  - Energy efficient

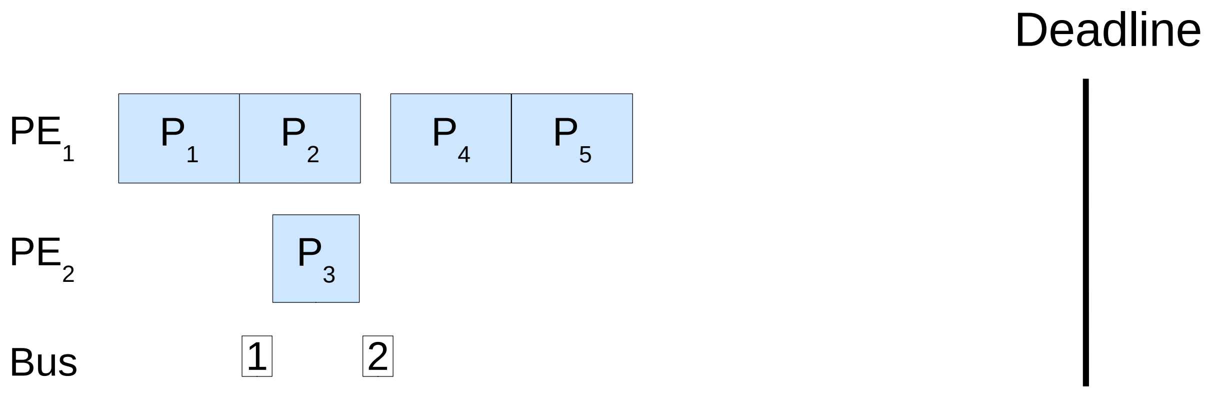
Embedded Systems Model

- Input
  - Application
  - Architecture
- Output
  - Mapping
  - Schedule

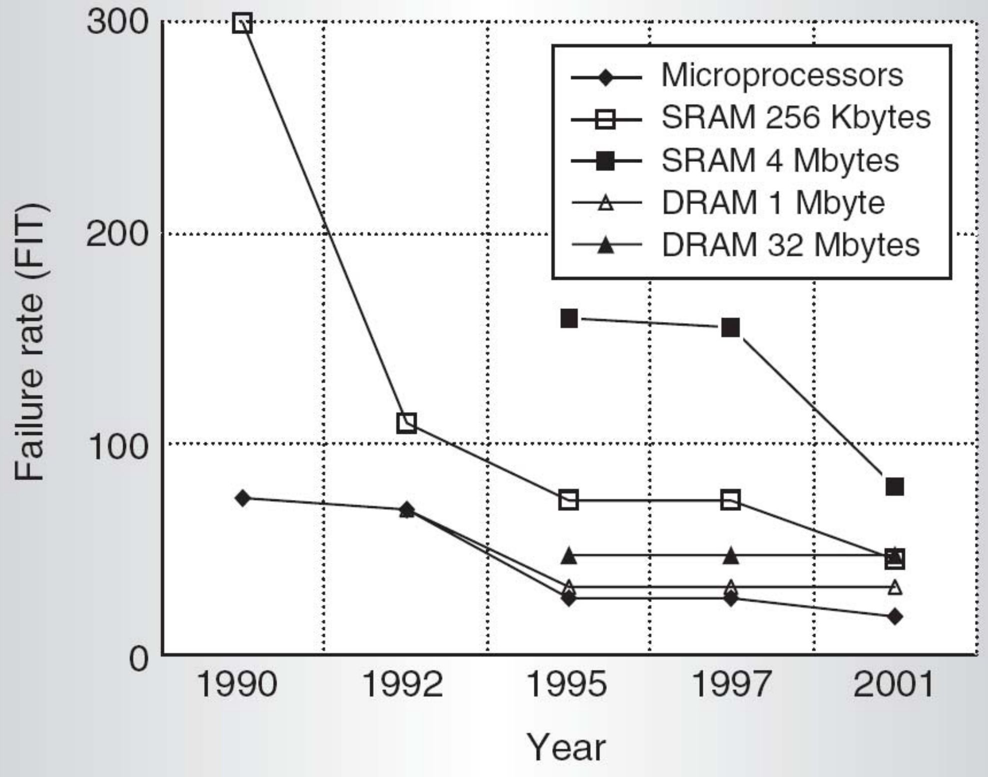


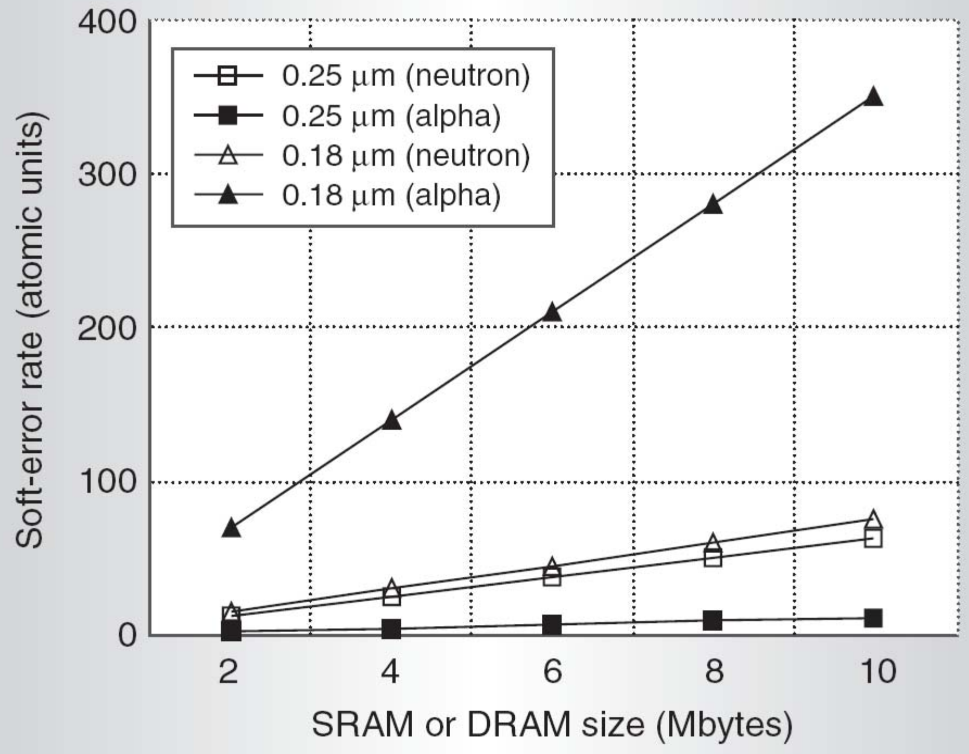
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4





Faults



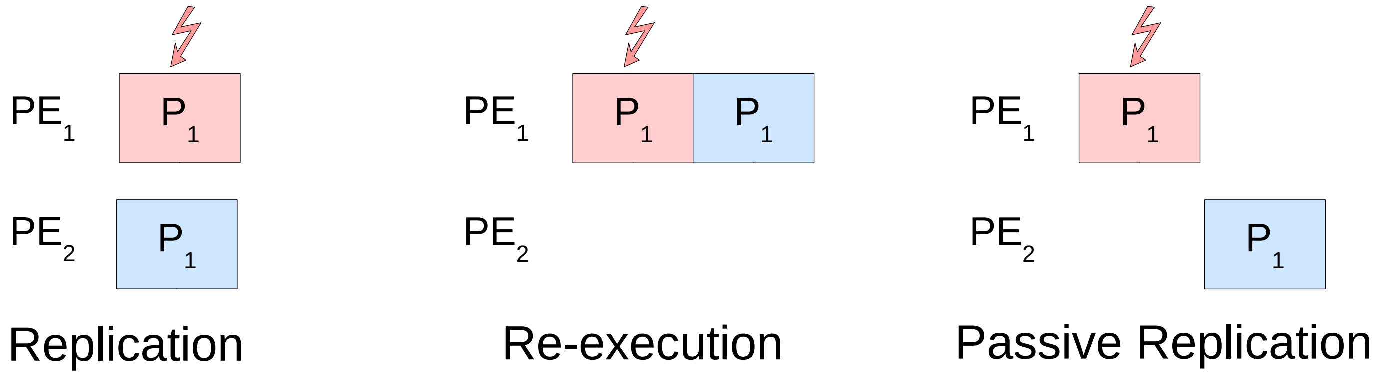


- Permanent faults are decreasing
- Transient faults are increasing

From: Cristian Continescu, Trends and challenges in VLSI circuit reliability, 2003

Fault-Tolerance

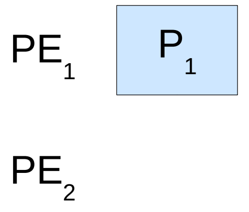
- Tolerate faults gracefully
- Expressions for reliability for fault-tolerance



Reliability

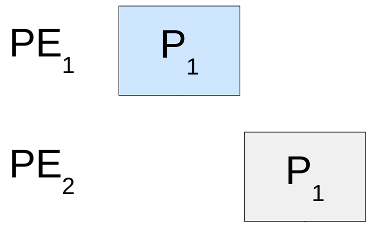
$$R_{single} = e^{-\lambda c} = 1 - \rho$$

Single execution:



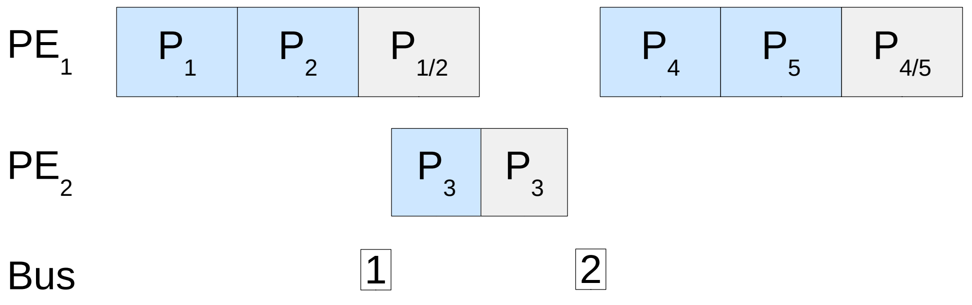
$$R_{FT} = 1 - \prod_{i=1}^{k} (1 - R_i)$$

Fault-tolerance:



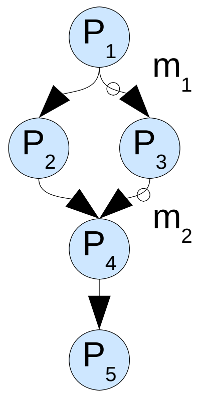
$$R_{App} = \prod_{P_i \in A} R_{P_i}$$

Application

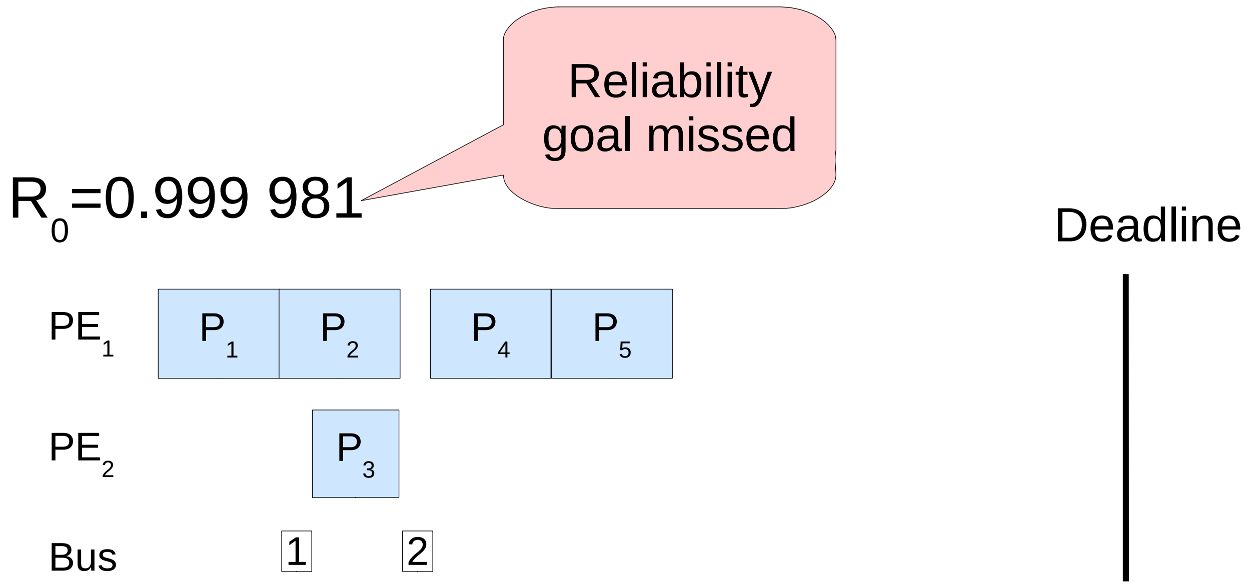


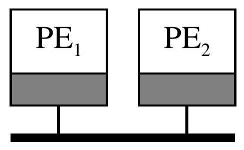
Embedded Systems Model

- Input
  - Application
  - Architecture
  - Reliability goal: 0.999 999 999



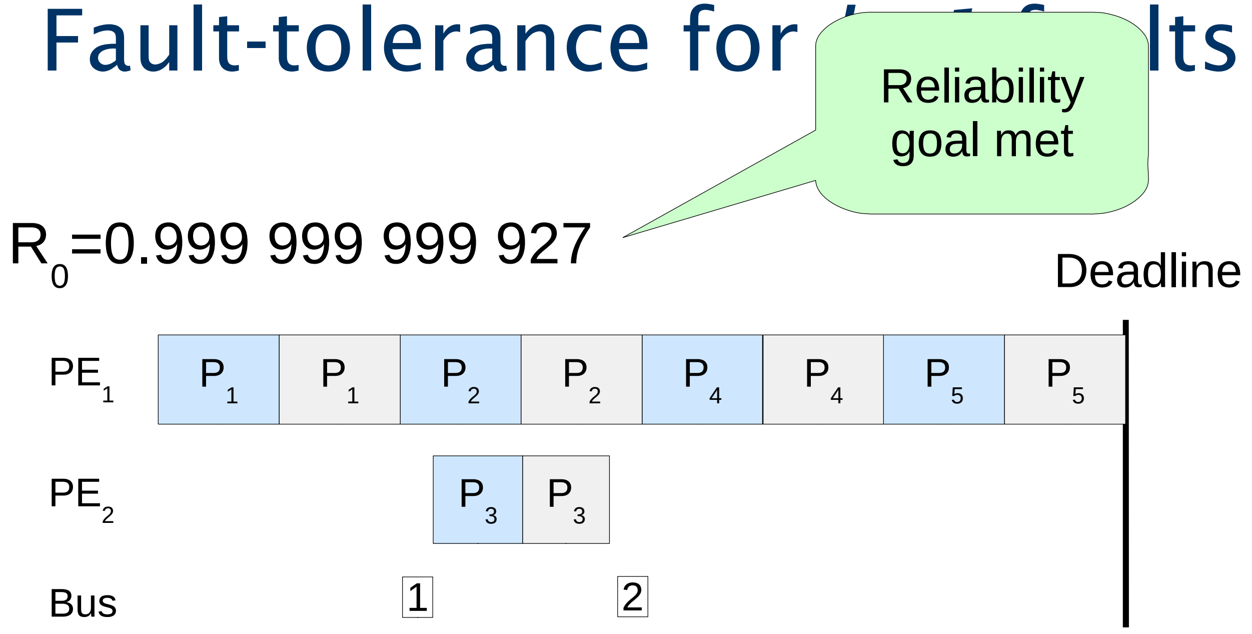
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

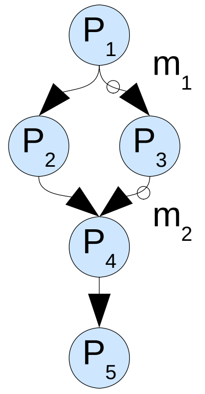




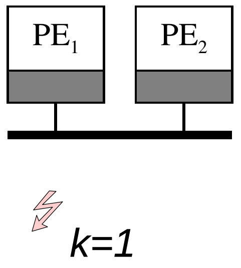
Fault-Tolerant Scheduling

- Input
  - Application
  - Architecture
  - Reliability goal: 0.999 999 999
- Fault-tolerance for k=1 faults



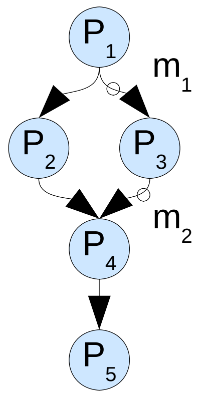


	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

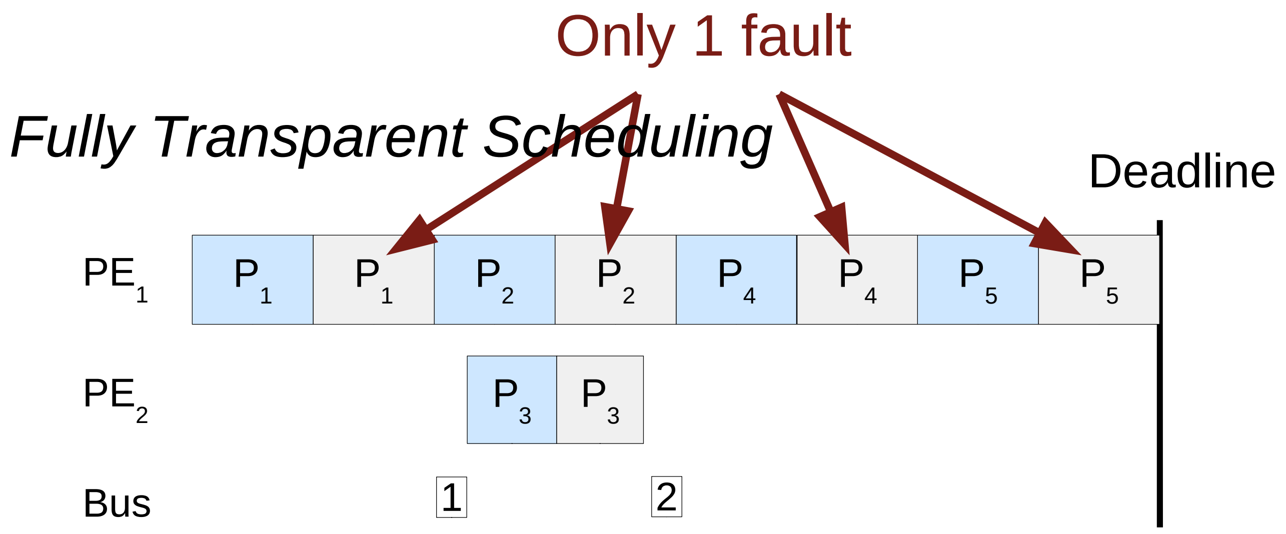


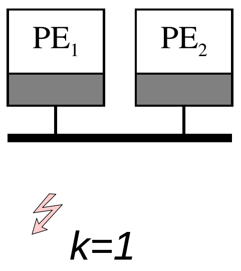
Fault-Tolerant Scheduling

- Fault tolerant scheduler
  - Full transparency
    - Good debugability
    - Little memory



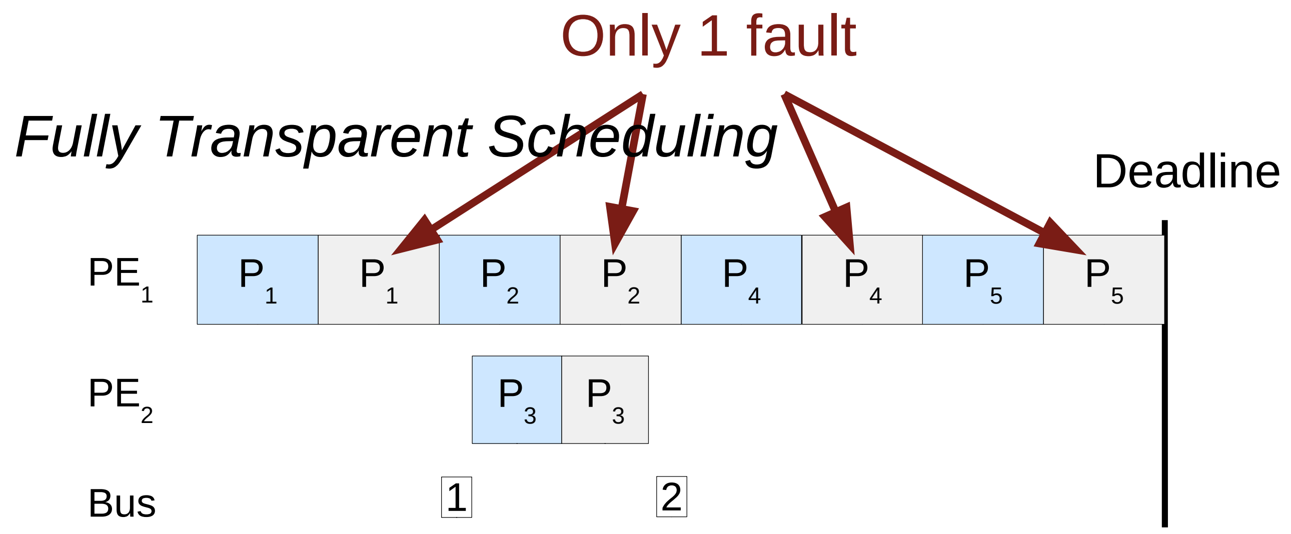
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

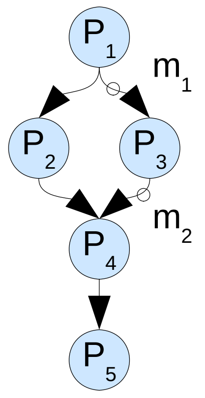




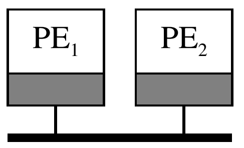
Fault-Tolerant Scheduling

- Can be done faster
  - Sacrifice local transparency





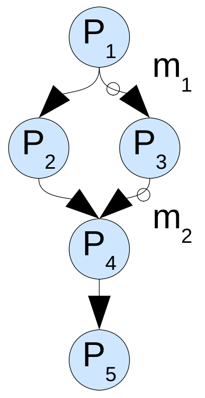
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4



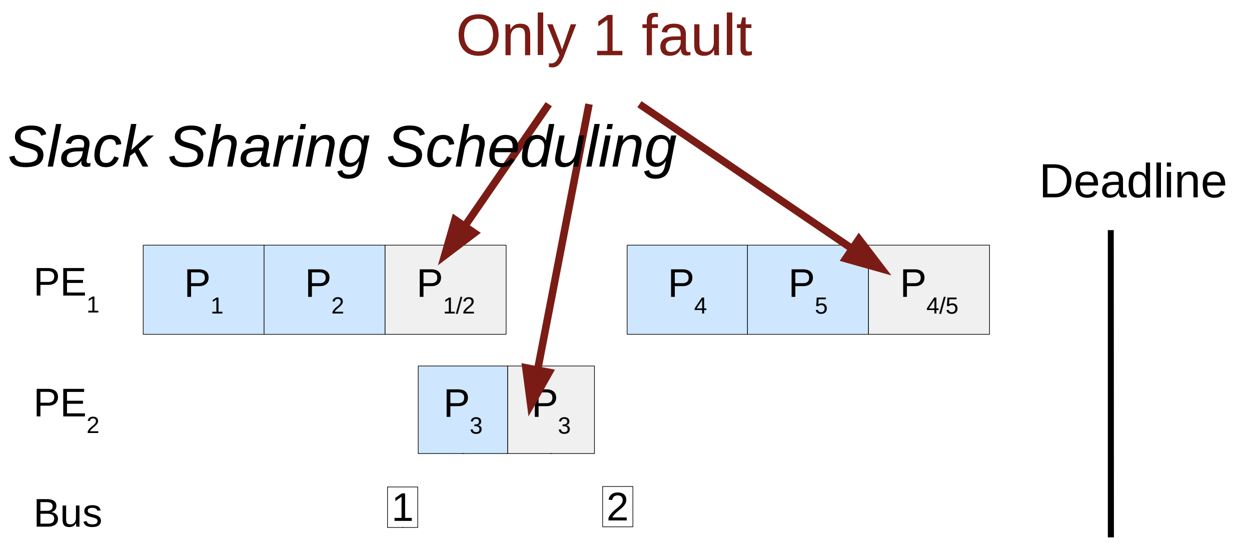
k=1

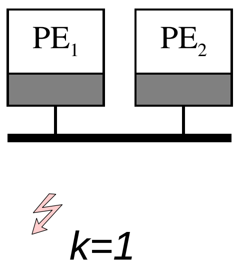
Fault-Tolerant Scheduling

- Can be done faster
  - Sacrifice local transparency
  - More complex online scheduler



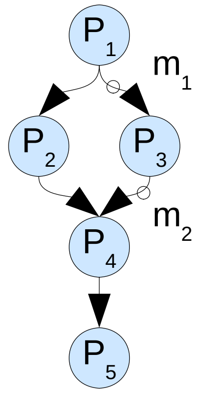
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4





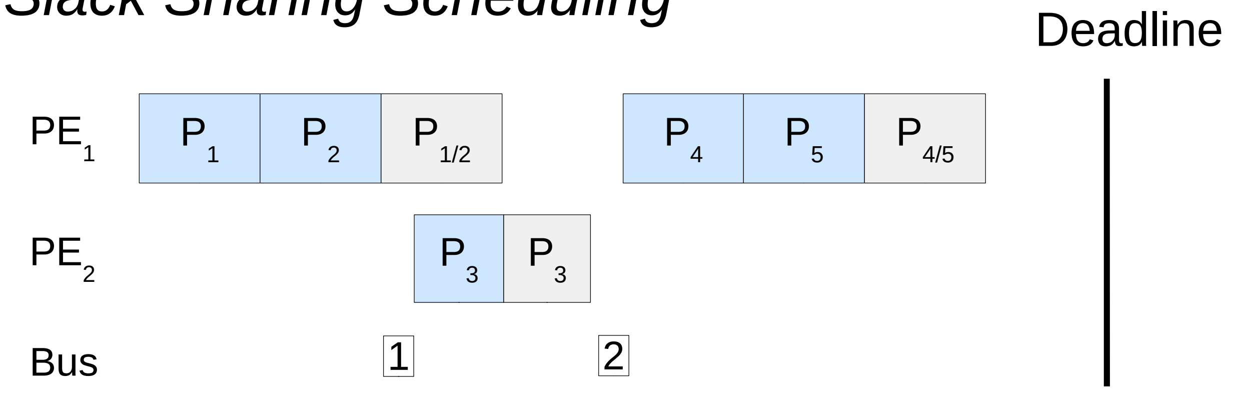
Fault-Tolerant Scheduling

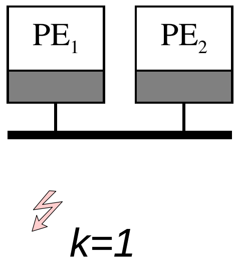
- Even faster
  - Sacrifice all transparency
  - Schedule for each fault scenario



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

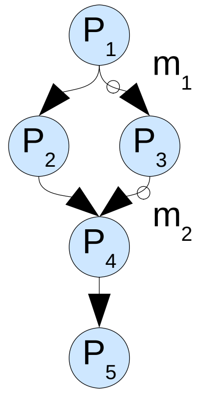
Slack Sharing Scheduling



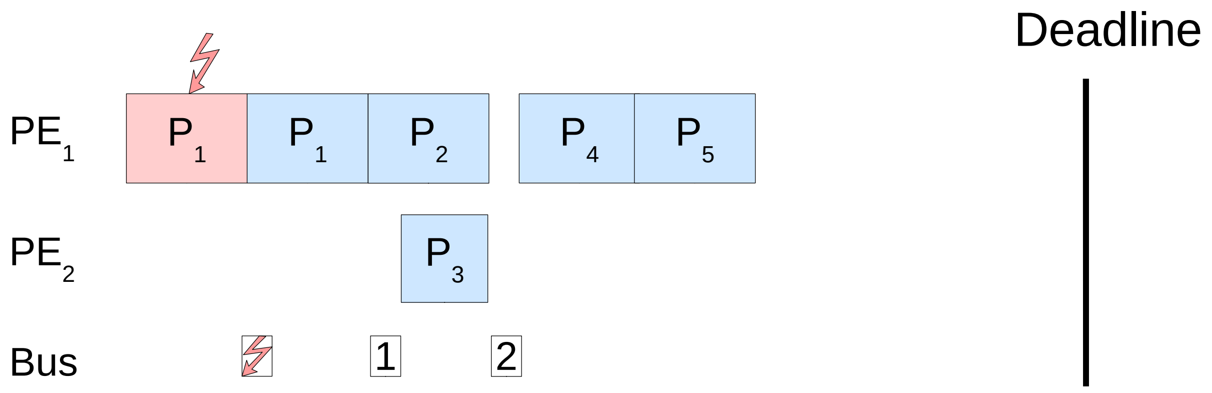


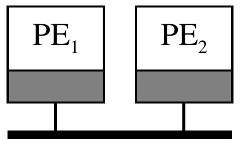
Fault-Tolerant Scheduling

- Even faster
  - Sacrifice all transparency
  - Schedule for each fault scenario
  - At most *k* re-executions



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

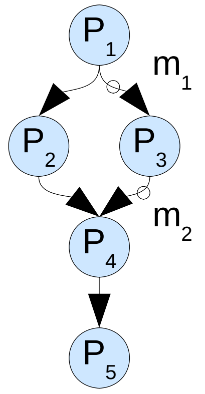




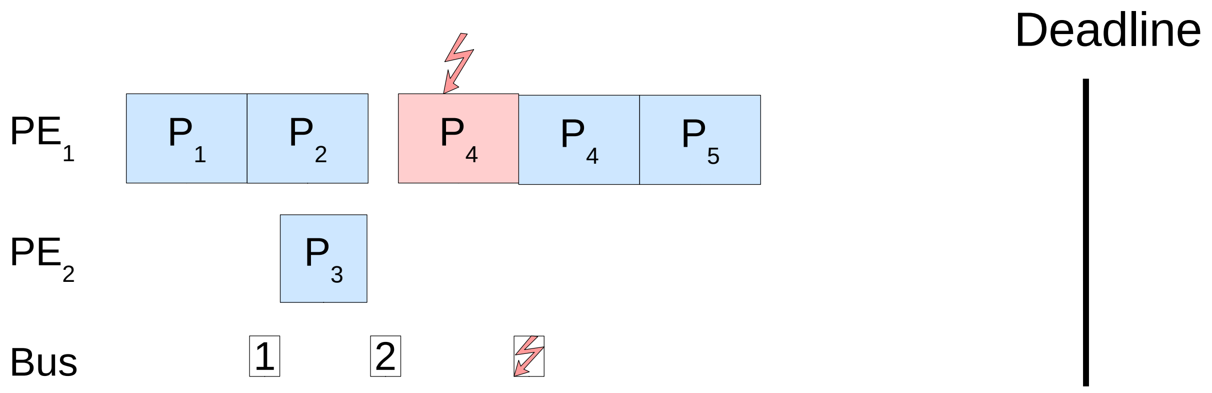
k=1

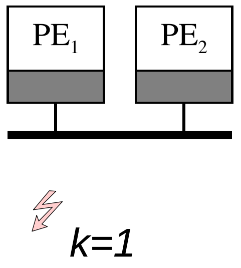
Fault-Tolerant Scheduling

- Even faster
  - Sacrifice all transparency
  - Schedule for each fault scenario
  - At most *k* re-executions



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

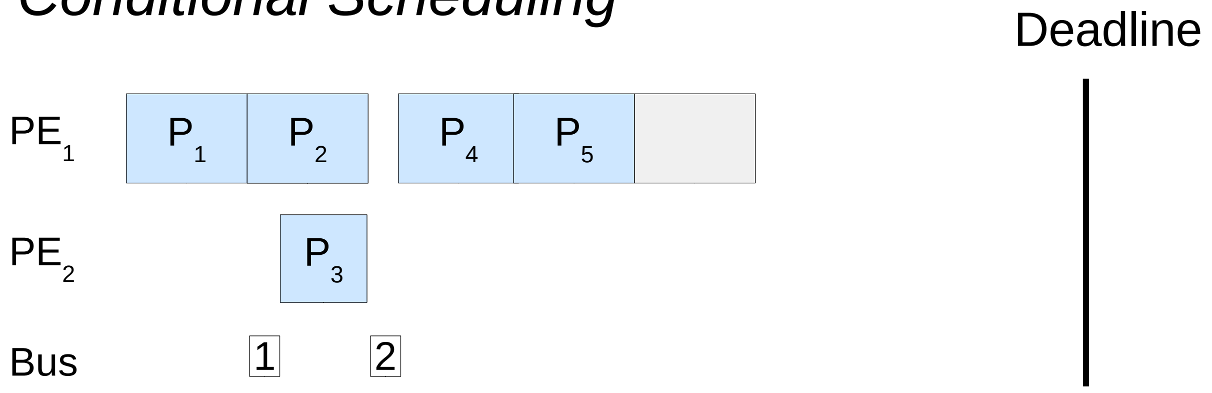


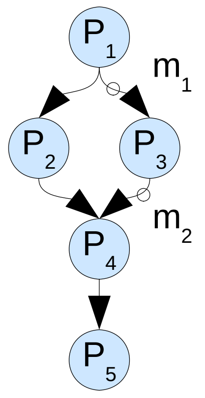


Fault-Tolerant Scheduling

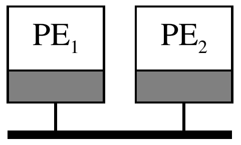
- Even faster
  - Sacrifice all transparency
  - Schedule for each fault scenario
  - At most *k* re-executions
  - All faults information is shared

Conditional Scheduling





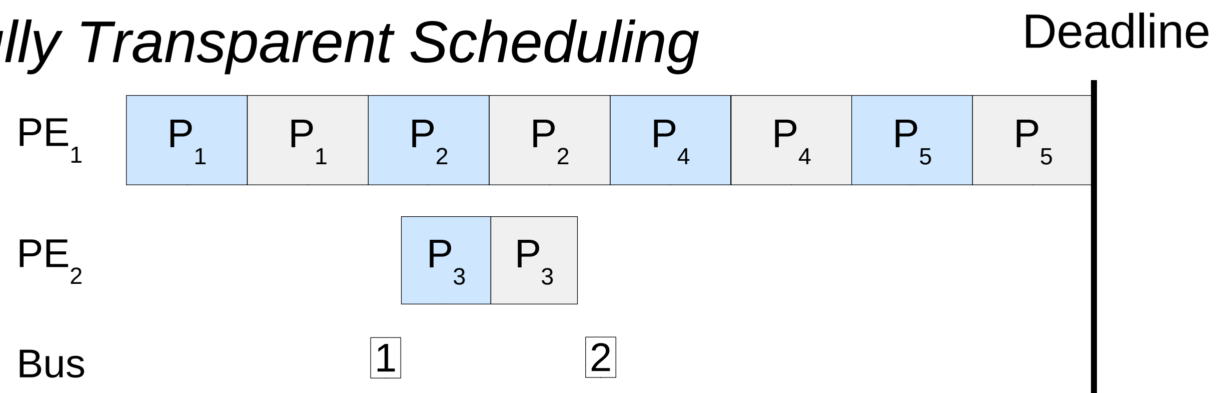
	PE1	PE2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4



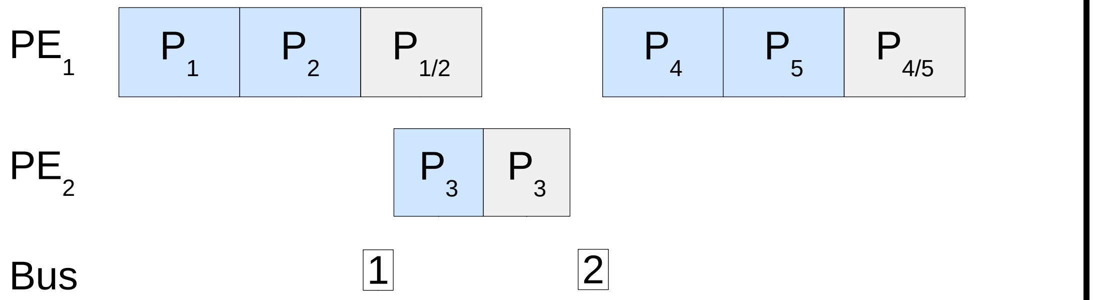
k=1

Fault-Tolerant Schedulings

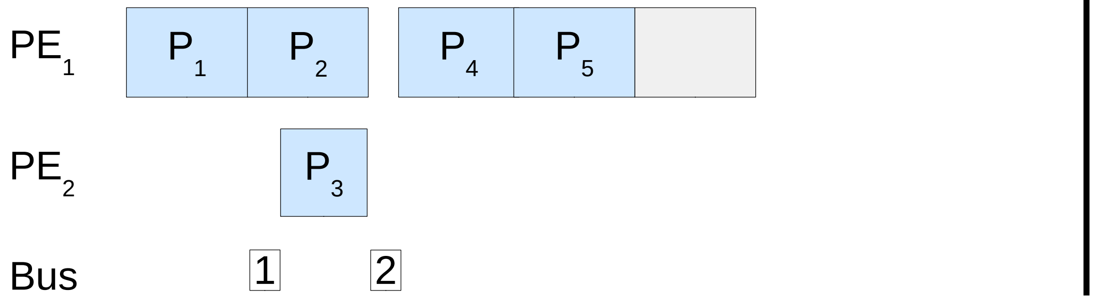
Fully Transparent Scheduling

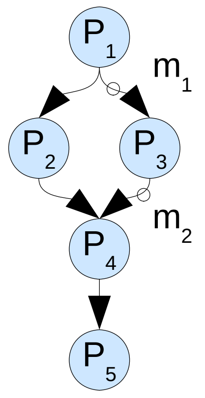


Slack Sharing Scheduling

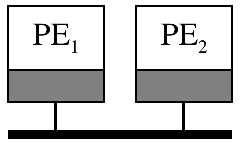


Conditional Scheduling





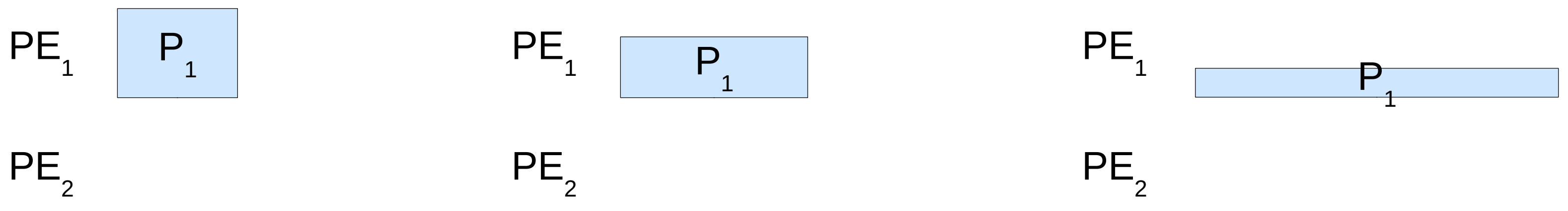
	PE$_1$	PE$_2$
P$_1$	4	4
P$_2$	4	4
P$_3$	3	3
P$_4$	4	4
P$_5$	4	4



k=1

Energy Management

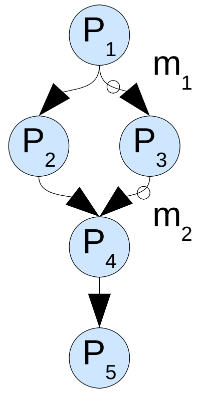
- Goal: minimise energy consumption
  - Dynamic voltage scaling



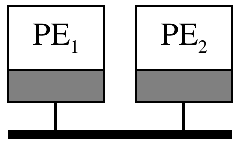
100% V_{ss} | 66% V_{ss} | 33% V_{ss}

100% E_0 | 44% E_0 | 11% E_0

Energy Management

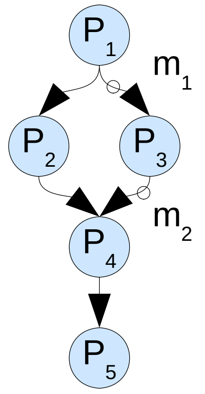


	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4



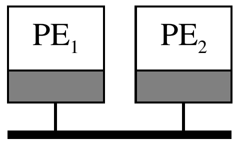
k=1

Energy Management



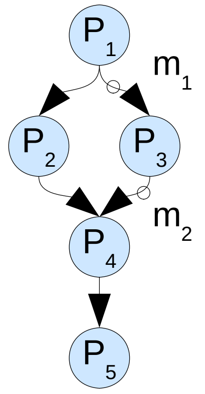
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

.



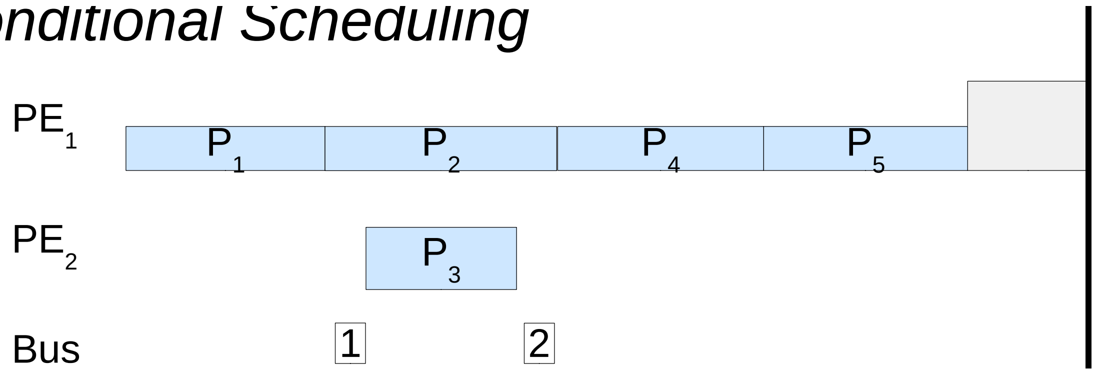
k=1

Energy Management

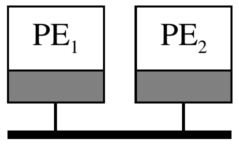


	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

Conditional Scheduling

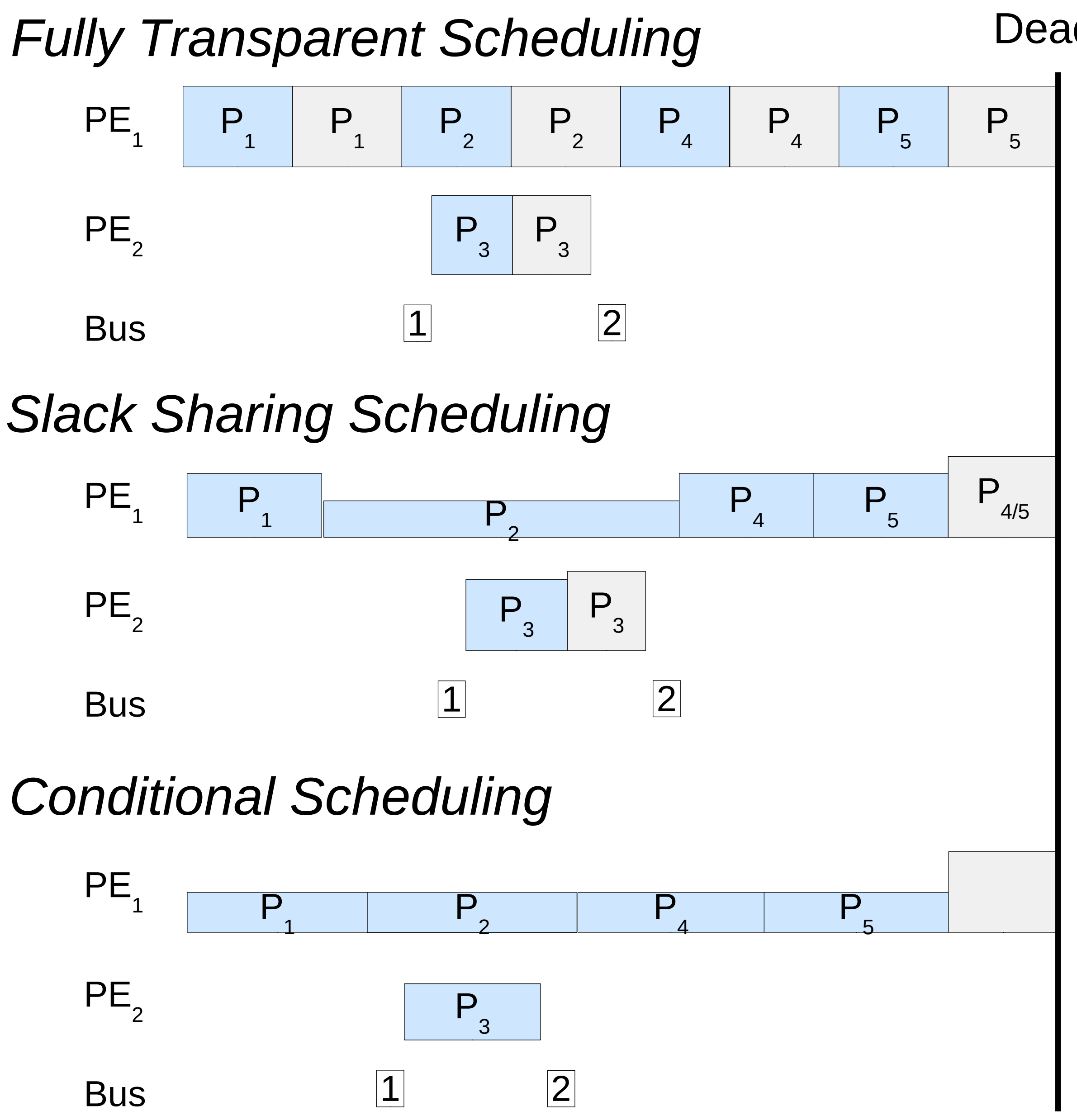


38% E_0



k=1

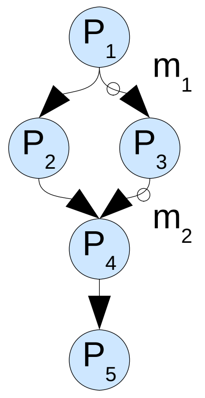
Energy Management



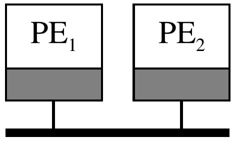
100% E_0

63% E_0

38% E_0



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4



k=1

Reliability Model

- Linear model (Fixed voltage)
  - Frequency is scaled
  - Linear relation between fault probability and frequency (due to longer execution time)
- Exponential model
  - Frequency and voltage is scaled equally
  - Lower voltages leads to smaller critical energy
  - Fault rate at minimum frequency

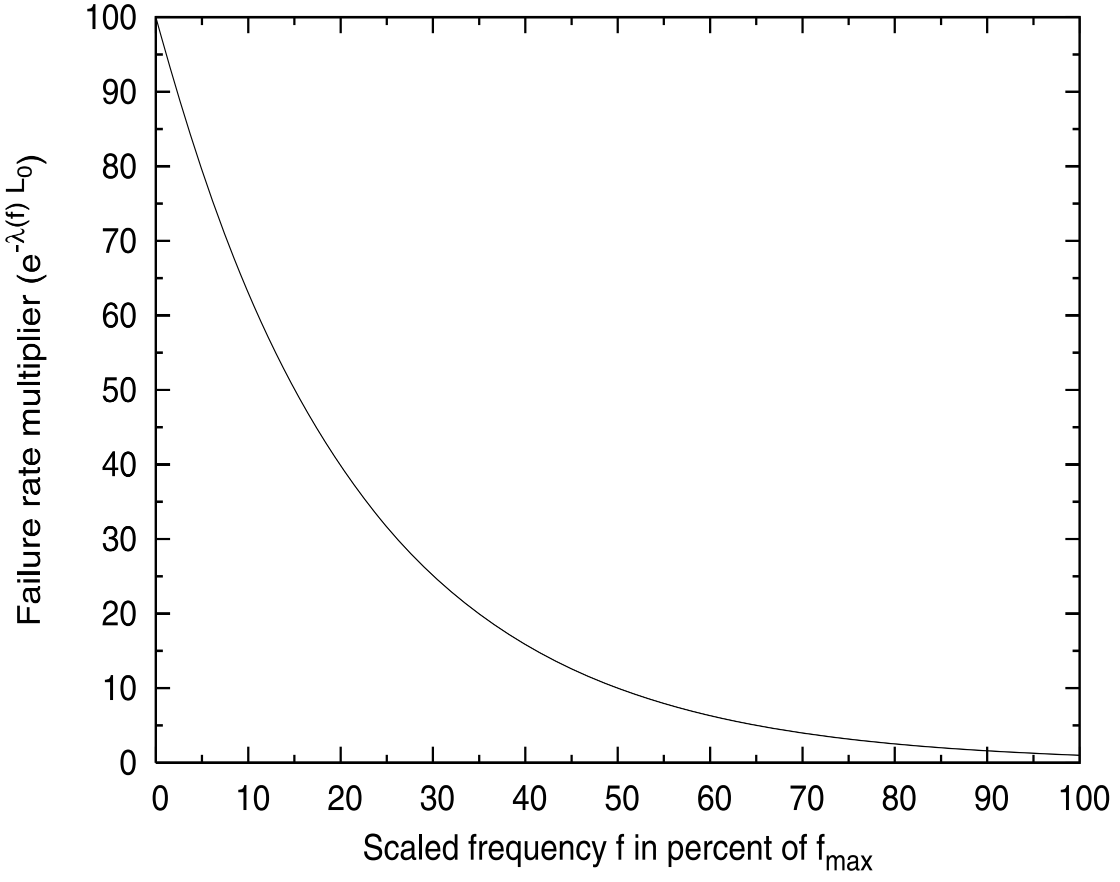
$$\lambda_0 \cdot 10^d, \quad d > 0$$

Dakai Zhu,the Effects of Energy Management on Reliability in RT Embedded Systems, 2004

Reliability and Energy

- Lowering voltage increases no. faults
  - Lower energy particles cause fault

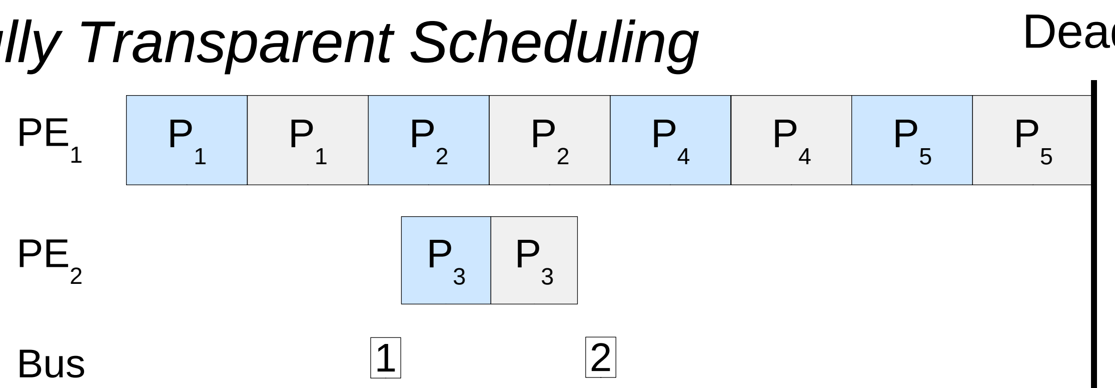
$$\lambda(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}$$



Dakai Zhu, Reliability-Aware Dynamic Energy Management in Dependable Embedded Real-Time Systems, 2006

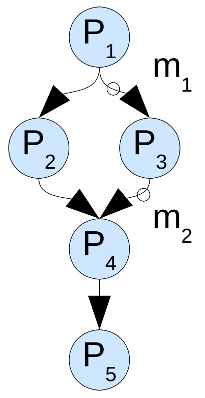
Energy Management

Fully Transparent Scheduling

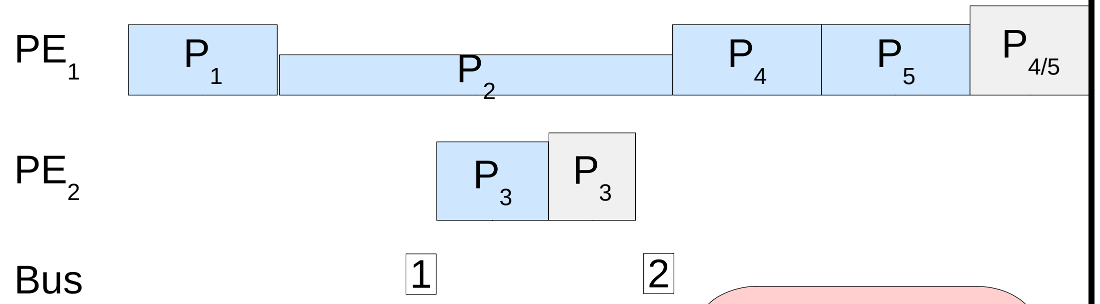


R=0.999 999 999 93

100% E_0



Slack Sharing Scheduling

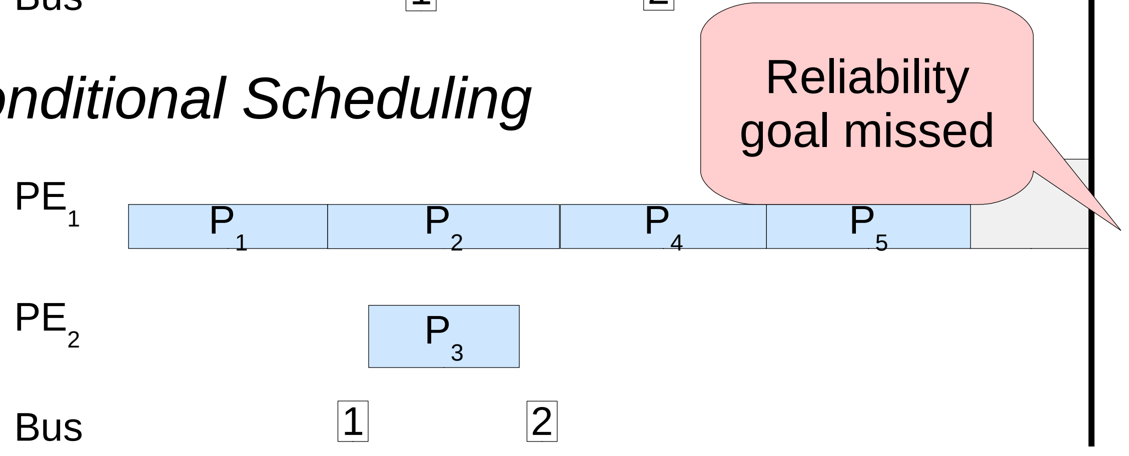


R=0.999 999 999 25

63% E_0

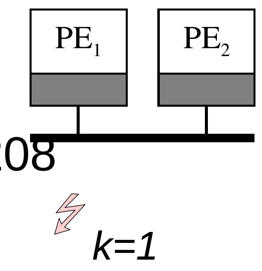
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

Conditional Scheduling



R=0.999 999 958 208

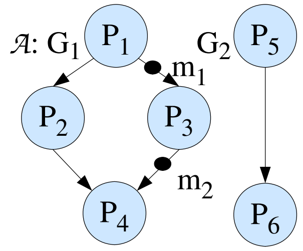
38% E_0



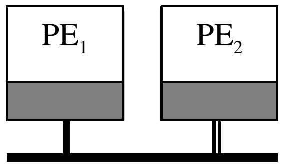
Outline

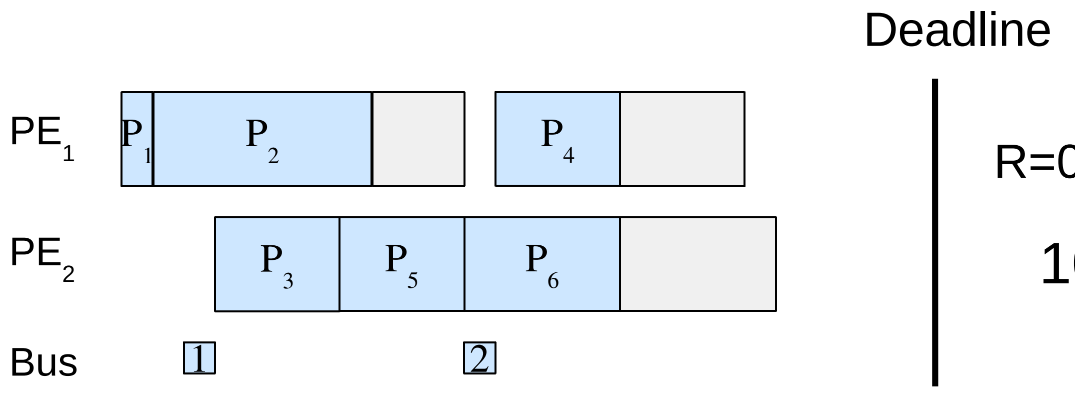
Energy/Reliability Trade-off

- Reliability goal: 0.999 999 9



	N_1	N_2
P_1	10	X
P_2	70	X
P_3	X	40
P_4	40	X
P_5	X	40
P_6	X	50





R=0.999 999 987

100% E_0

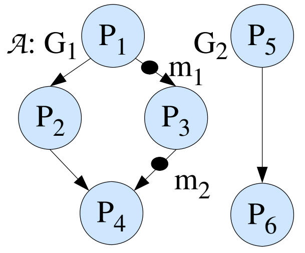
Voltage levels

N_1	100%	66%	33%
N_2	100%	66%	33%

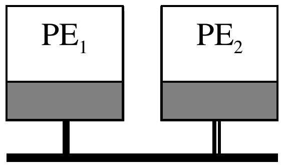
k = 1

Energy/Reliability Trade-off

- Reliability goal: 0.999 999 9
- Set reliability as hard constraint



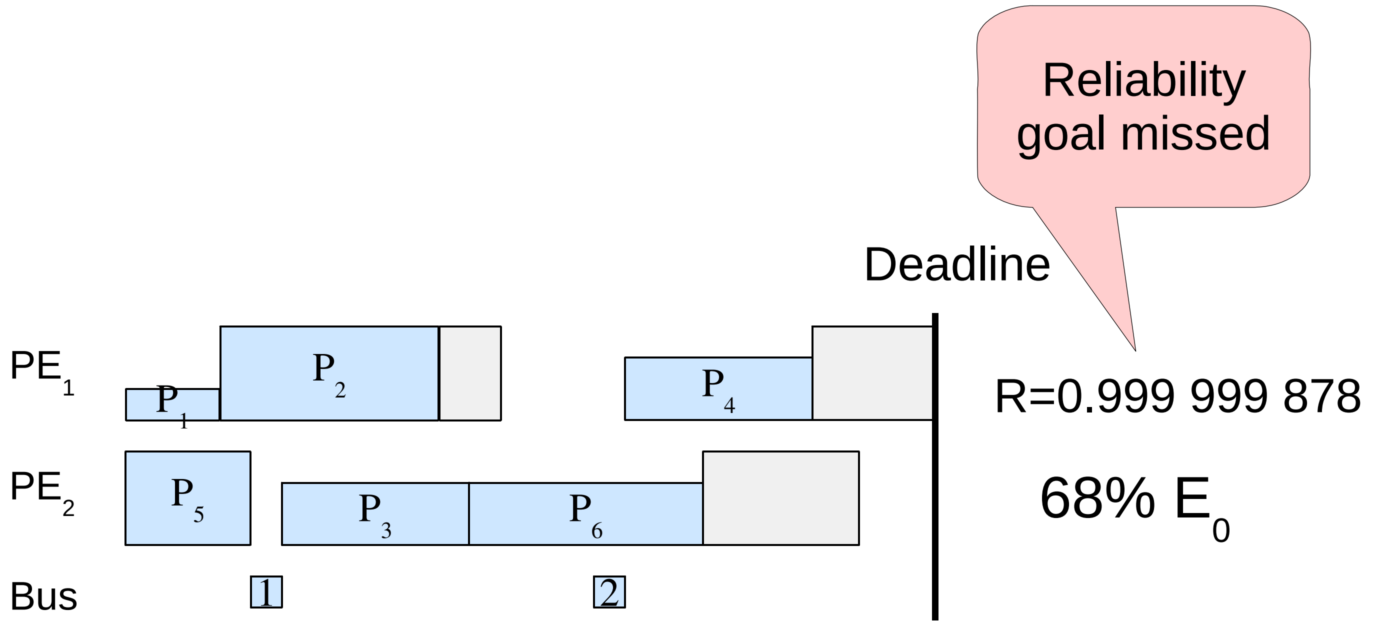
	N_1	N_2
P_1	10	X
P_2	70	X
P_3	X	40
P_4	40	X
P_5	X	40
P_6	X	50



Voltage levels

N_1	100%	66%	33%
N_2	100%	66%	33%

k = 1

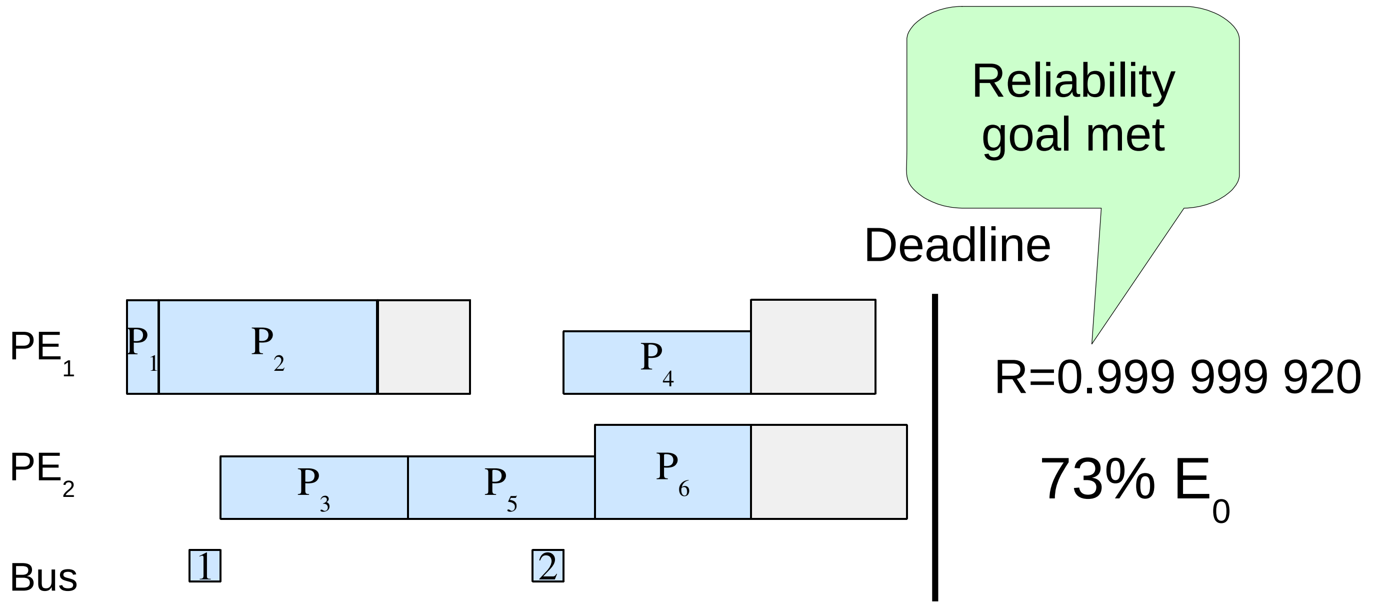


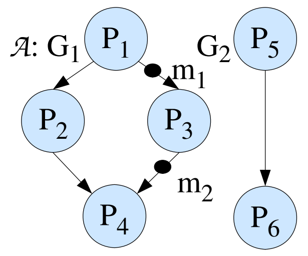
R=0.999 999 878

68% E_0

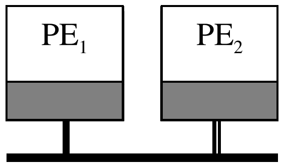
Energy/Reliability Trade-off

- Reliability goal: 0.999 999 9
- Set reliability as hard constraint
- Trade-off 5% energy
- Meets reliability goal





	N_1	N_2
P_1	10	X
P_2	70	X
P_3	X	40
P_4	40	X
P_5	X	40
P_6	X	50



Voltage levels

N_1	100%	66%	33%
N_2	100%	66%	33%

k = 1

Outline

- Introduction
- Motivation
- *Problem Formulation*
- Implementation
- Experiments and Results
- Conclusions
- Q&A

Problem Formulation

- Input
  - Application
  - Architecture
  - Reliability goal
- Decide
  - Fault-Tolerant Scheduling
  - Mapping
  - Fault-Tolerance Policy
- While optimising for
  - Energy
  - Under hard reliability goal

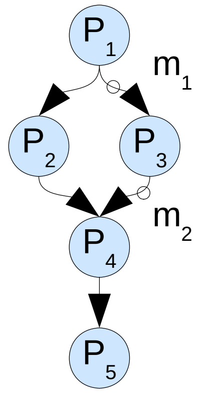
Implementation

- Problem is NP-Complete
  - Normally solved using “best effort” heuristics
- Use constraint logic programming
  - Good performance with NP-completeness
  - Optimal solutions are feasible
  - Flexible model
  - ECLiPSe-CLP

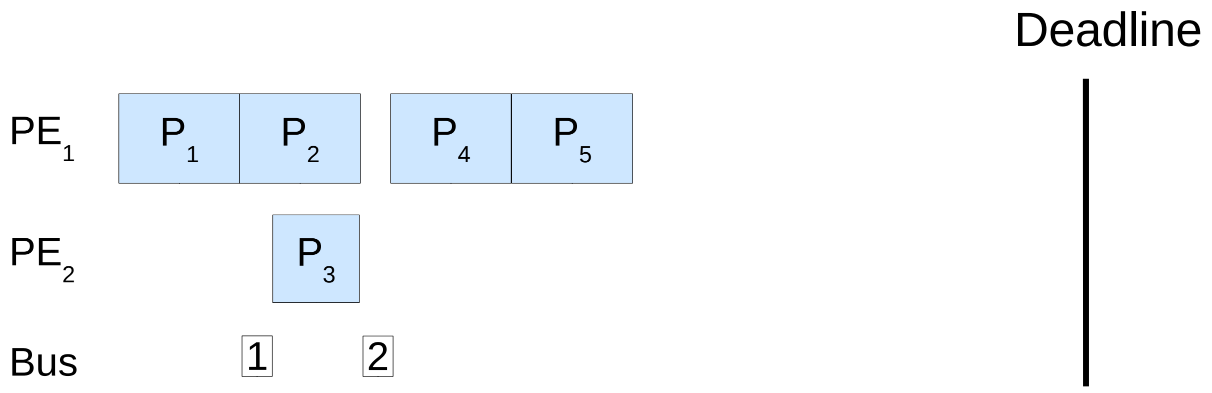
Constraints for Embedded Systems

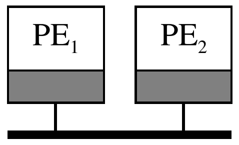
- Precedence constraint

$$Start(P_j) \geq \forall_{e_{ij}} Start(P_i) + Duration(P_i)$$



	PE1	PE2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

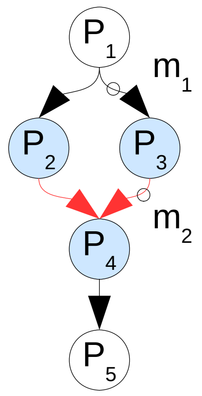




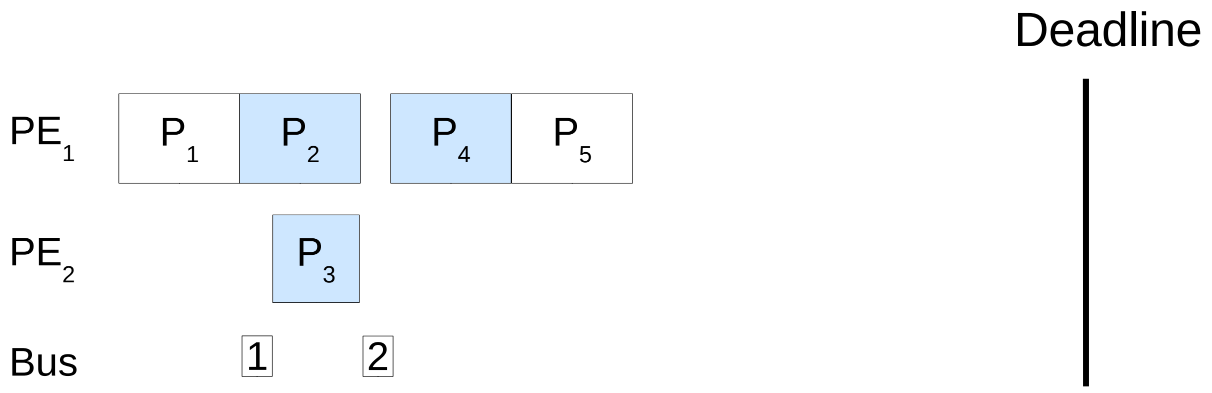
Constraints for Embedded Systems

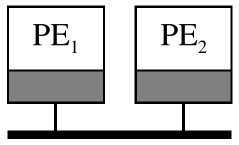
- Precedence constraint

$$Start(P_j) \geq \forall_{e_{ij}} Start(P_i) + Duration(P_i)$$



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

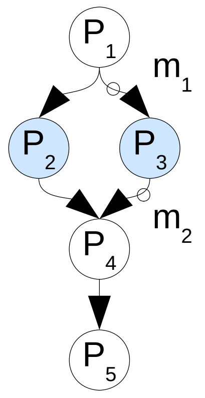




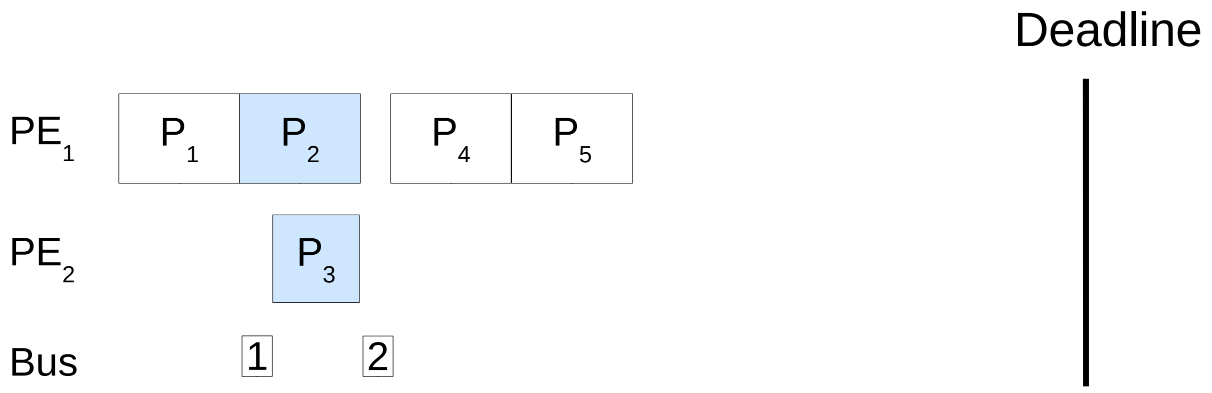
Constraints for Embedded Systems

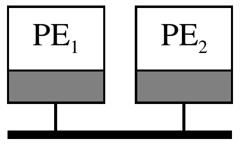
- Precedence constraint
- Resource constraint

$$Mapping(P_i) \neq Mapping(P_j)$$
$$v\ Start(P_i) \geq Start(P_j) + Duration(P_j)$$
$$v\ Start(P_j) \geq Start(P_i) + Duration(P_i)$$



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

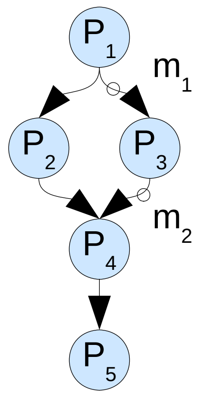




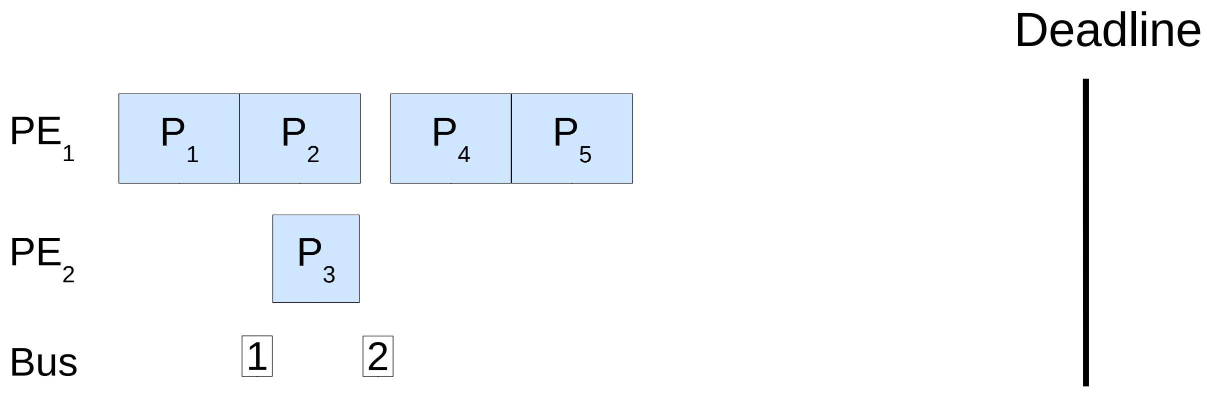
Constraints for Embedded Systems

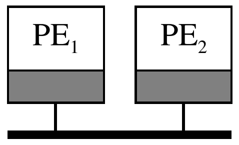
- Precedence constraint
- Resource constraint
- Timing constraint

$$Start(P_i) + Duration(P_i) \leq Deadline$$



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

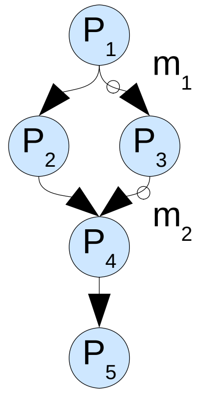




Fault-Tolerance Constraints

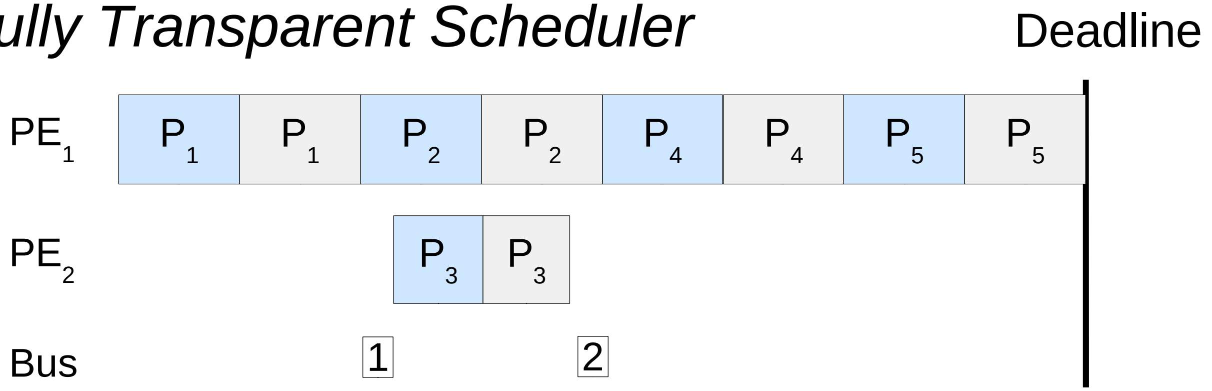
- Changed precedence constraint

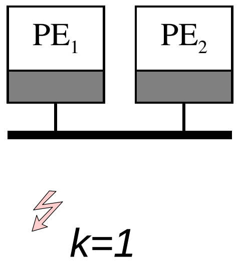
$$Start(P_j) \geq \forall_{e_{ij}} Start(P_i) + Duration(P_i)(k+1)$$



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

Fully Transparent Scheduler

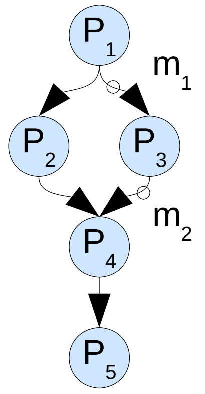




Fault-Tolerance Constraints

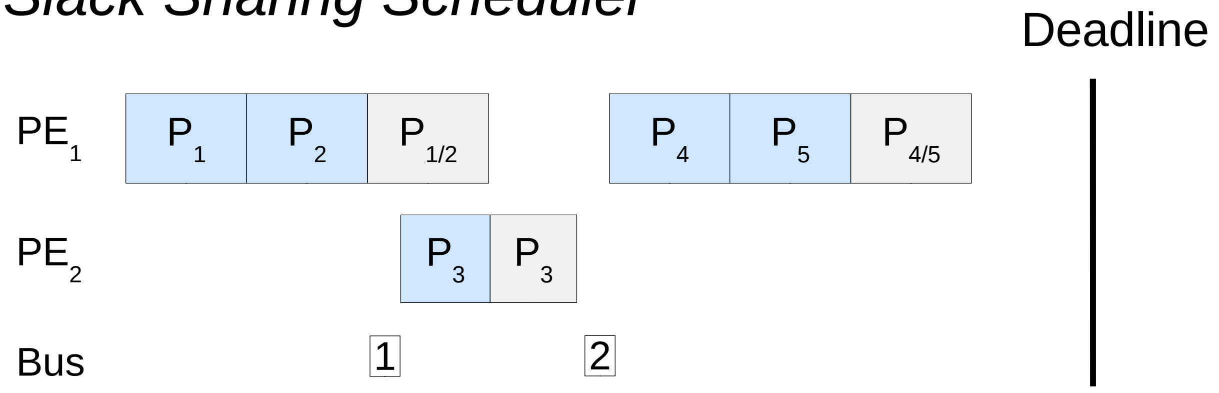
- More complex to model
  - Create separate schedule for recoveries

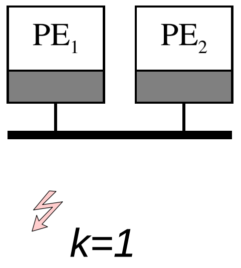
$$Mapping(P_i) = Mapping(P_j)$$
$$\vee\ Start(P_j) \geq Endtime(S_i)$$



	PE1	PE2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

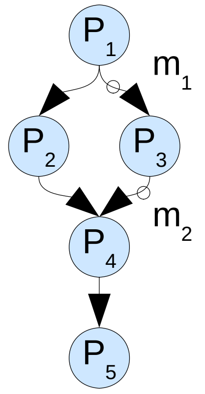
Slack Sharing Scheduler





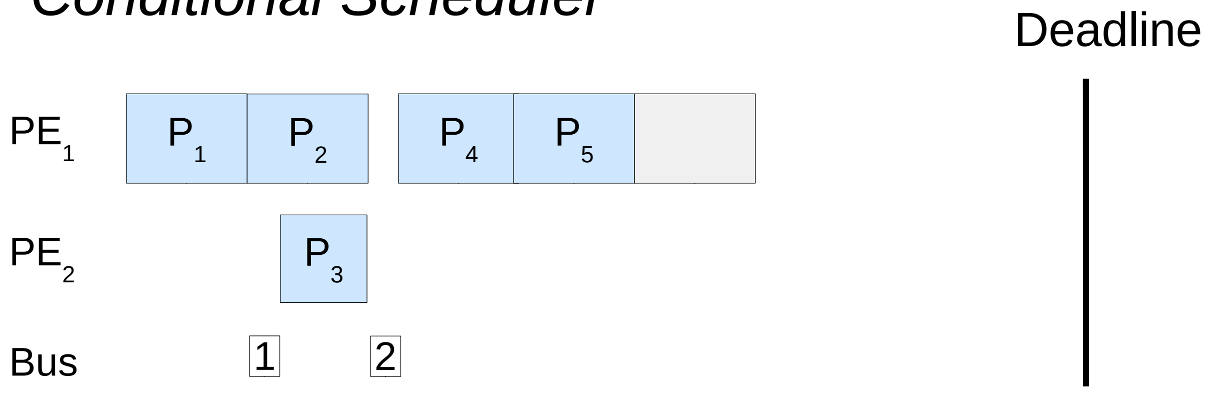
Fault-Tolerance Constraints

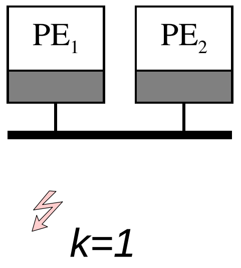
- Schedule for all fault scenarios
  - These are captured by an FT-CPG



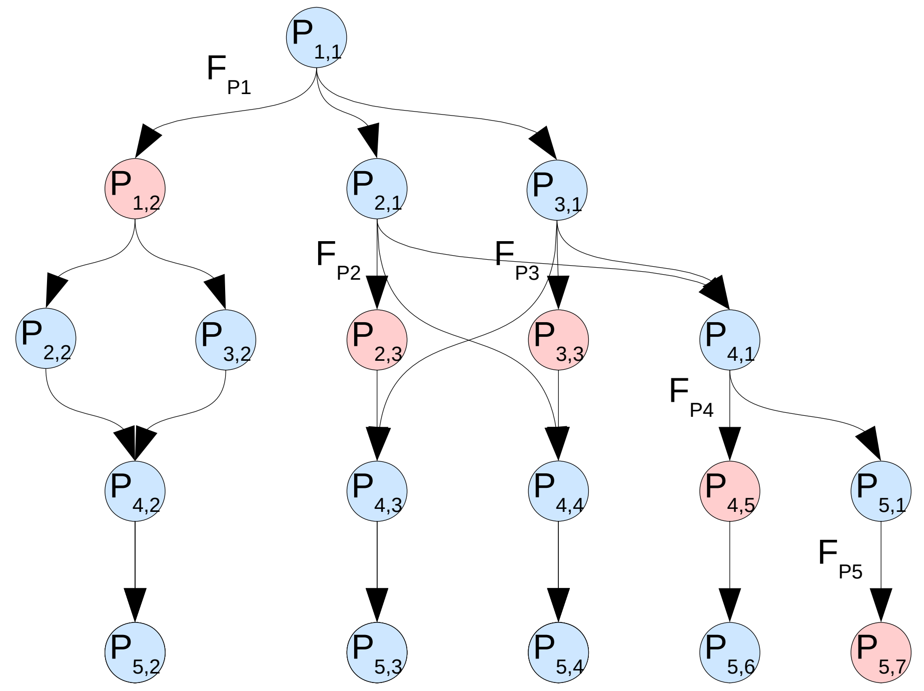
	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

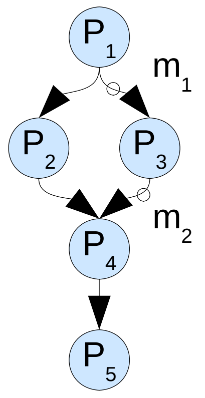
Conditional Scheduler





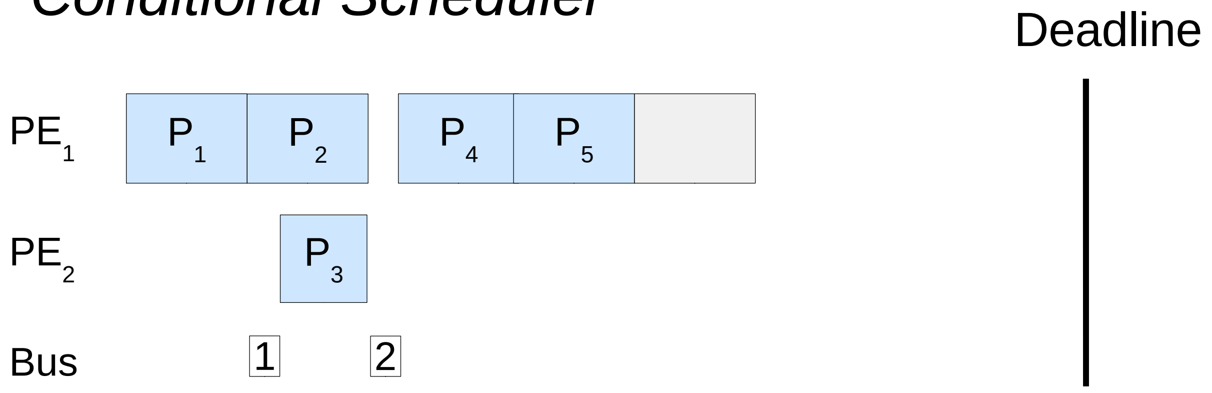
Fault-Tolerance Constraints

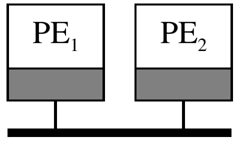




	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

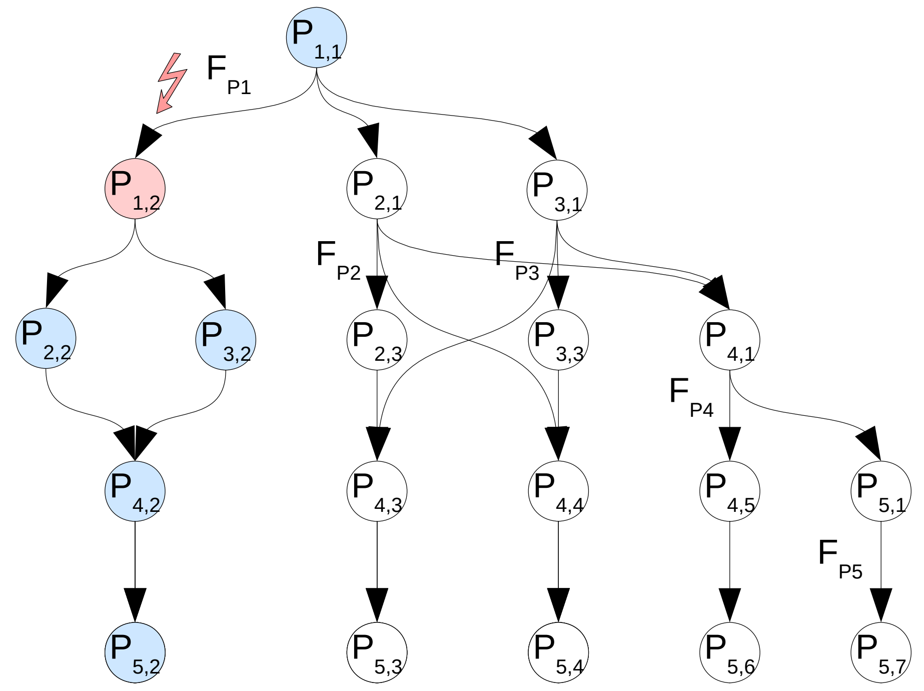
Conditional Scheduler

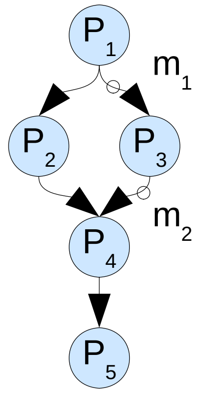




k=1

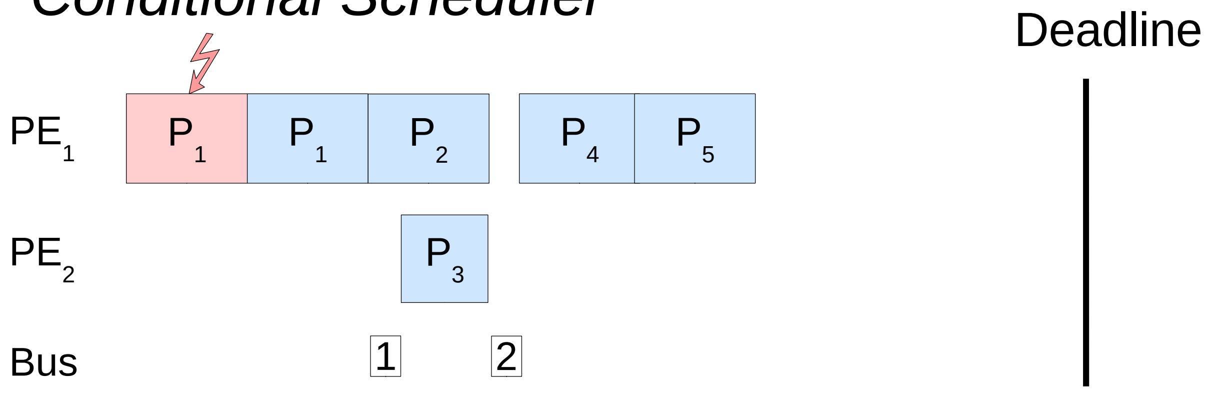
Fault-Tolerance Constraints

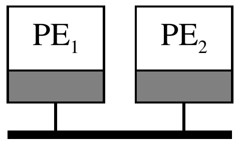




	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

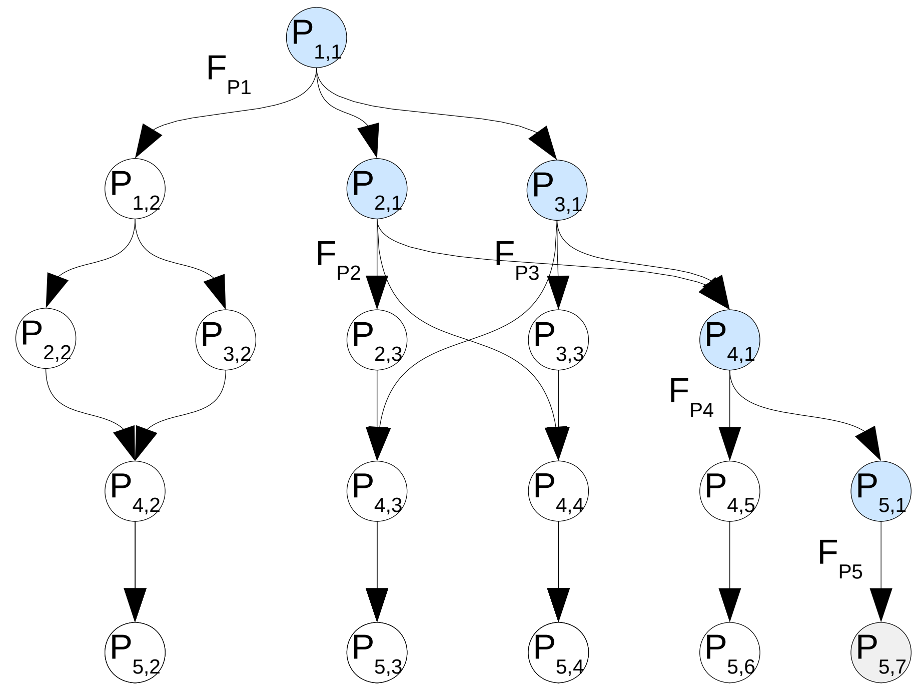
Conditional Scheduler

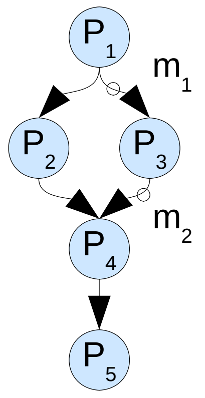




k=1

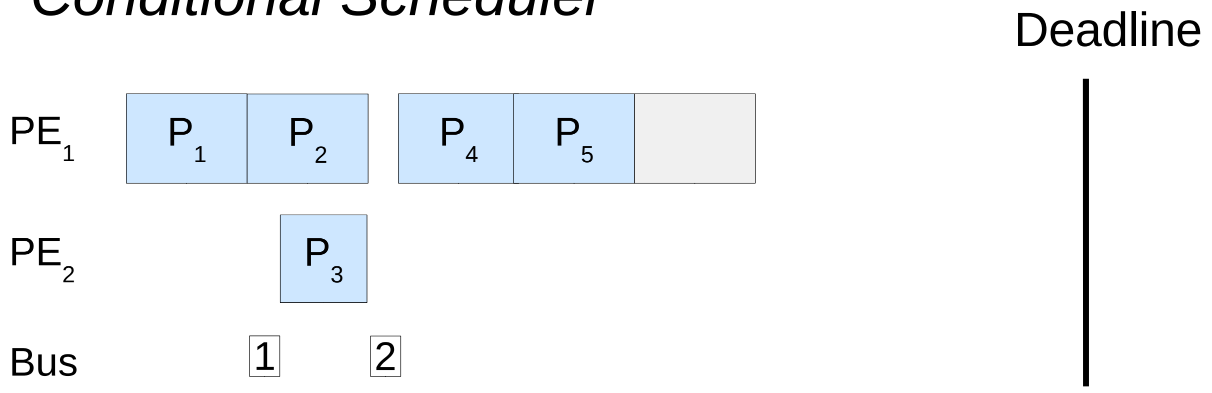
Fault-Tolerance Constraints

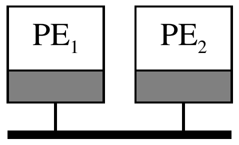




	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

Conditional Scheduler



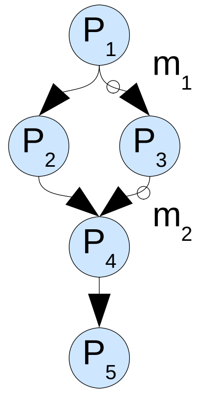


k=1

Fault-Tolerance Constraints

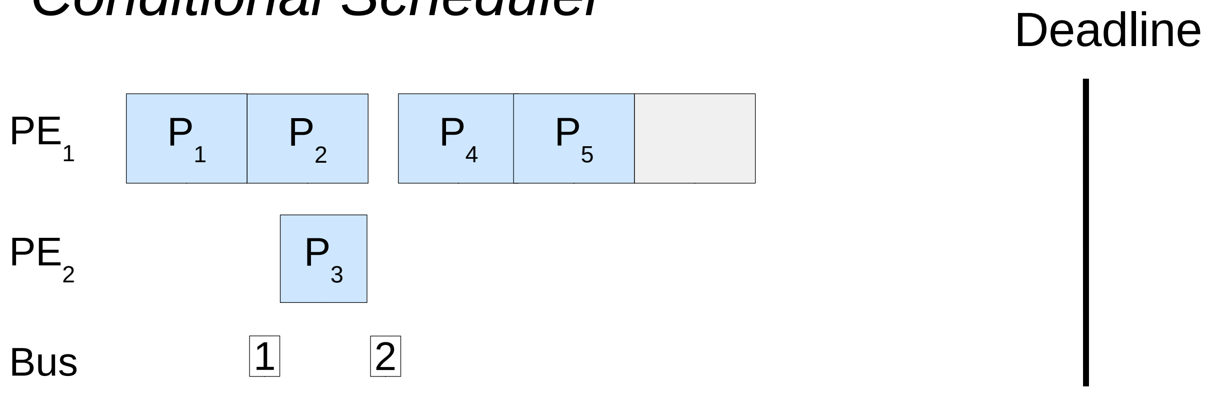
- Schedule for all fault scenarios
  - These are captured by an FT-CPG

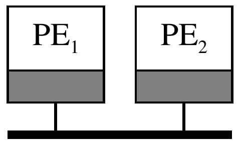
$$\mathit{MutuallyExclusive}(P_i, P_j)$$
$$\vee\ \mathit{ResourceConstraint}$$



	PE_1	PE_2
P_1	4	4
P_2	4	4
P_3	3	3
P_4	4	4
P_5	4	4

Conditional Scheduler



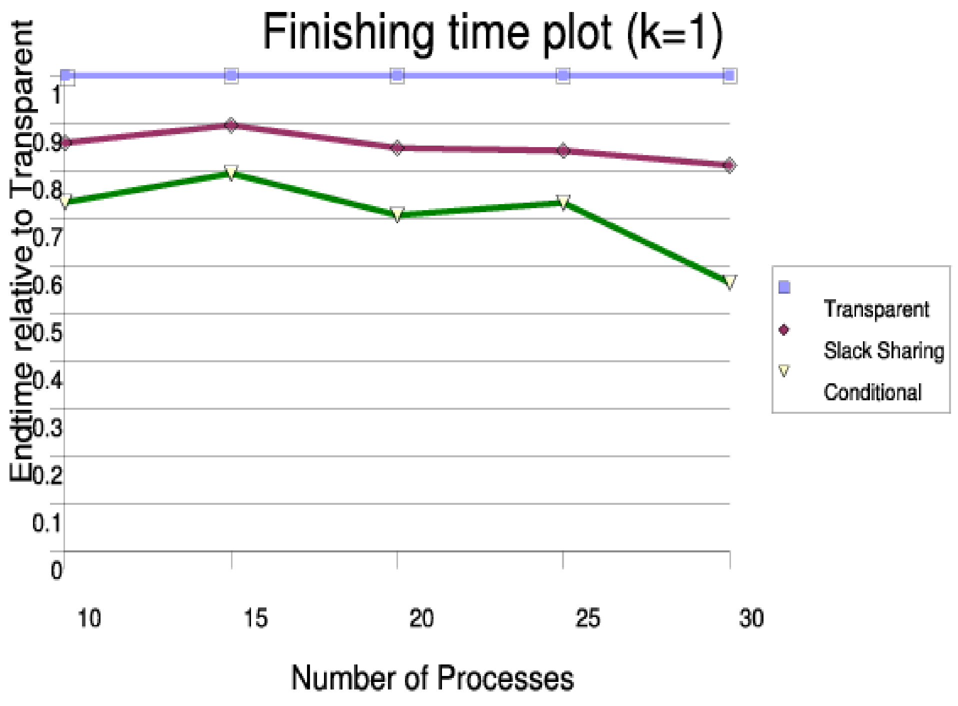


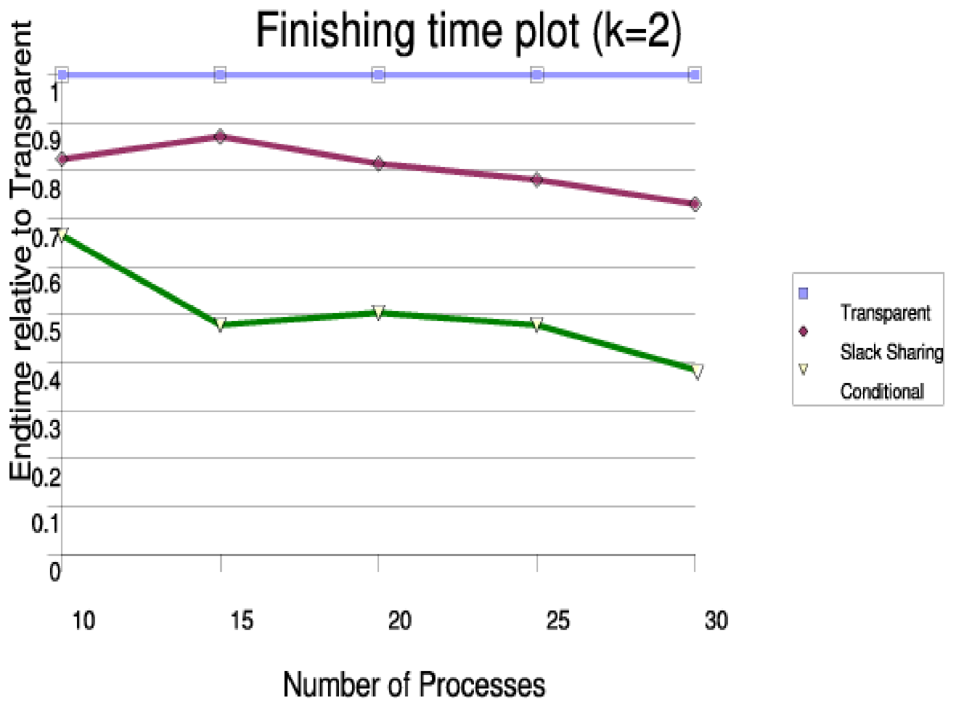
k=1

Outline

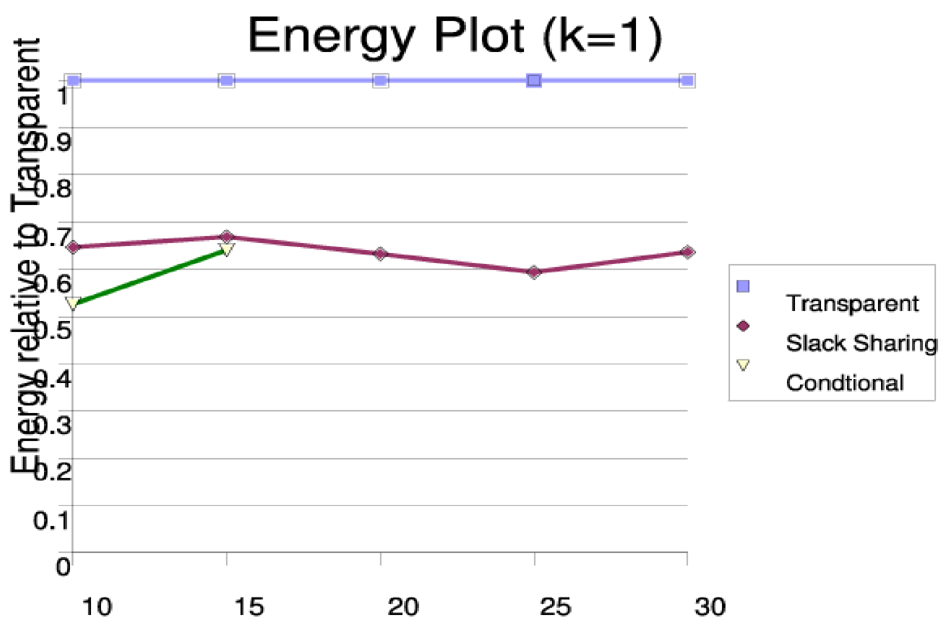
- Introduction
- Motivation
- Problem Formulation
- Implementation
- *Experiments and Results*
- Conclusions
- Q&A

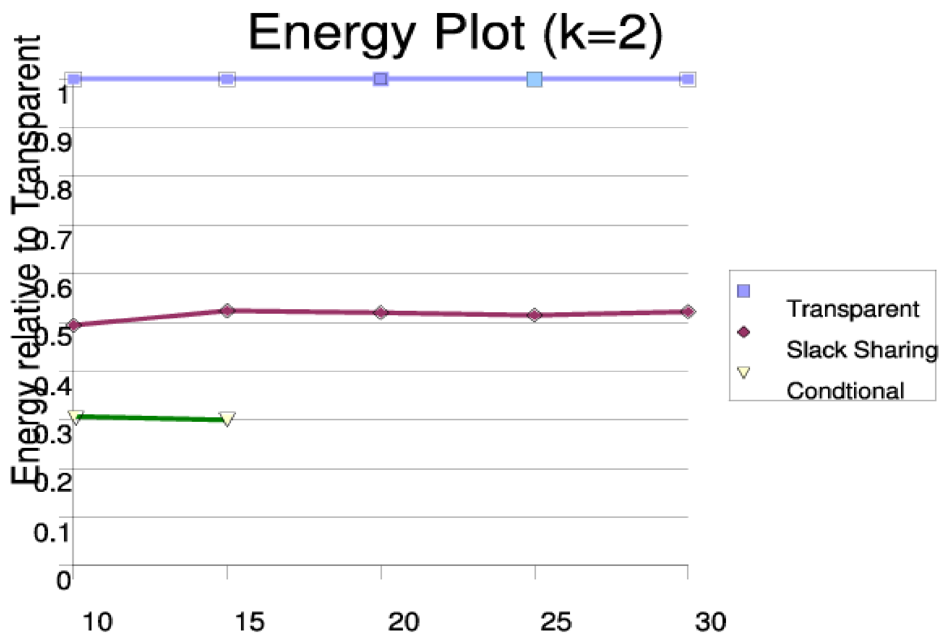
Comparison of Schedulers

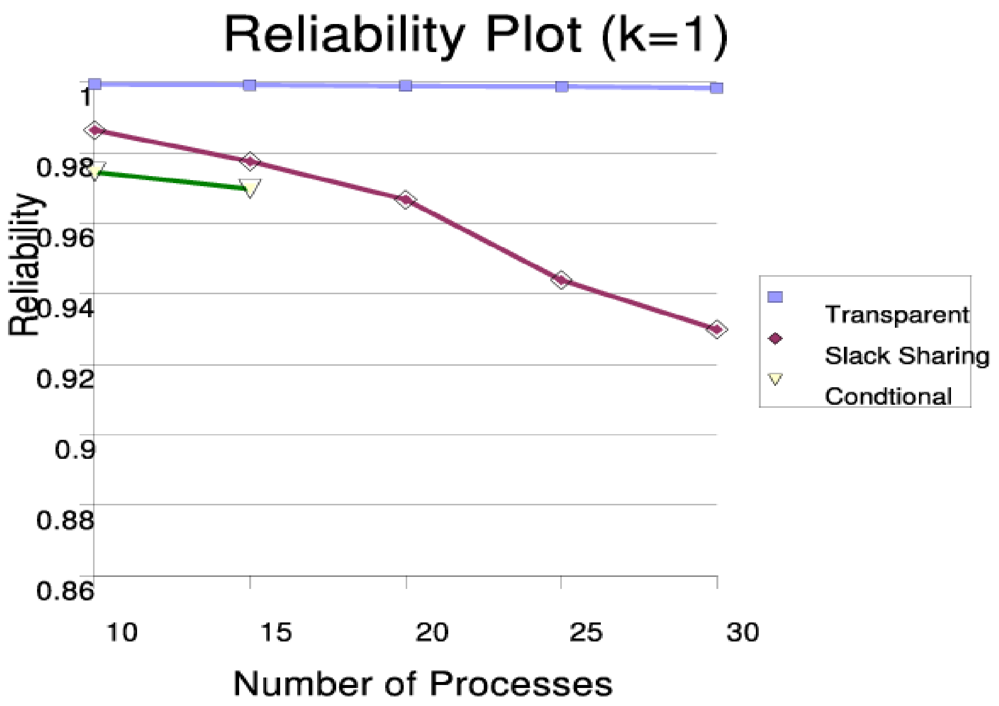


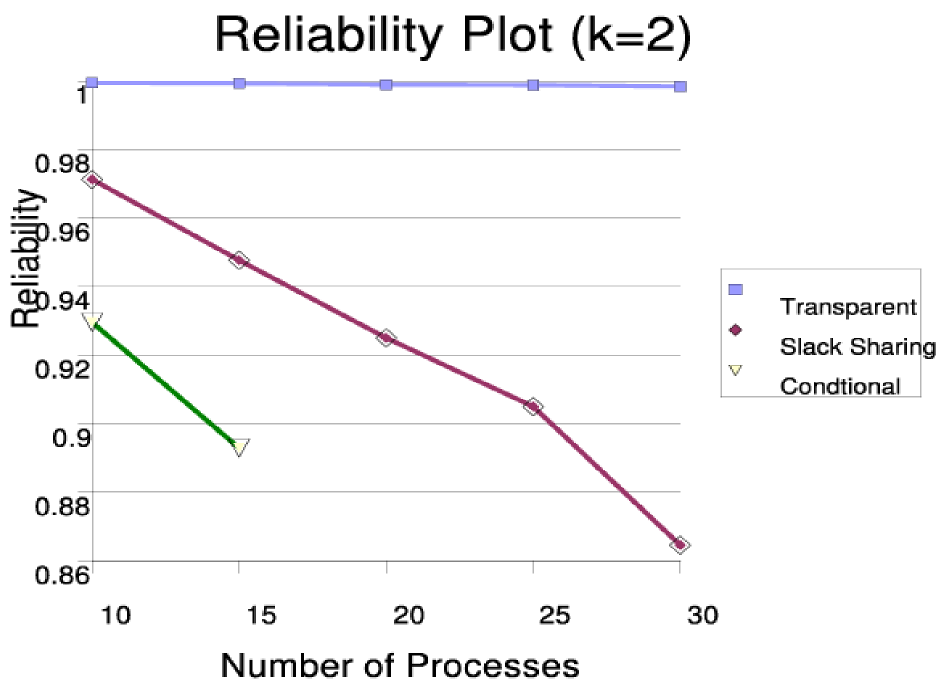


Comparison of Schedulers

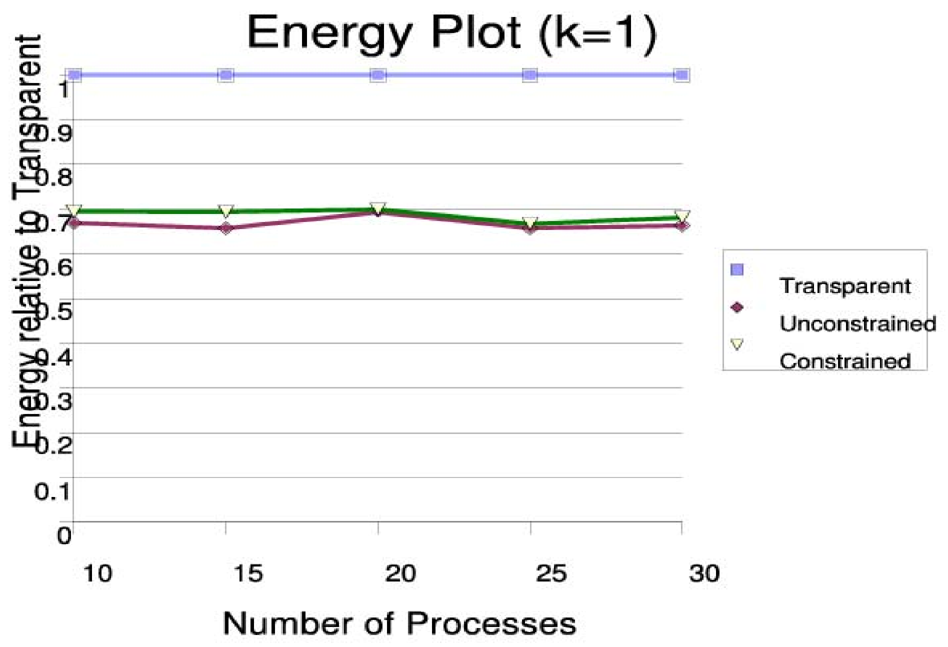


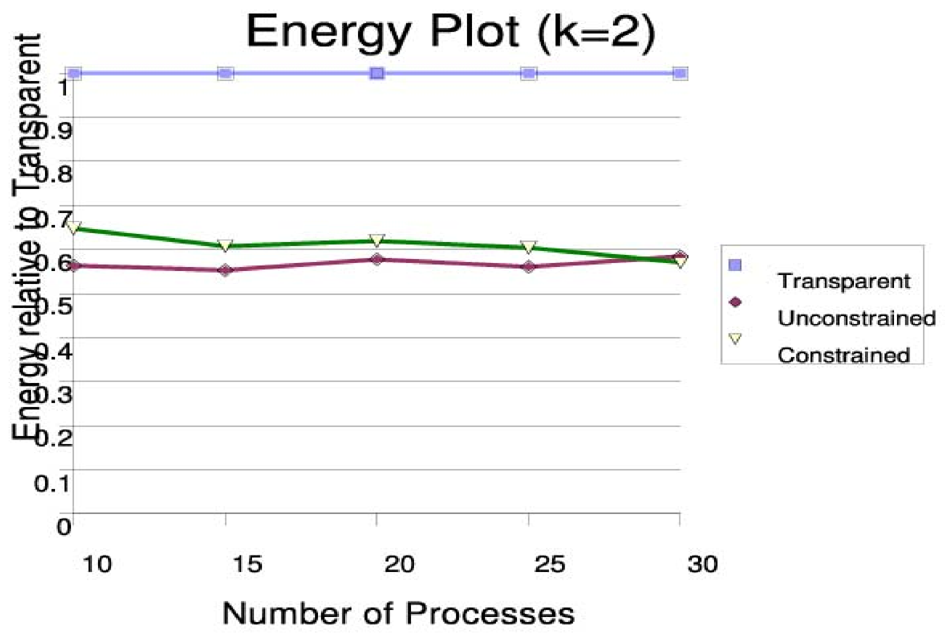


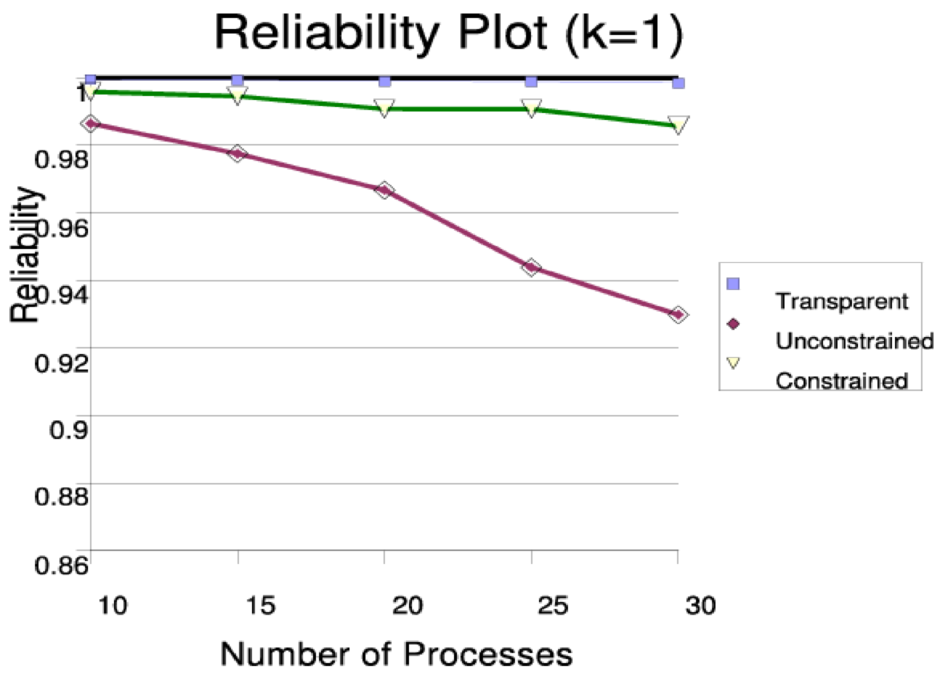


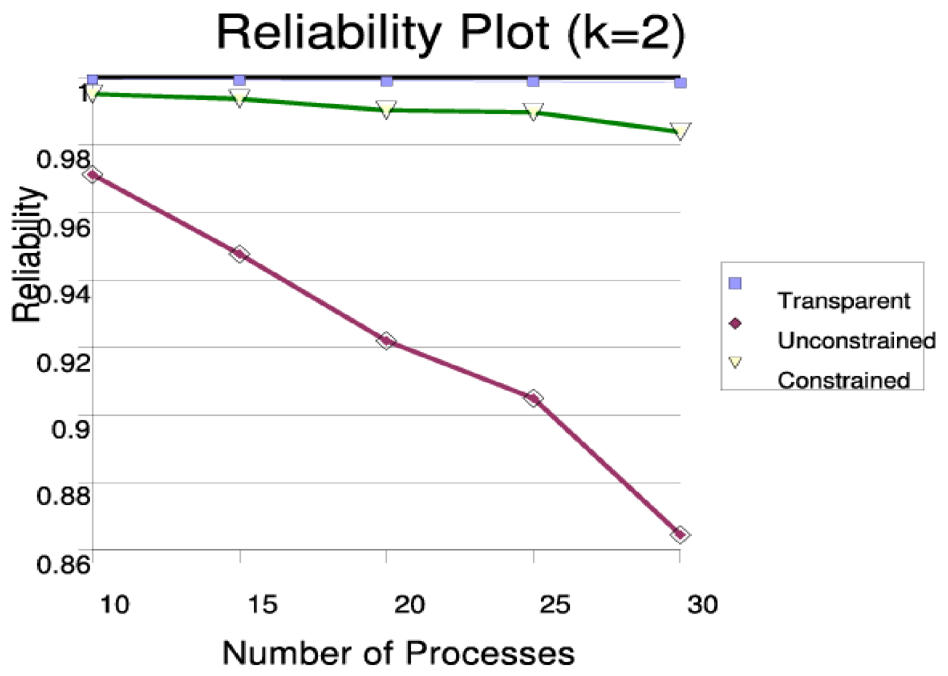


Reliability and Energy Trade-offs









Outline

Conclusions

- Design tool for doing
  - Fault tolerant scheduling
  - Mapping
  - Policy assignment
- Optimising for
  - Minimal energy
  - Hard constraints for timing and reliability
- Message:
  - Reliability can be met at little energy cost

Contributions

- Design optimisation for energy minimisation under reliability and timing constraints
  - *“Design Optimistaion of Low-Power Reliable Real-Time Embedded Systems”* RTSS (in preparation)
- Optimisation method that decides the voltage scaling
  - *“Scheduling and Voltage Scaling for Energy/Reliability Trade-offs in Fault-Tolerant Time-Triggered Embedded Systems”* CODES+ISSS (submitted)
- Efficient constraint logic programming-based scheduling technique
  - *“A Constraint Logic Programmnig Framework for the Synthesis of Fault-Tolerant Schedules for Distributed Embedded Systems”* ETFA (submitted)

Present Work

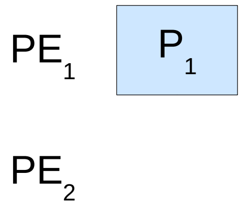
- Credit search based heuristic
- Addition of messages
  - TDMA FT scheduling
- Heterogeneous architectures
  - Both in terms of speed and reliability

Questions and Answers

Reliability

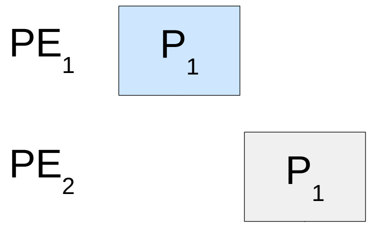
$$R_{single} = e^{-\lambda c} = 1 - \rho$$

Single execution:



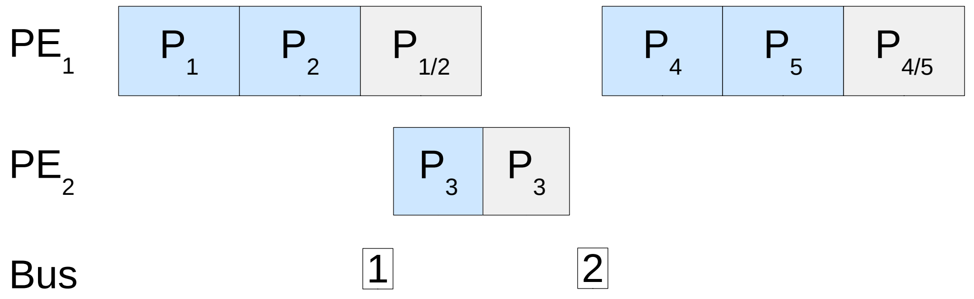
$$R_{FT} = 1 - \prod_{i=1}^{k} (1 - R_i)$$

Fault-tolerance:



$$R_{App} = \prod_{P_i \in A} R_{P_i}$$

Application



PE_1
P_1
P_2
$P_{1/2}$
P_4
P_5
$P_{4/5}$
PE_2
P_3
P_3
Bus
1
2