HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

Full virtualization of real-time systems by temporal partitioning

Authors

Timo Kerstan (morpheus@upb.de)

Daniel Baldin (dbaldin@upb.de)

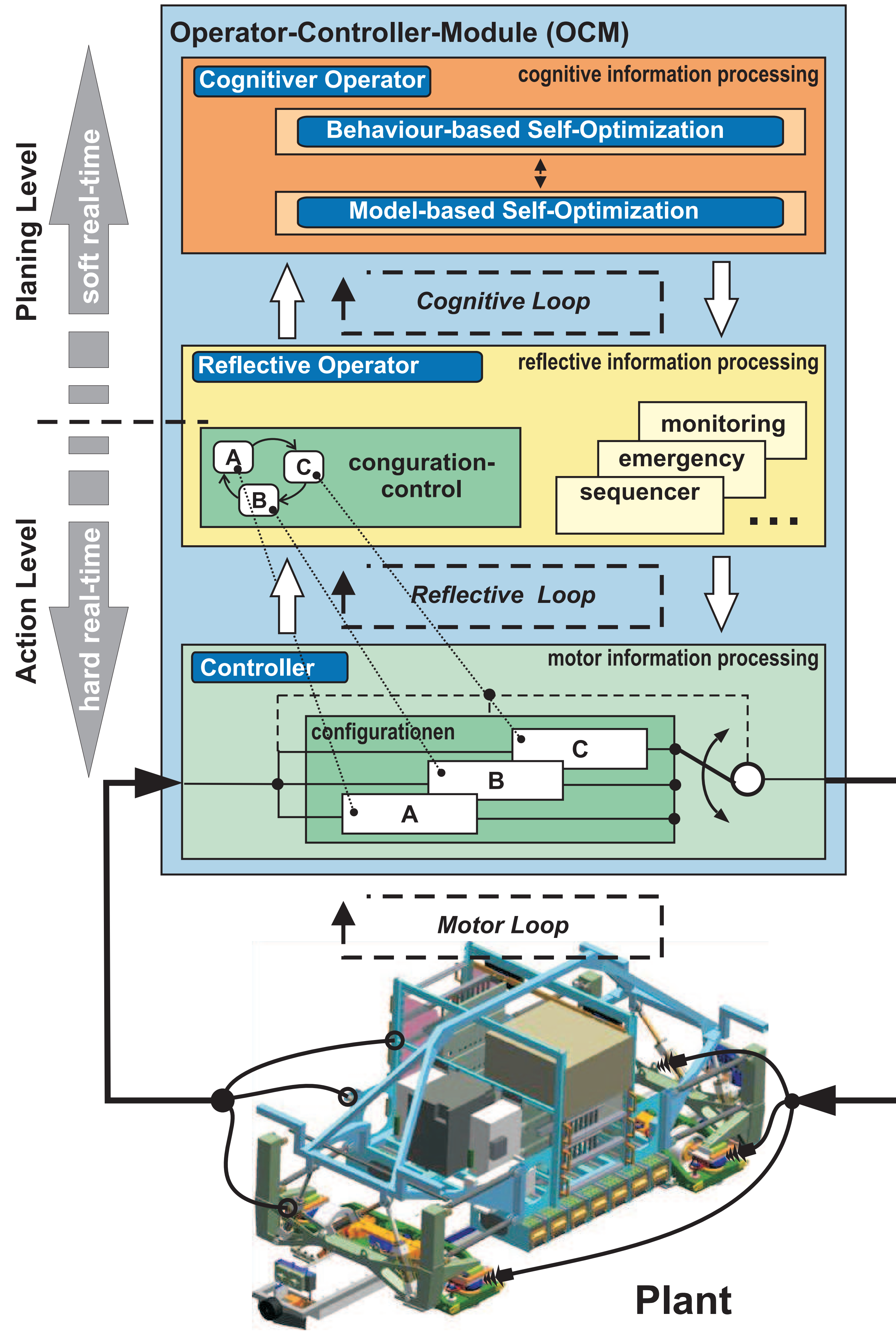
Stefan Grösbrink (morenga@upb.de)

Heinz Nixdorf Institute
University of Paderborn, Germany

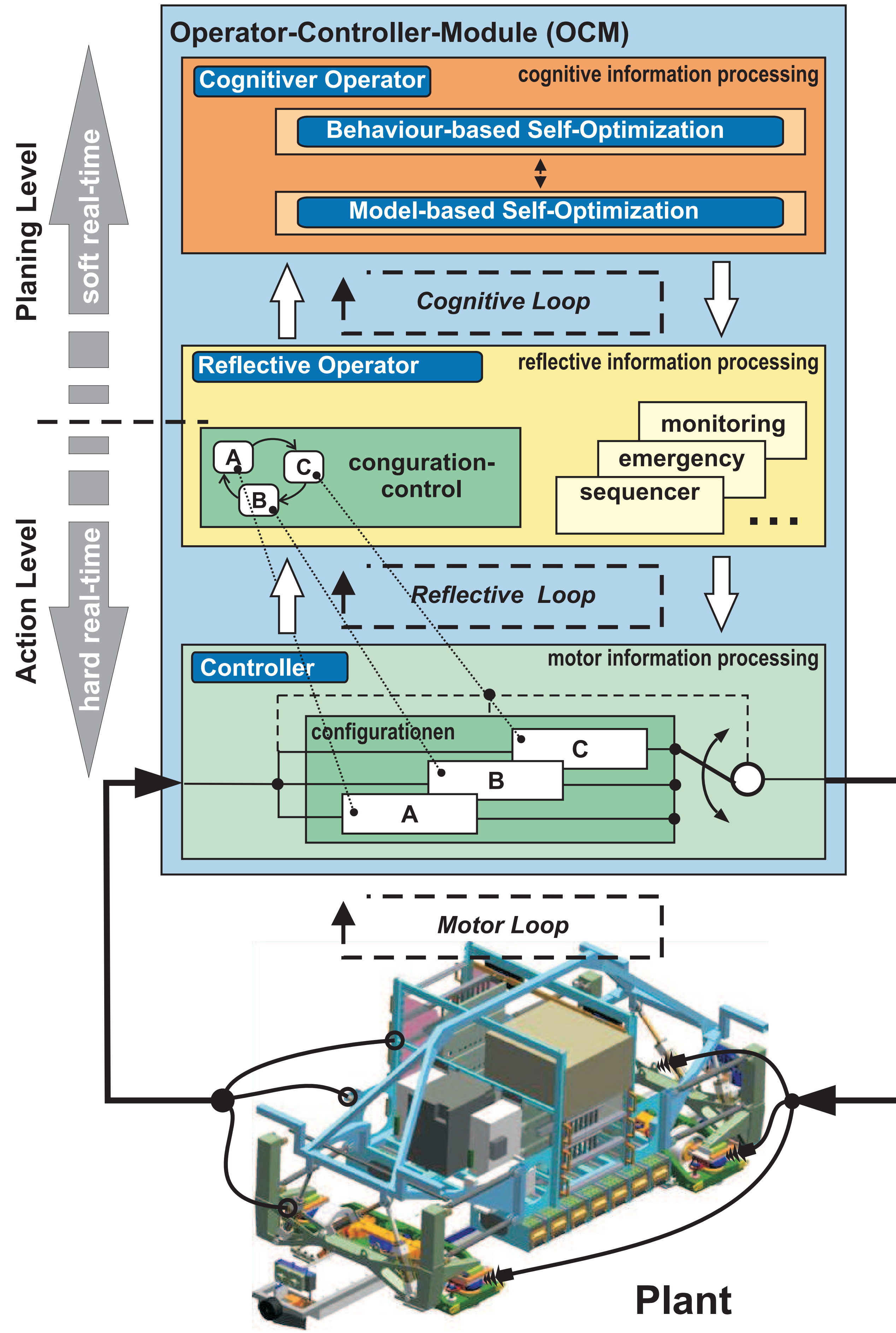
- **CRC614 performs research of self optimizing mechatronical systems**

- **OCM is the core module of the CRC 614**
  - Cognitive operator
  - Reflective operator
  - Controller

- **Properties:**
  - Hard & Soft realtime
  - High dynamics in
    - resource requirements
    - active services
    - operation modes
  - Fail safe behaviour

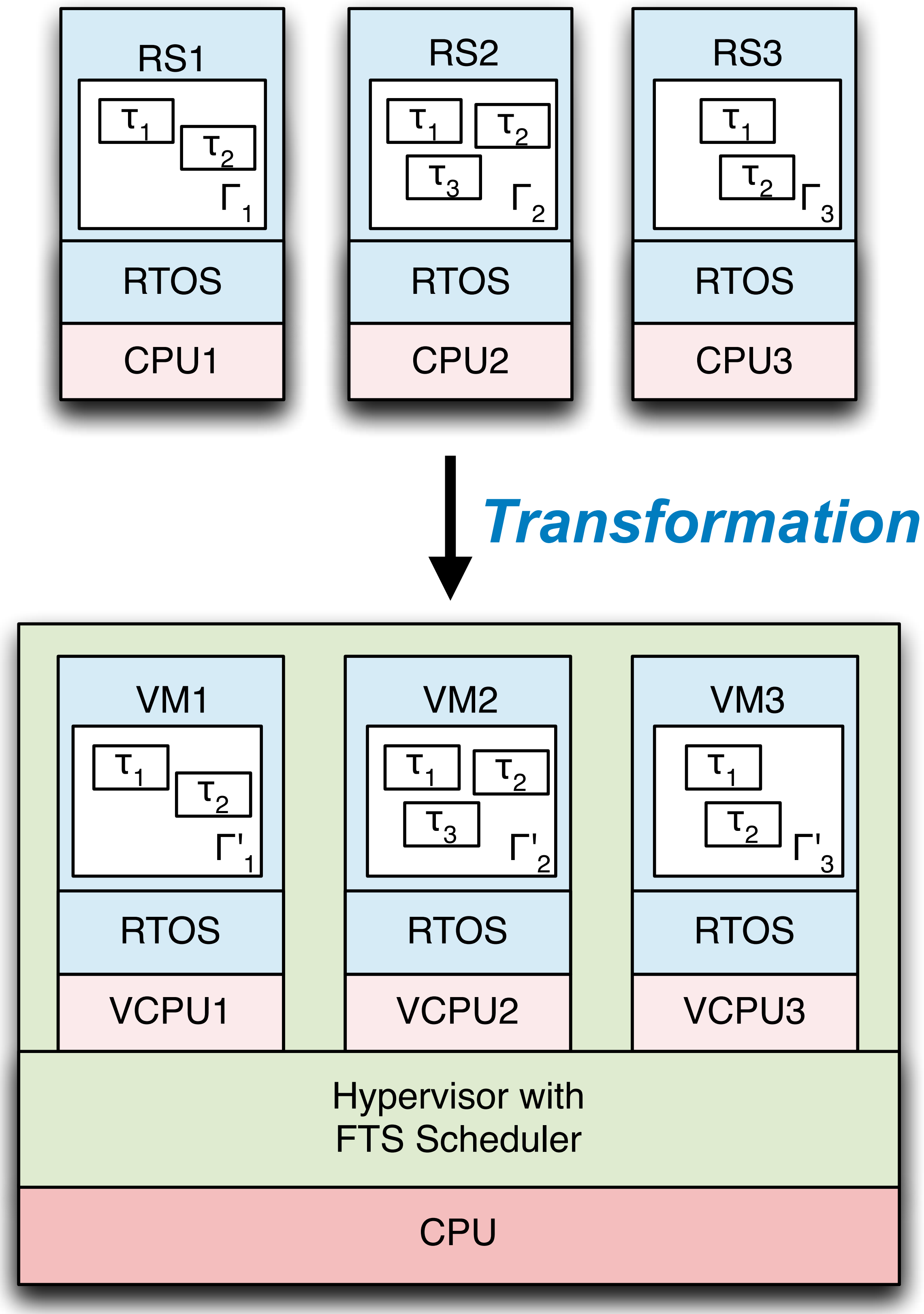


- **Problems**
  - Complexity
    - Distribution
  - Dependability
    - Redundancy
  - Different requirements on
    - Functionality
      - High Level API
      - Low level API
    - Timing
      - Non real time
      - Soft real time
      - Hard real time
    - OS platform
      - Linux
      - RTOS
- **Integration?**



Goal

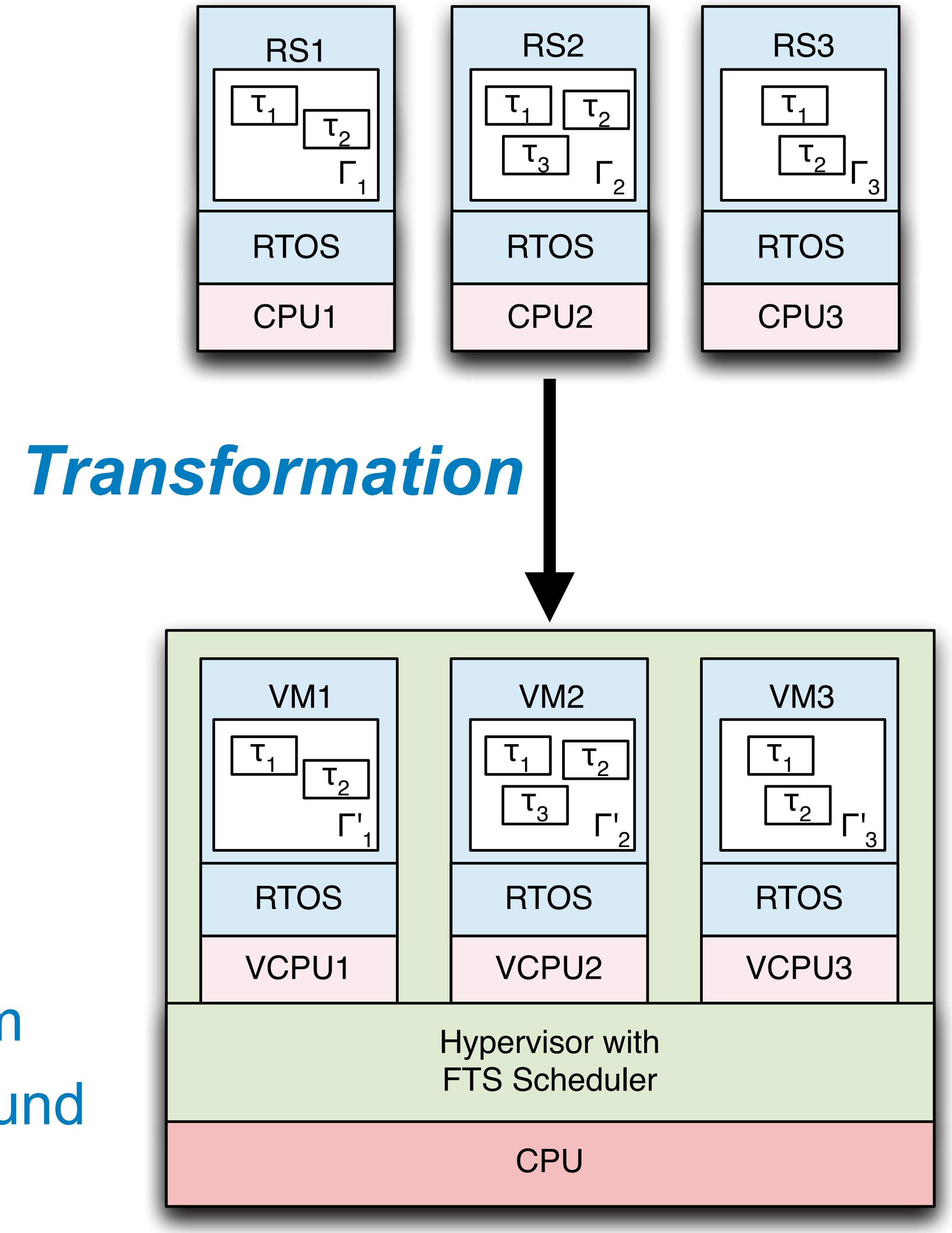
- **Assumptions:**
  - Given real-time systems $RS_1,\ldots,RS_n$ executing their periodic tasksets $\Gamma_1,\ldots,\Gamma_n$
  - RTOS using its own scheduler
    - EDF or
    - RM
  - Executed on dedicated CPU



- **Goal: Execution of $RS_1,\ldots,RS_n$ as virtual machines $VM_1,\ldots,VM_n$ on a single CPU**
  - Question 1: What CPU to use for the virtual real-time system?
  - Question 2: How to schedule the virtual machines while preserving full virtualization?

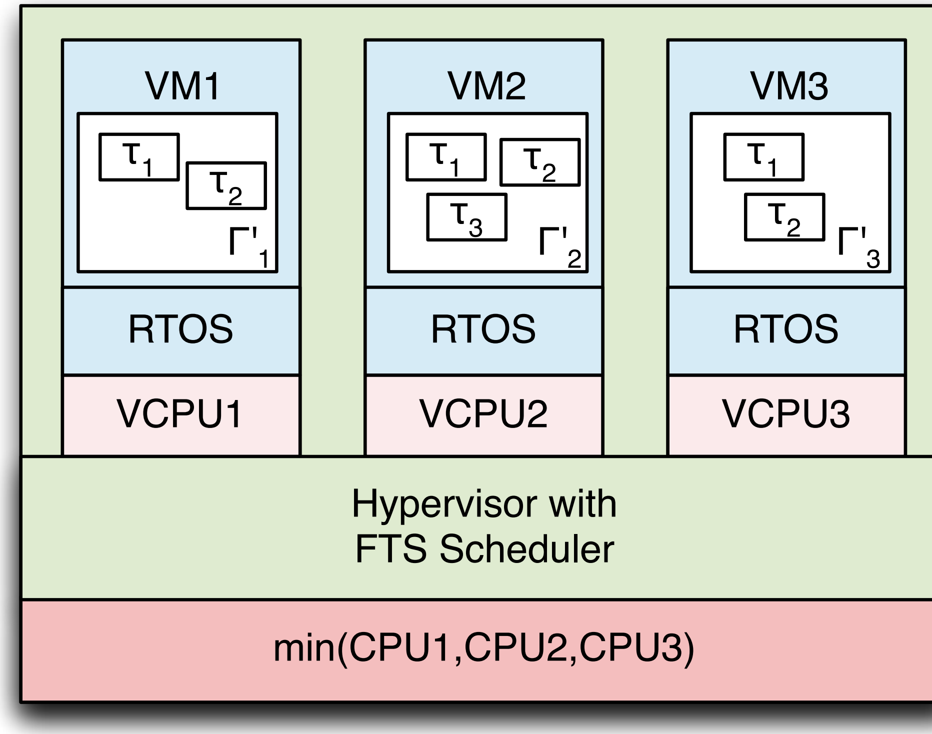
Transformation from Γ_i to Γ'_i

- **Question 1: What CPU to use for the virtual real-time system?**
  - Idea: Normalization on slowest given RS.
  - Assumptions
    - Comparable CPUs!
    - Infinitesimal time slicing

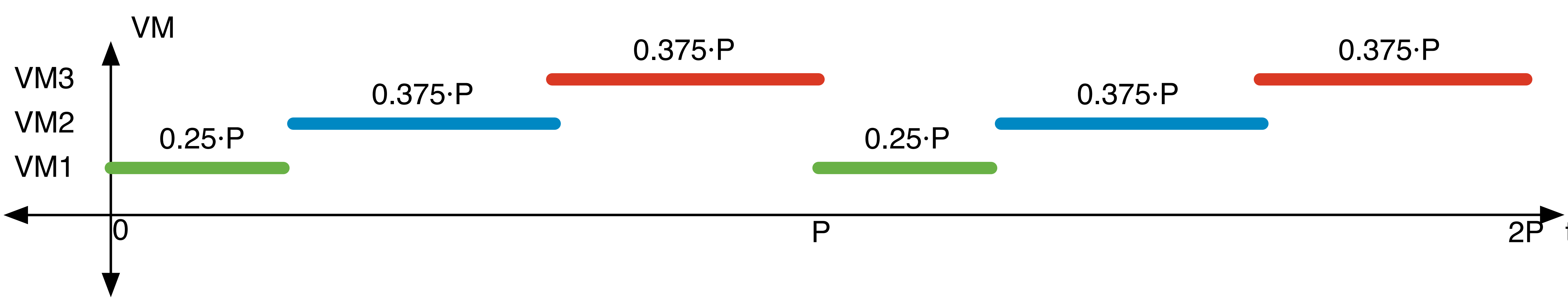


- **Normalized system executable?**
  - Examination of its utilization
  - Decision depends on applied scheduling algorithm
  - Speedup based on Utilization and Scheduling Bound of applied scheduling algorithm

Transformation from Γ_i to Γ'_i (EDF)



- **Time slices based on proportional weight W of the normalized tasksets Γ_{is}.**
- **Example:**
  - $U(\Gamma_{1s}) = 0.5$, $U(\Gamma_{2s}) = 0.75$, $U(\Gamma_{3s}) = 0.75$
  - $W(\Gamma_{1s}) = 0.25$, $W(\Gamma_{2s}) = 0.375$, $W(\Gamma_{3s}) = 0.375$
- **Assumption: Infinitesimal time slicing: P→0**

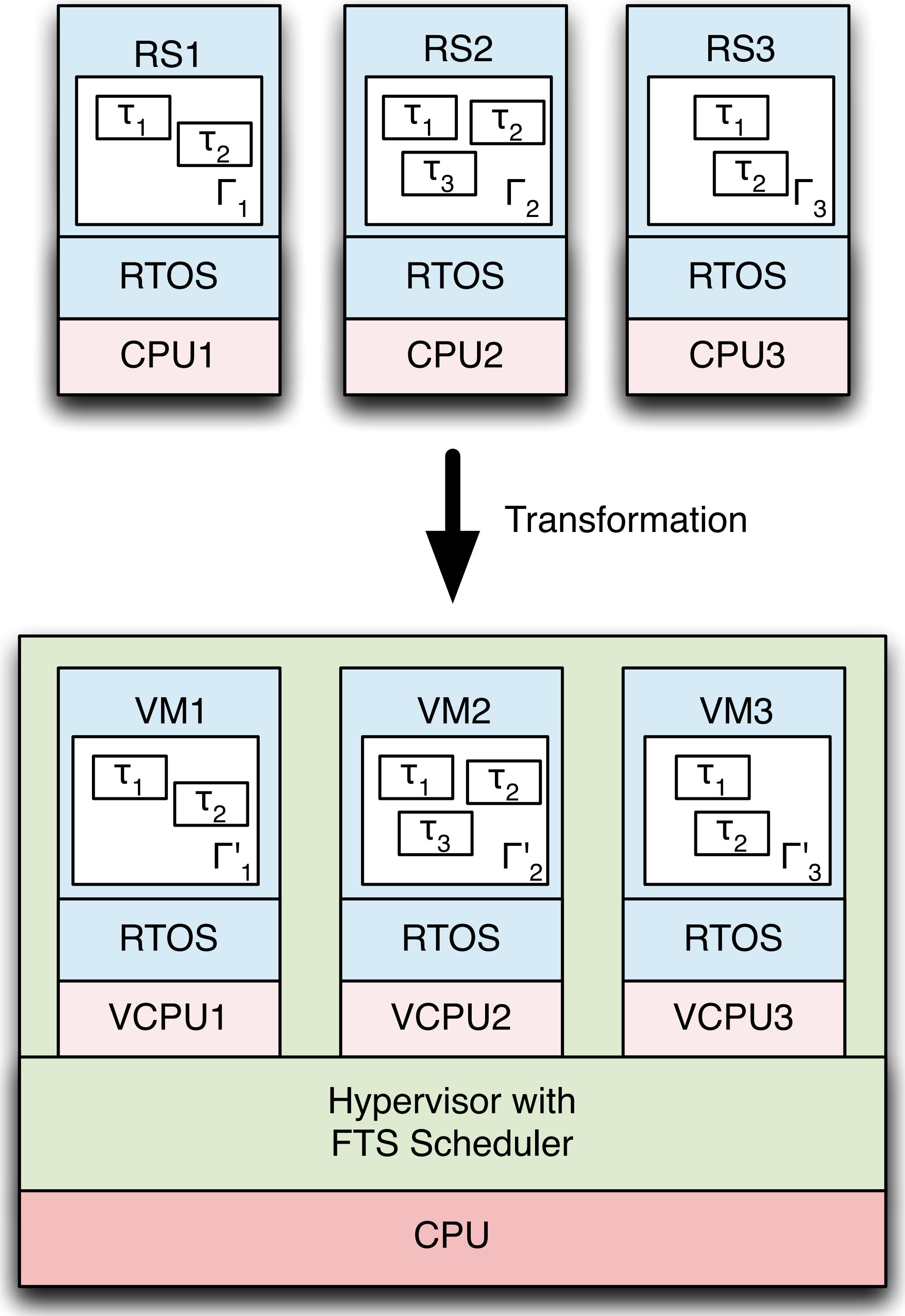


- **Example:**
  - Speedup: $U(\Gamma_{1s}) = 0.5$, $U(\Gamma_{2s}) = 0.75$, $U(\Gamma_{3s}) = 0.75 \Rightarrow$ S=2

    $\Rightarrow U(\Gamma_1') = 0.5$, $U(\Gamma_2') = 0.375$, $U(\Gamma_3') = 0.375$
  - Speedup S is a design hint for choosing the minimal needed CPU speed.

Summary

- **Goal: Execution of $RS_1,\dots,RS_n$ as virtual machines $VM_1,\dots,VM_n$ on a single CPU**
  - Question 1: What CPU to use for the virtual real-time system?
    - Normalization on slowest given RS.
    - Speedup based on EDF/RM
    - Transformation of $\Gamma_1,\dots,\Gamma_n$ into $\Gamma'_1,\dots,\Gamma'_n$
    - Up to now not realizable
      - Assumption: Infinitesimal time slicing!
  - Question 2: How to schedule the virtual machines while preserving full virtualization?
    - Idea: Usage of single time slot periodic partitions
      - Period length not infinitesimal!
      - But how to choose the period?



Resource Partitions

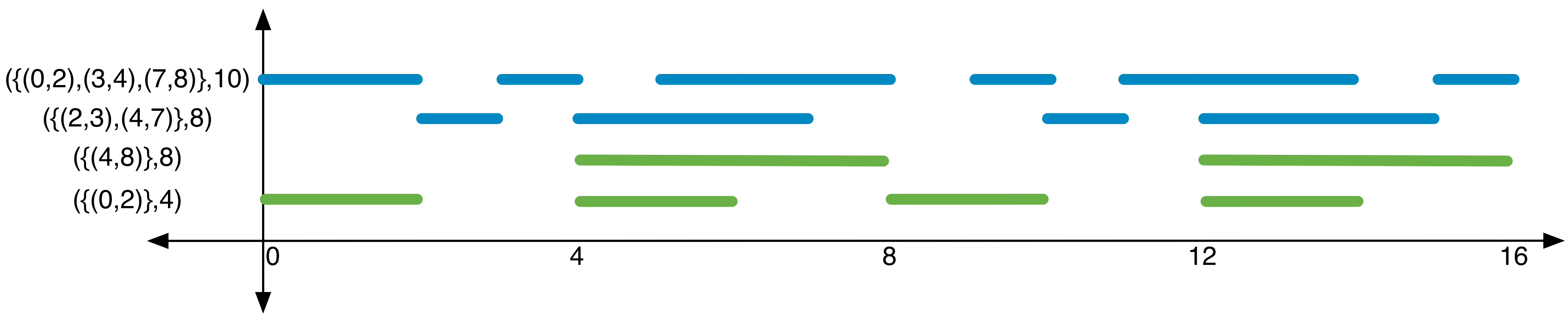
A resource partition Π is a tuple (γ,P)

$\gamma = \{(S_1,E_1),\ldots,(S_N,E_N)\}$ with $0 \le S_1 < E_1 < \ldots < S_N < E_N$ for some $N \ge 1$

P is the partition period.

Physical resource is available only during intervals $(S_i + j \cdot P, E_i + j \cdot P)$, $1 \le i \le N$, $j \ge 0$.

- **A resource partition with multiple time pairs (N>1) is called Multiple Time Slot Periodic Partition**
- **A resource partition with only one time pair (N=1) is called Single Time Slot Periodic Partition**
- **Examples:**



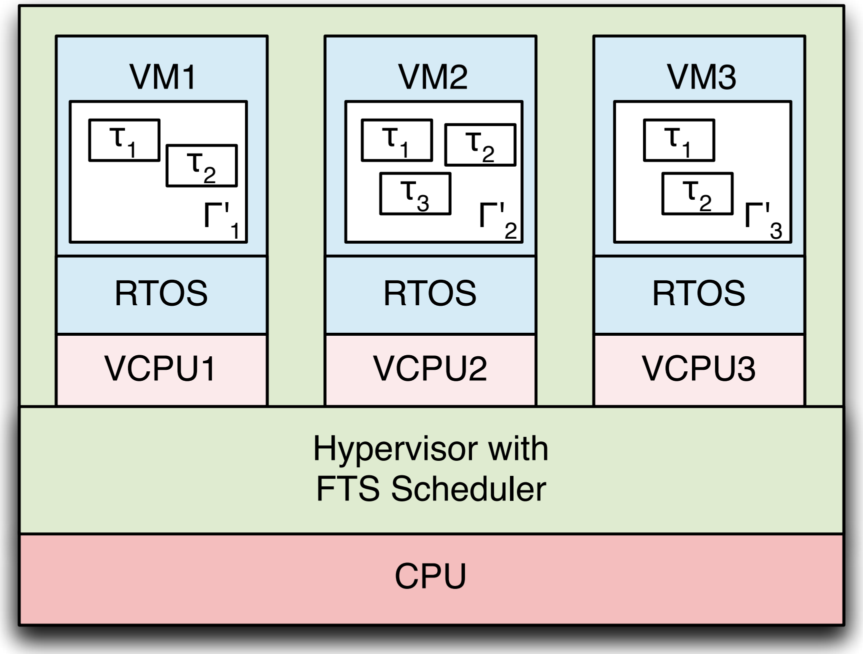
Virtual real-time system

Virtual real-time system consists of:

$$\Gamma'_i = \{\tau_k(T_k, C_k) | k = 1, \ldots, m\}, \; i = 1, \ldots, n$$

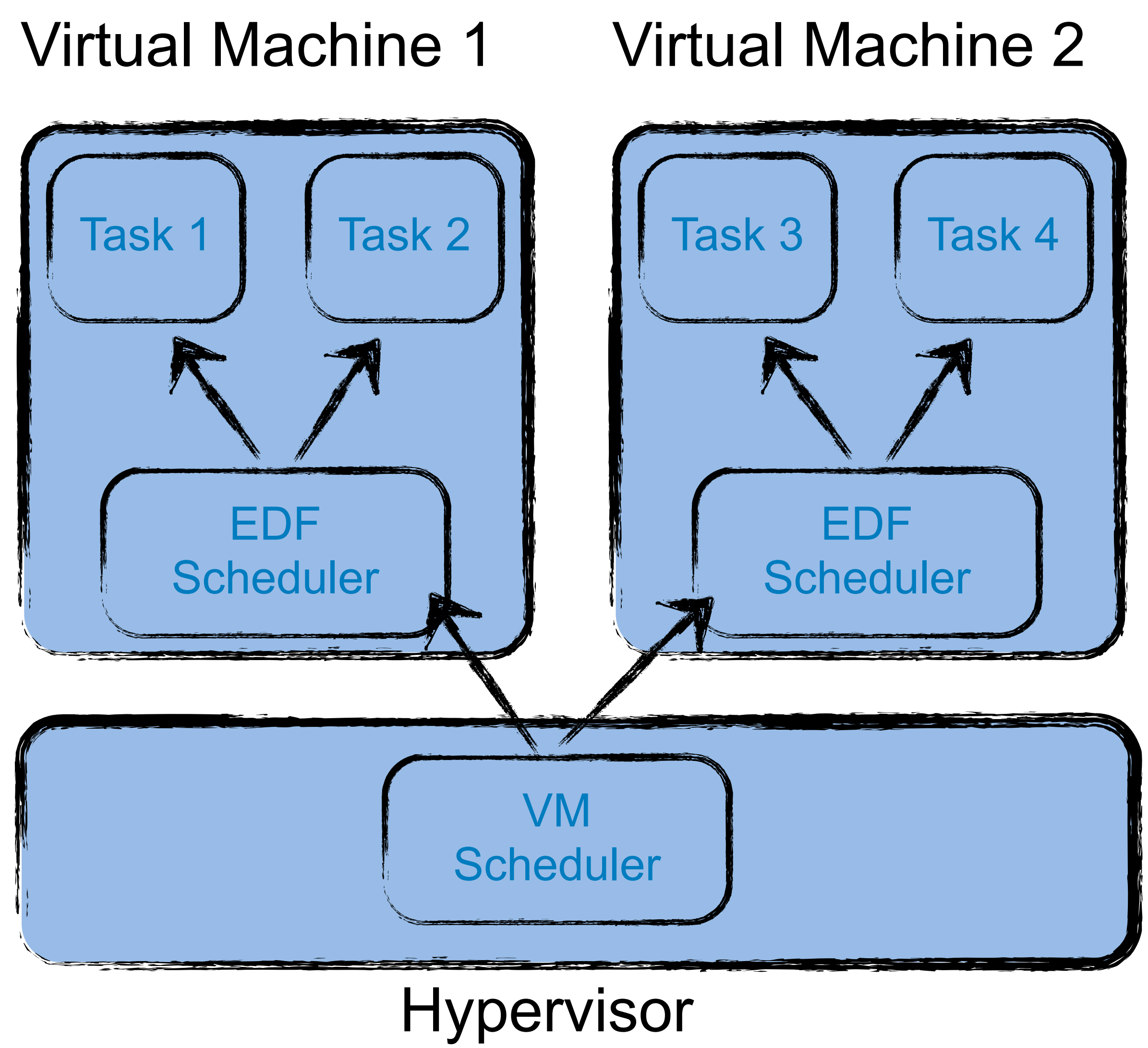
$$\Pi_i = \{\{(S_i, E_i)\}, P \mid S_1 = 0, E_i = S_i + \alpha_i \cdot U(\Gamma'_i) \cdot P, \; S_i = E_{i-1}\}$$, with P being equal for all Π_i.

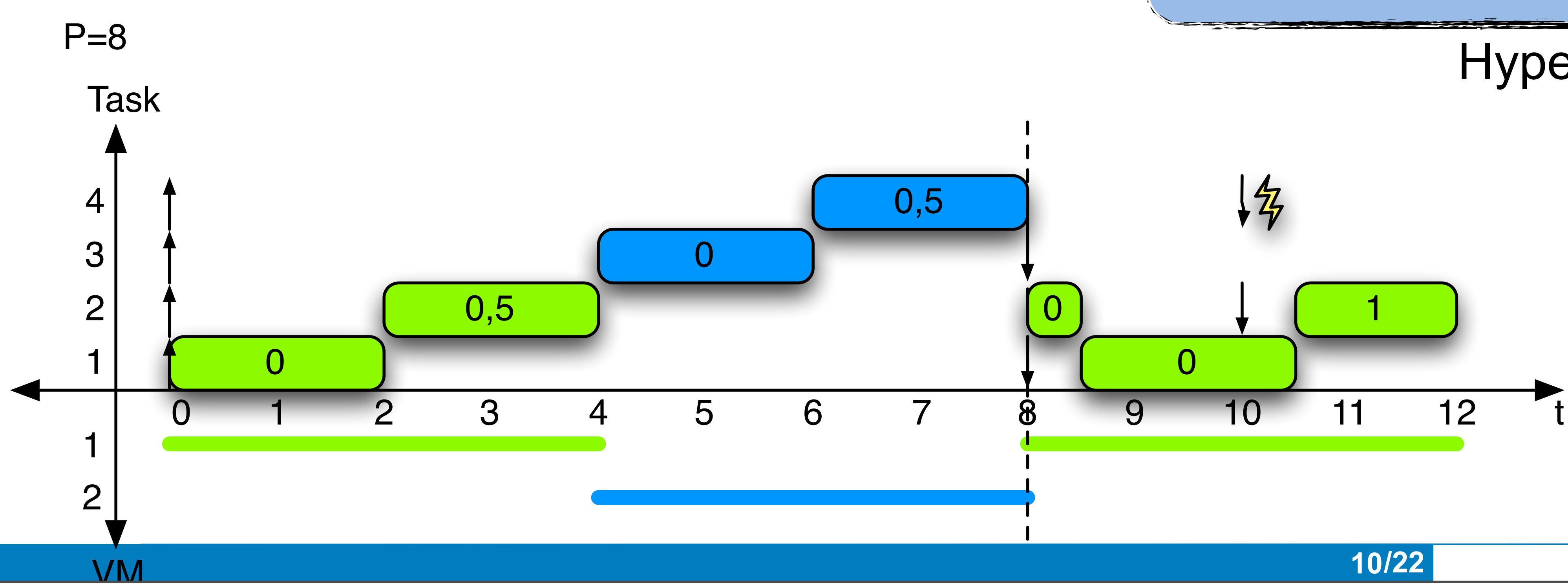
- **Activation length of a virtual machine depends on the utilization of Γ'_i being $U(\Gamma'_i)$ and on its used scheduling algorithm**
- **α_i is a scaling factor depending on the utilization bound of the applied scheduling algorithm**
  - **$\alpha_i = 1$ for EDF**
  - **$\alpha_i = 1/U_{lub}$ for RM**



The period p cannot be choosen arbitrarily!

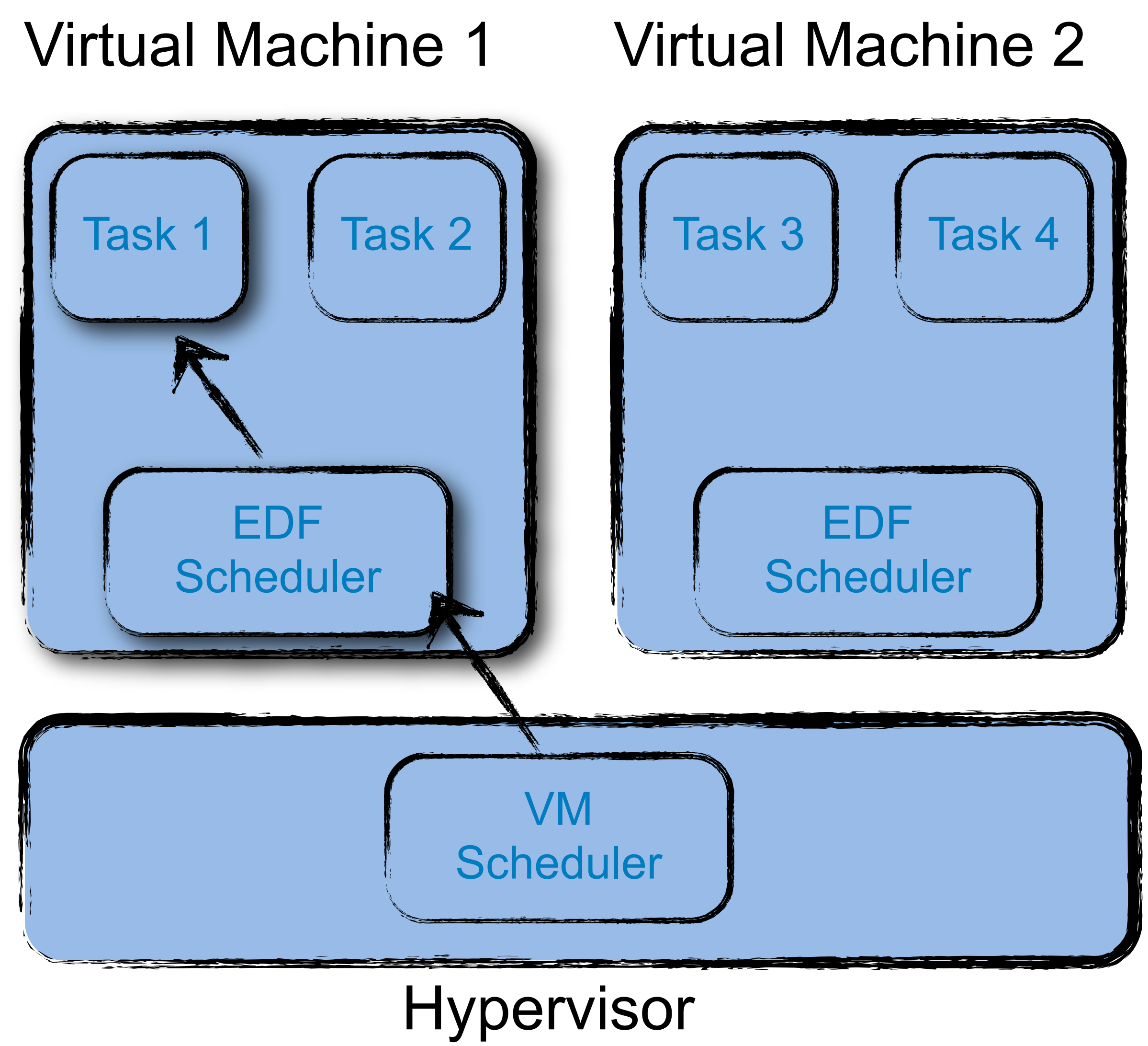
- Two EDF virtual machines
  - $T_1=\{(2,8),(2.5,10)\}$
  - $T_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=\{(0,4),8\}$
  - $STSPP_2=\{(4,8),8\}$

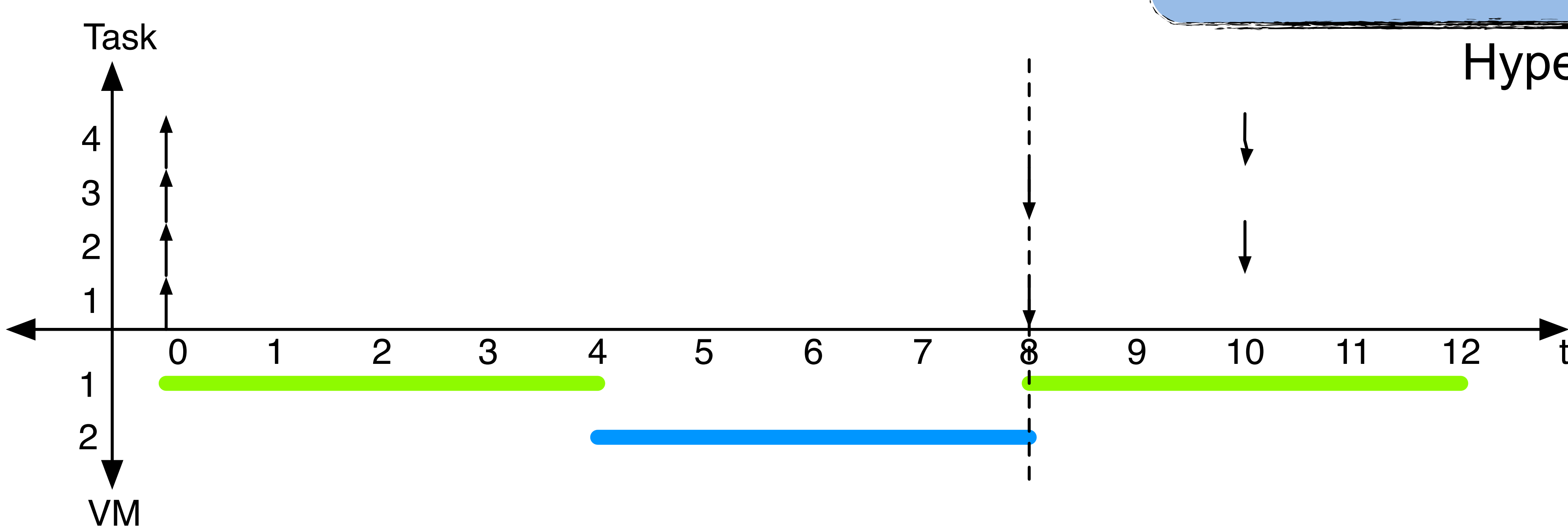




The period p cannot be choosen arbitrarily!

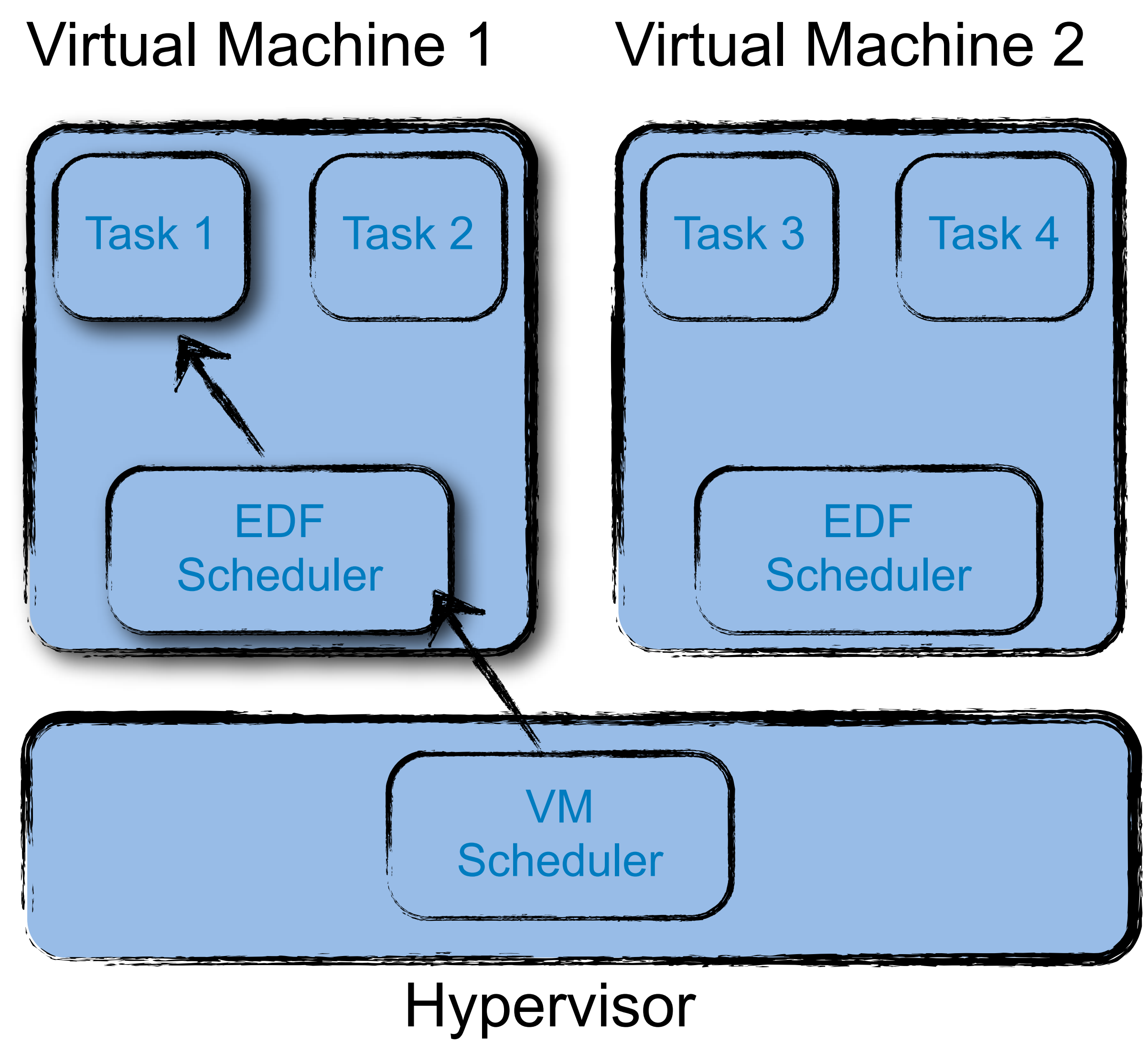
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$

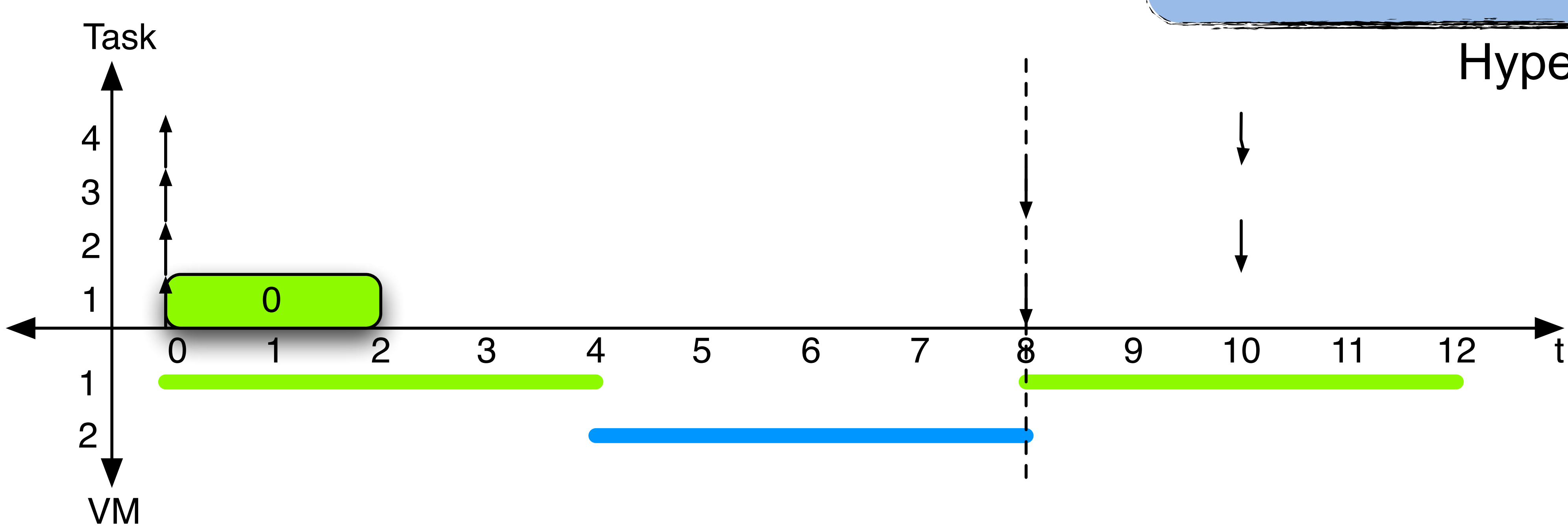




The period p cannot be choosen arbitrarily!

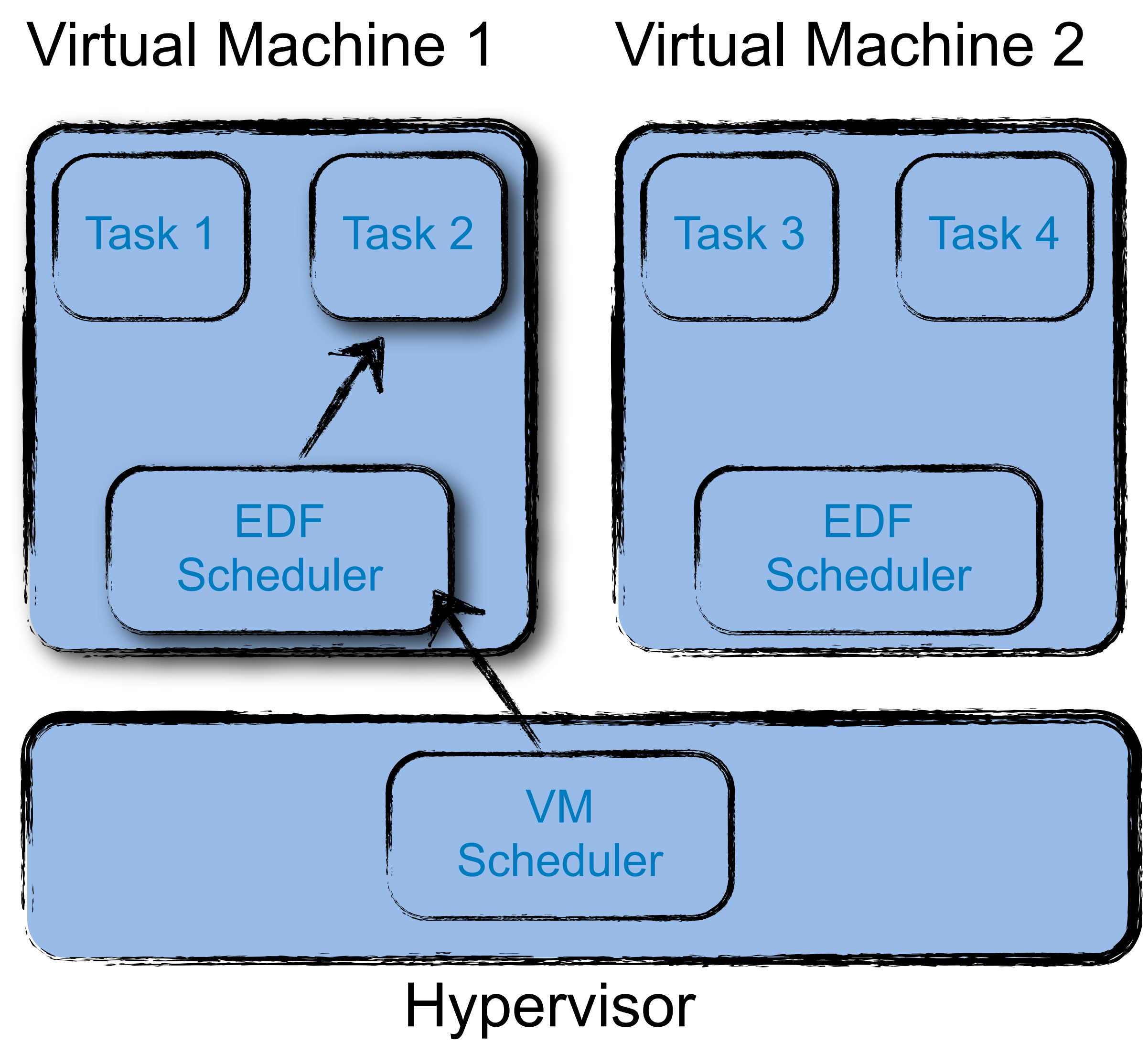
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$

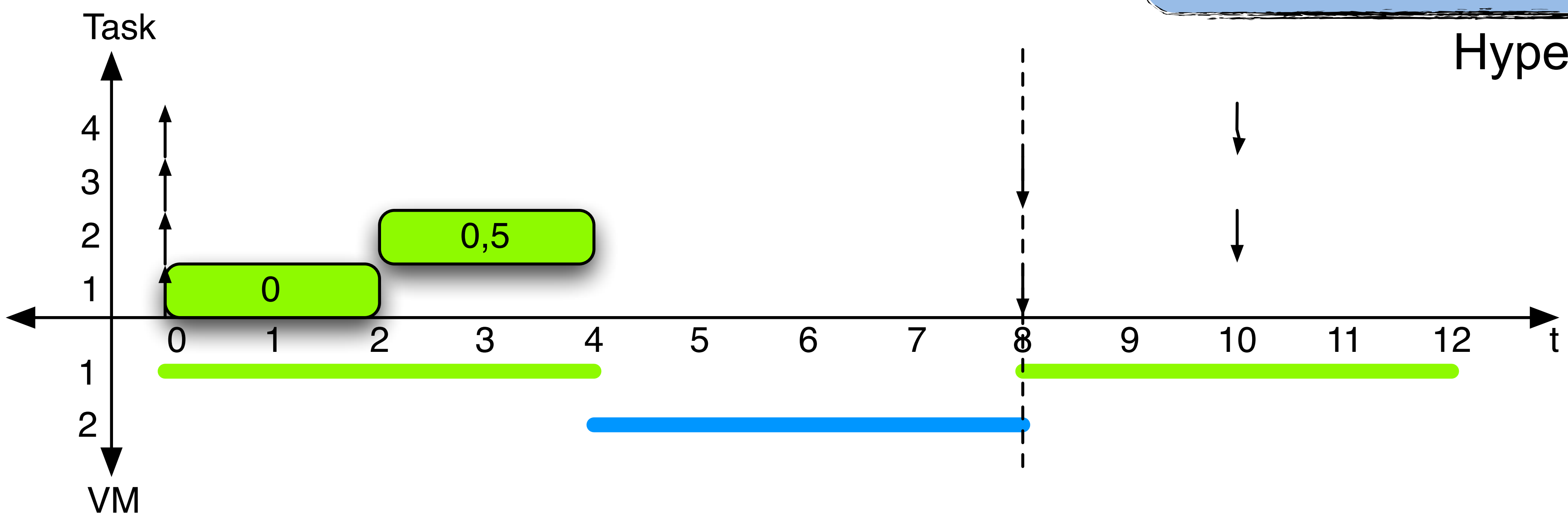




The period p cannot be choosen arbitrarily!

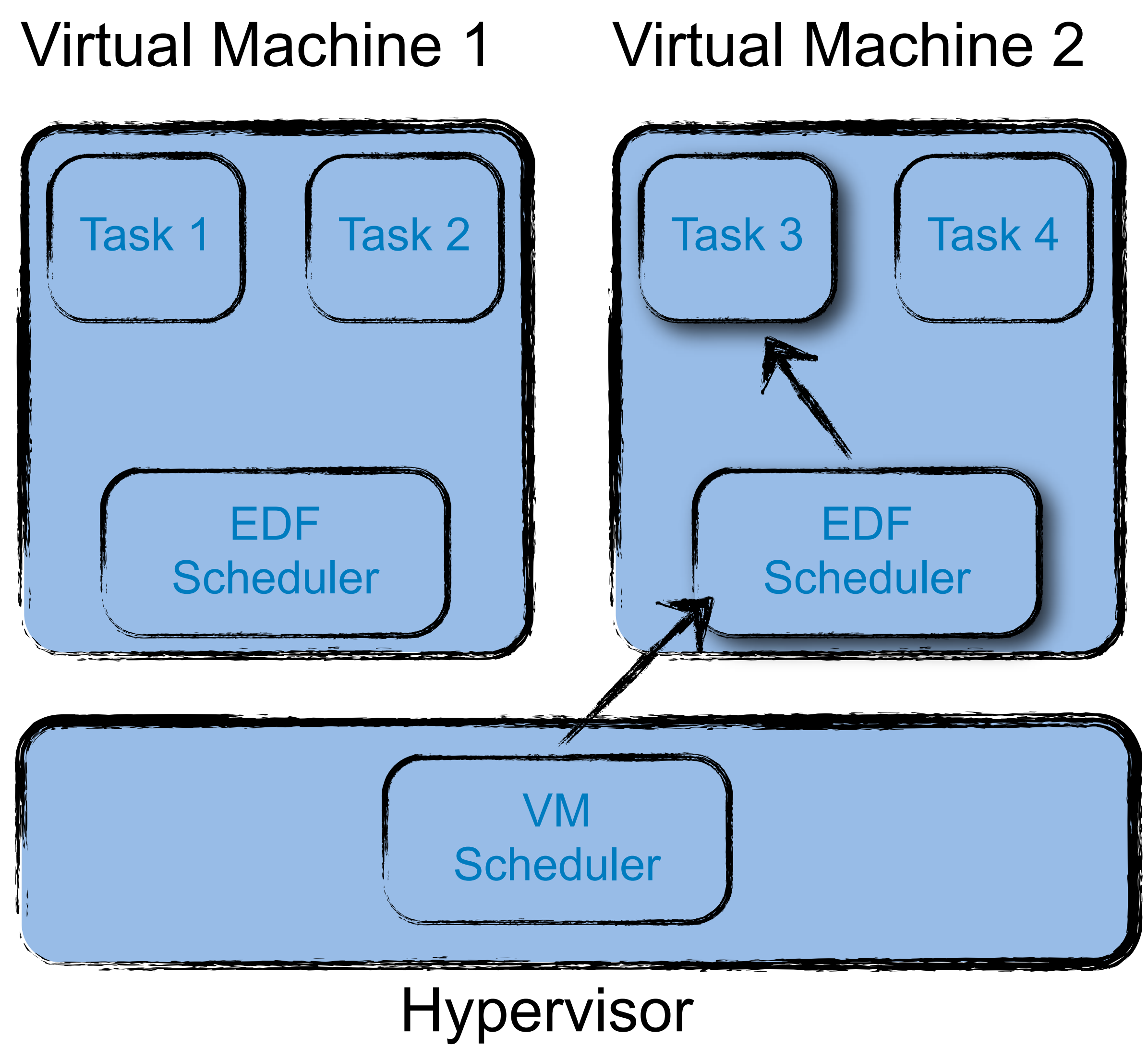
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$

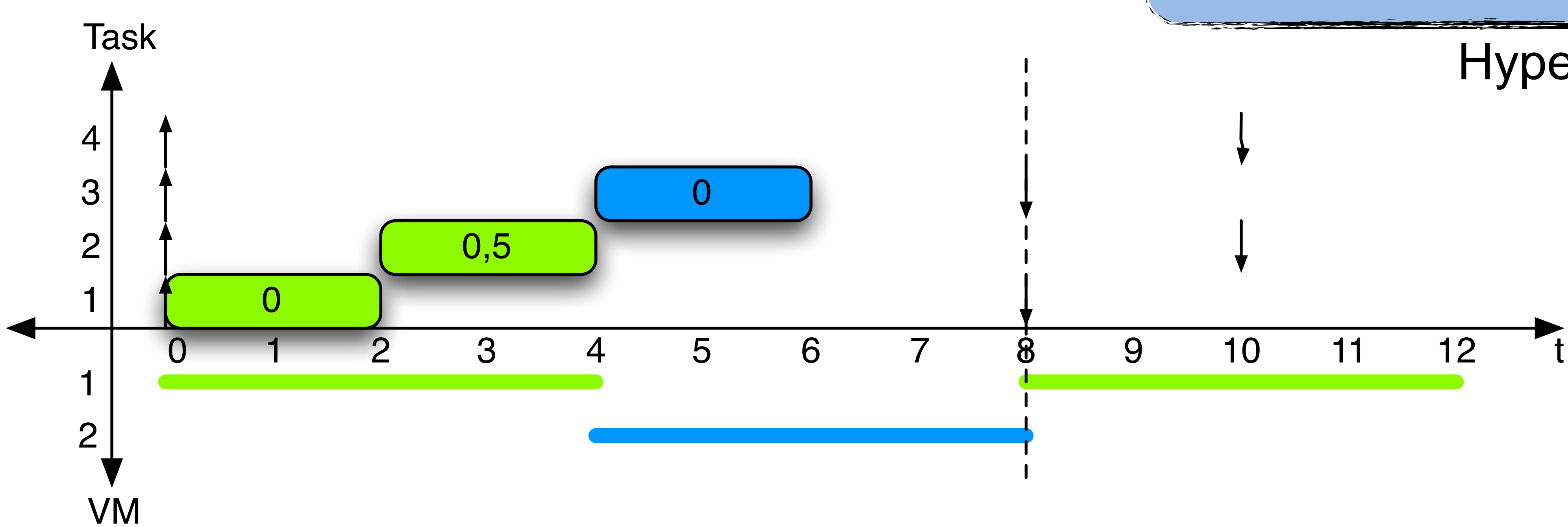




The period p cannot be choosen arbitrarily!

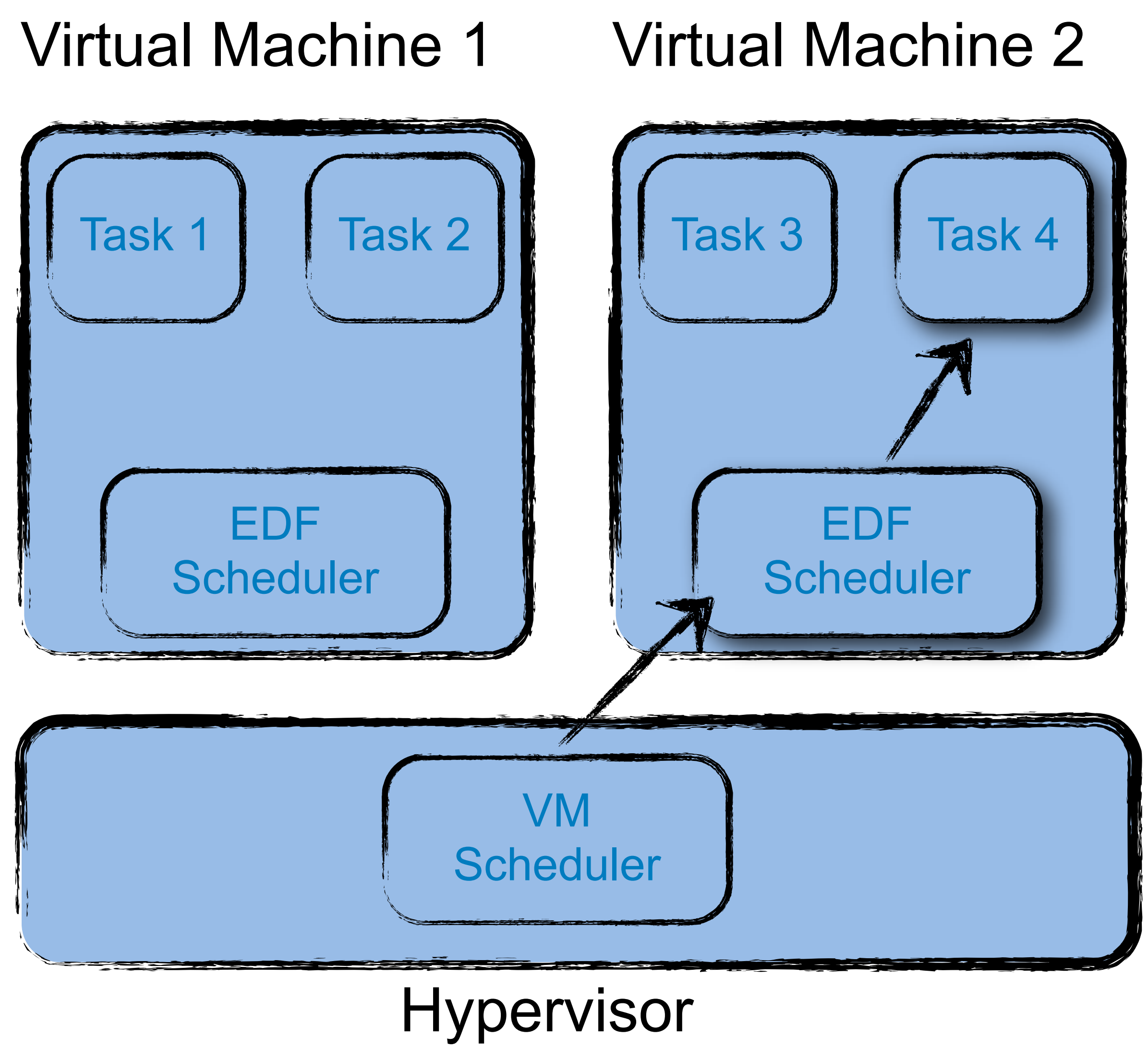
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$

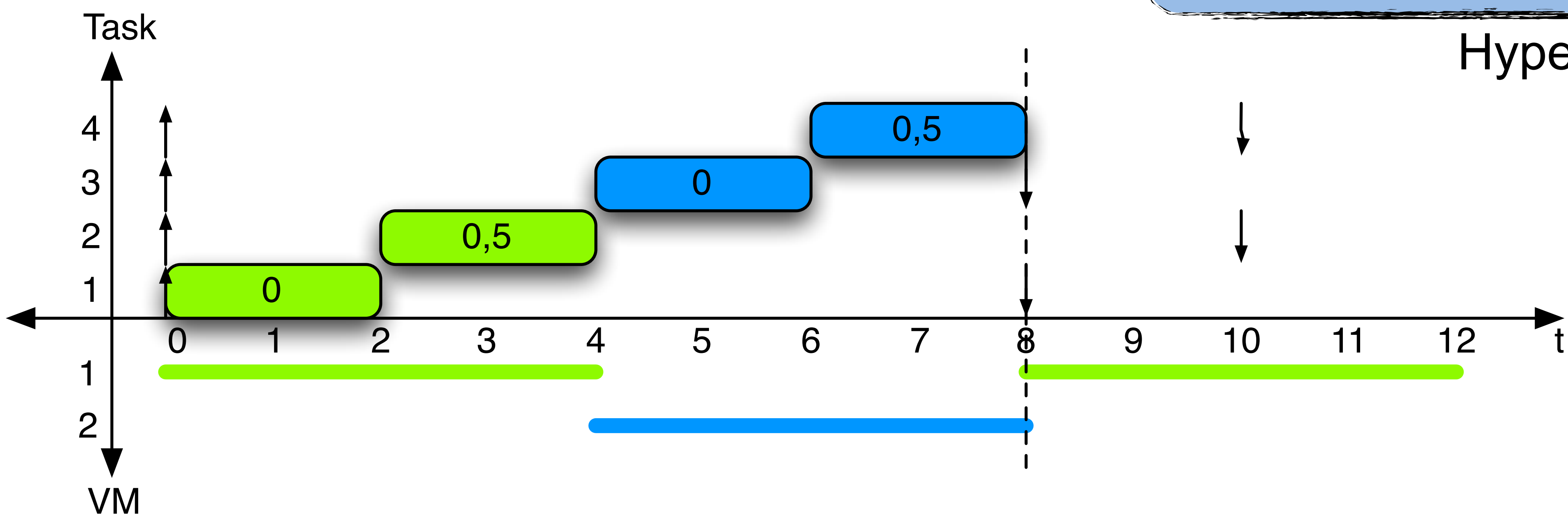




The period p cannot be choosen arbitrarily!

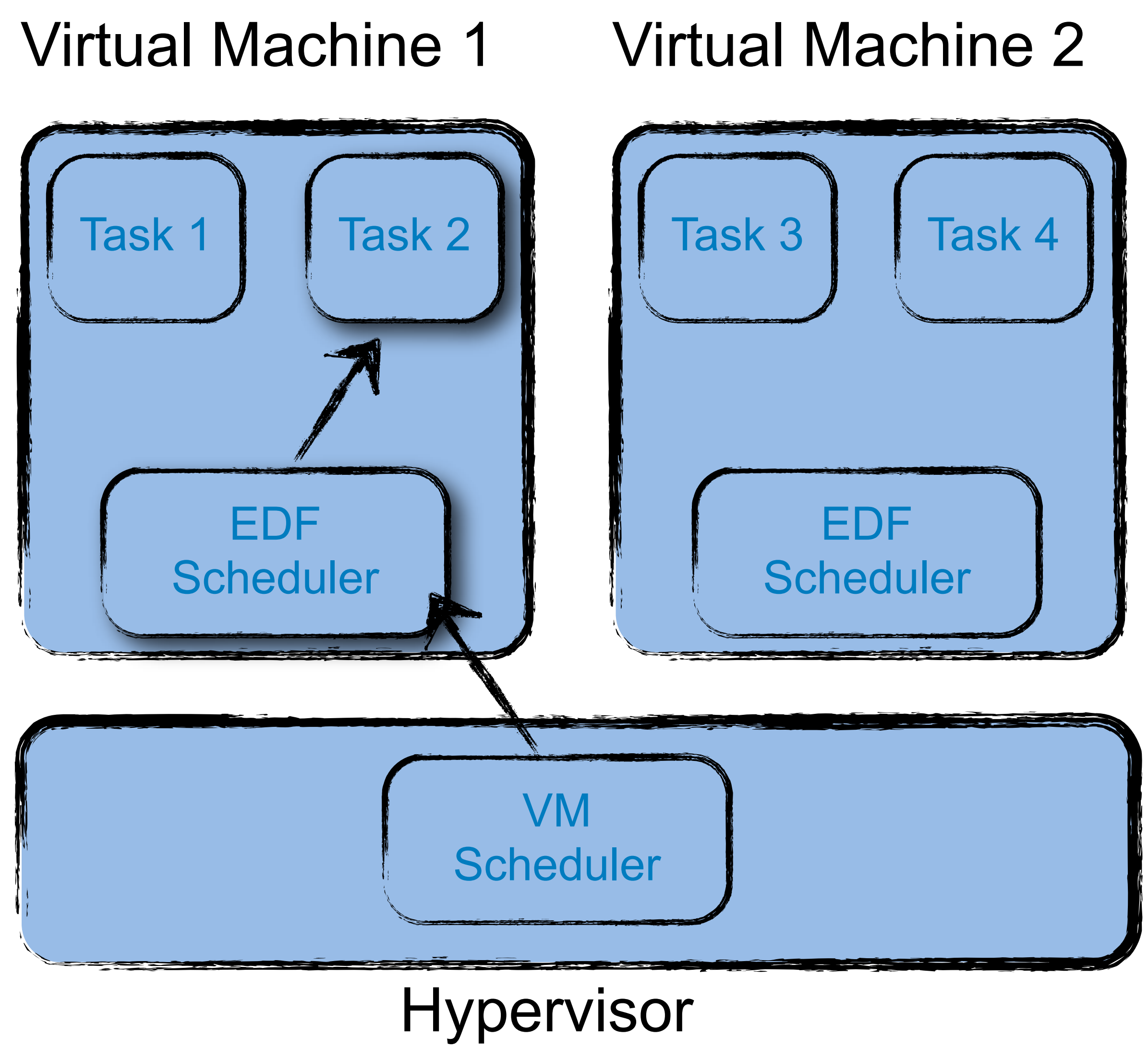
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$

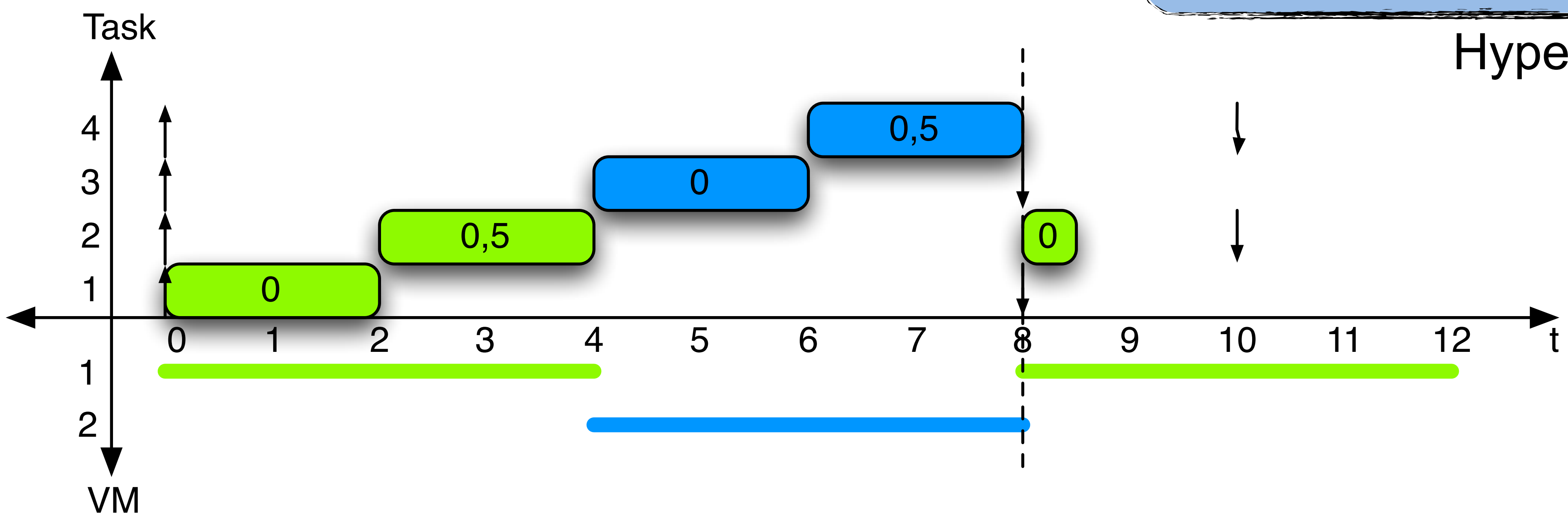




The period p cannot be choosen arbitrarily!

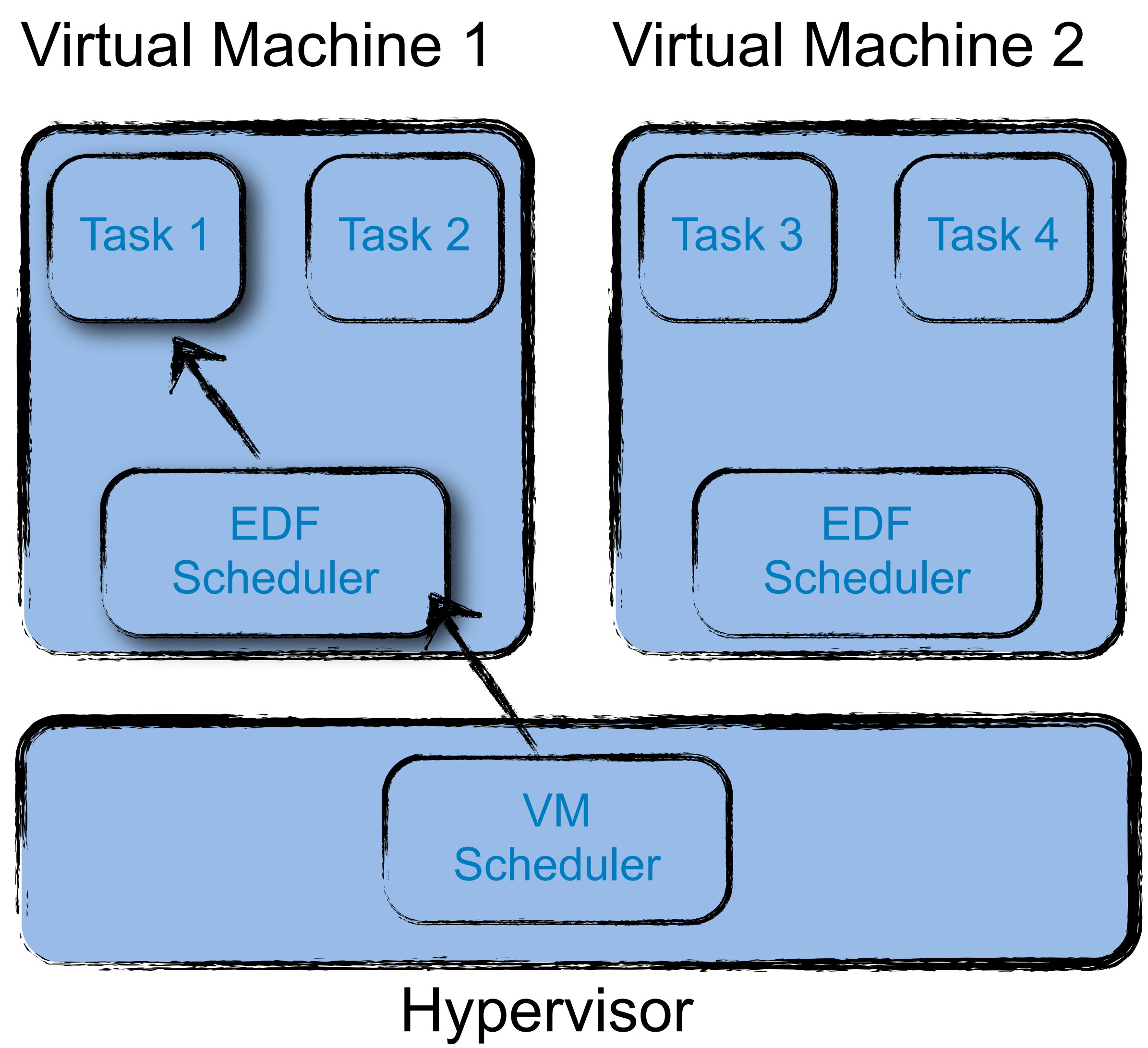
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$

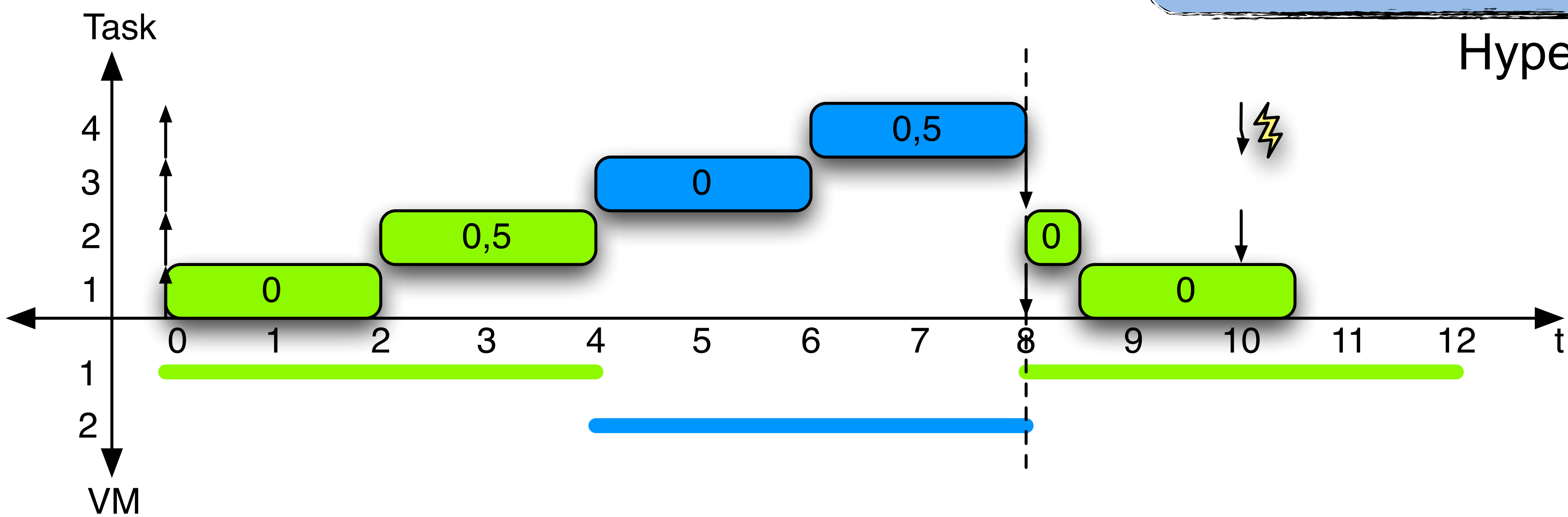




The period p cannot be choosen arbitrarily!

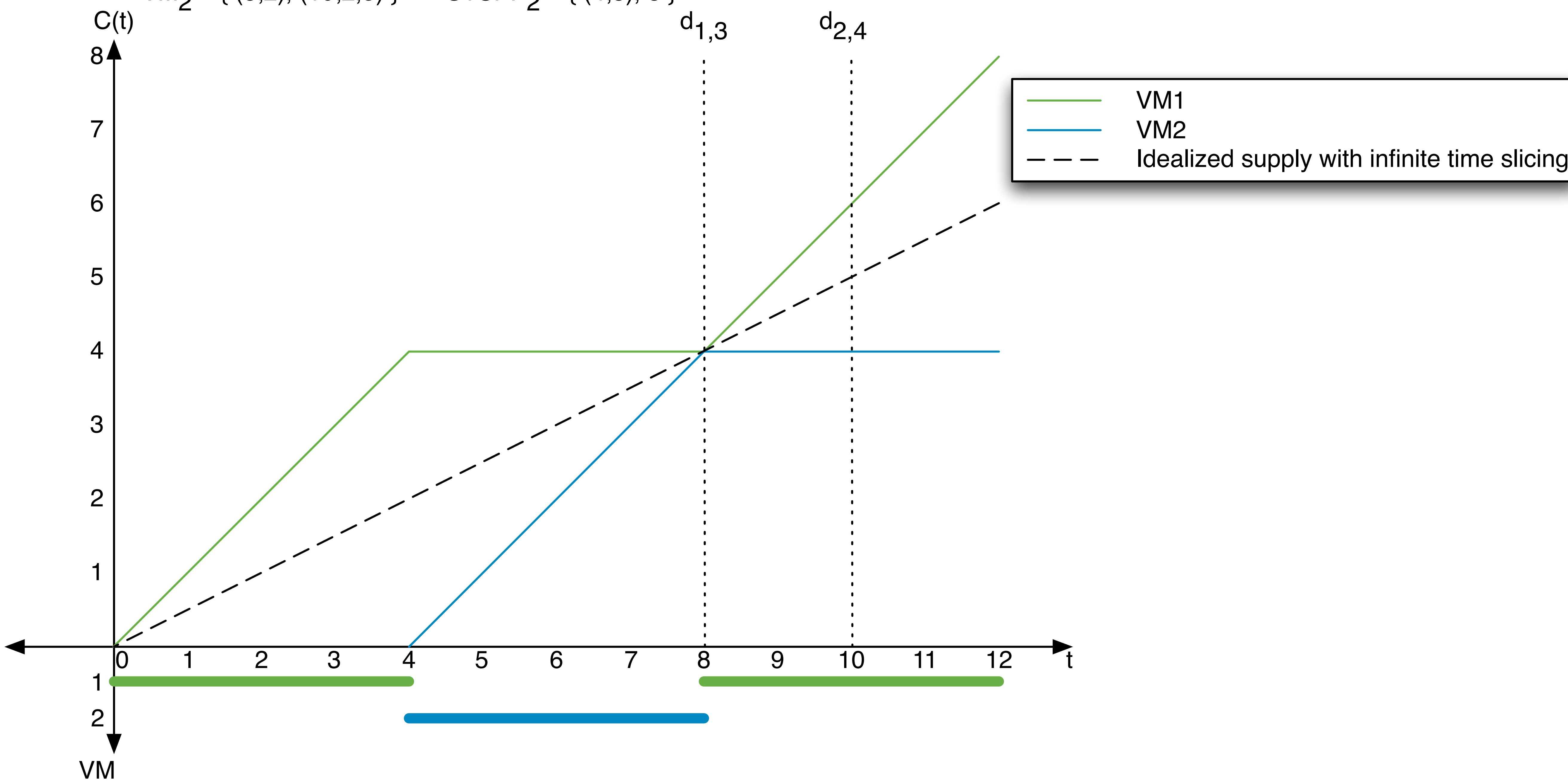
- Two EDF virtual machines
  - $VM_1=\{(2,8),(2.5,10)\}$
  - $VM_2=\{(2,8),(2.5,10)\}$
  - $U_1=U_2=0.5$
  - $STSPP_1=(\{(0,4)\},8)$
  - $STSPP_2=(\{(4,8)\},8)$





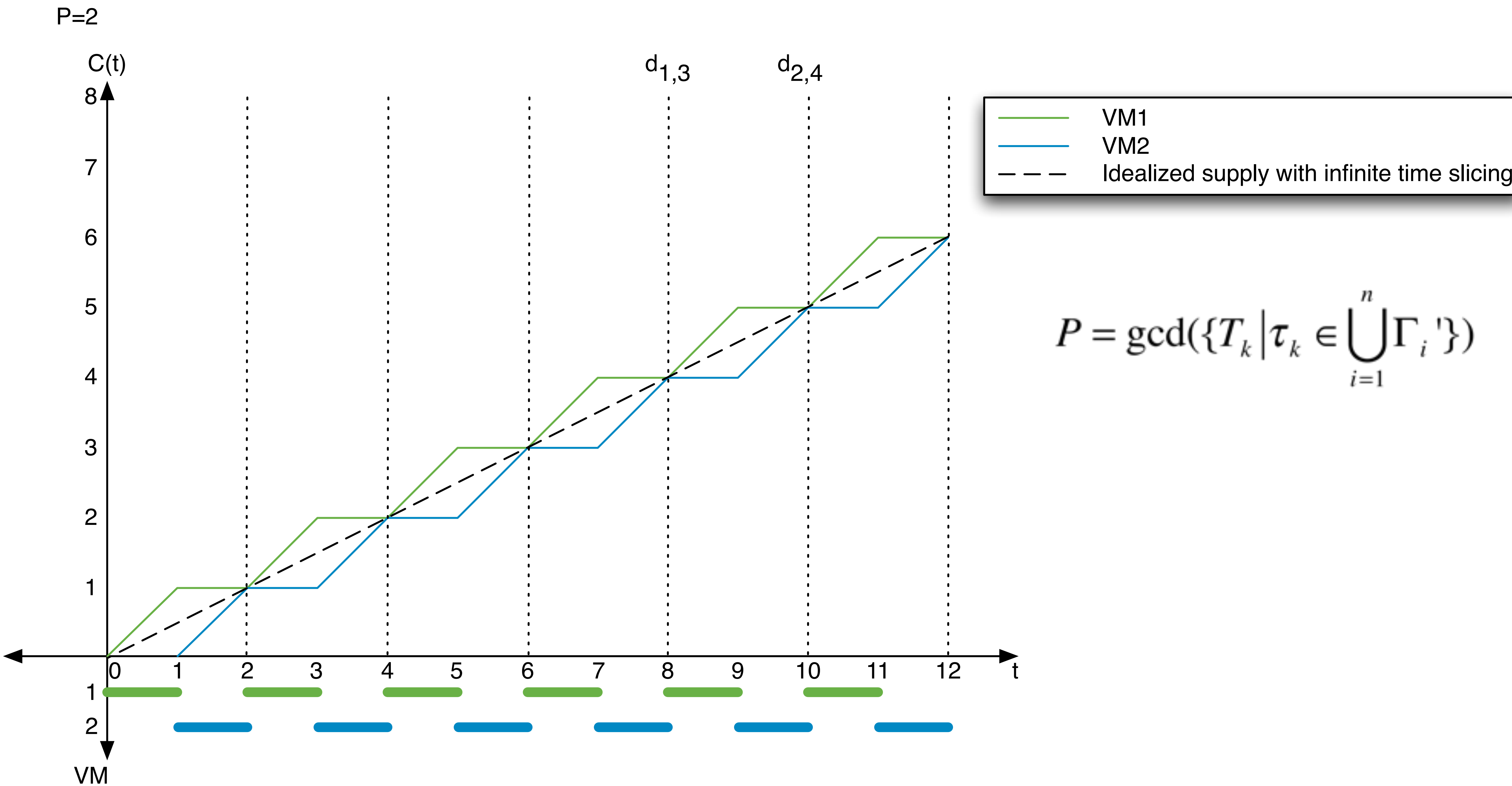
Supply Function of Example Schedule

P=8 $VM_1 = \{ (8;2), (10;2,5) \}$ $STSPP_1 = \{ (0;4), 8 \}$
$VM_2 = \{ (8;2), (10;2,5) \}$ $STSPP_2 = \{ (4;8), 8 \}$



Supply Function of Example Schedule

HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig



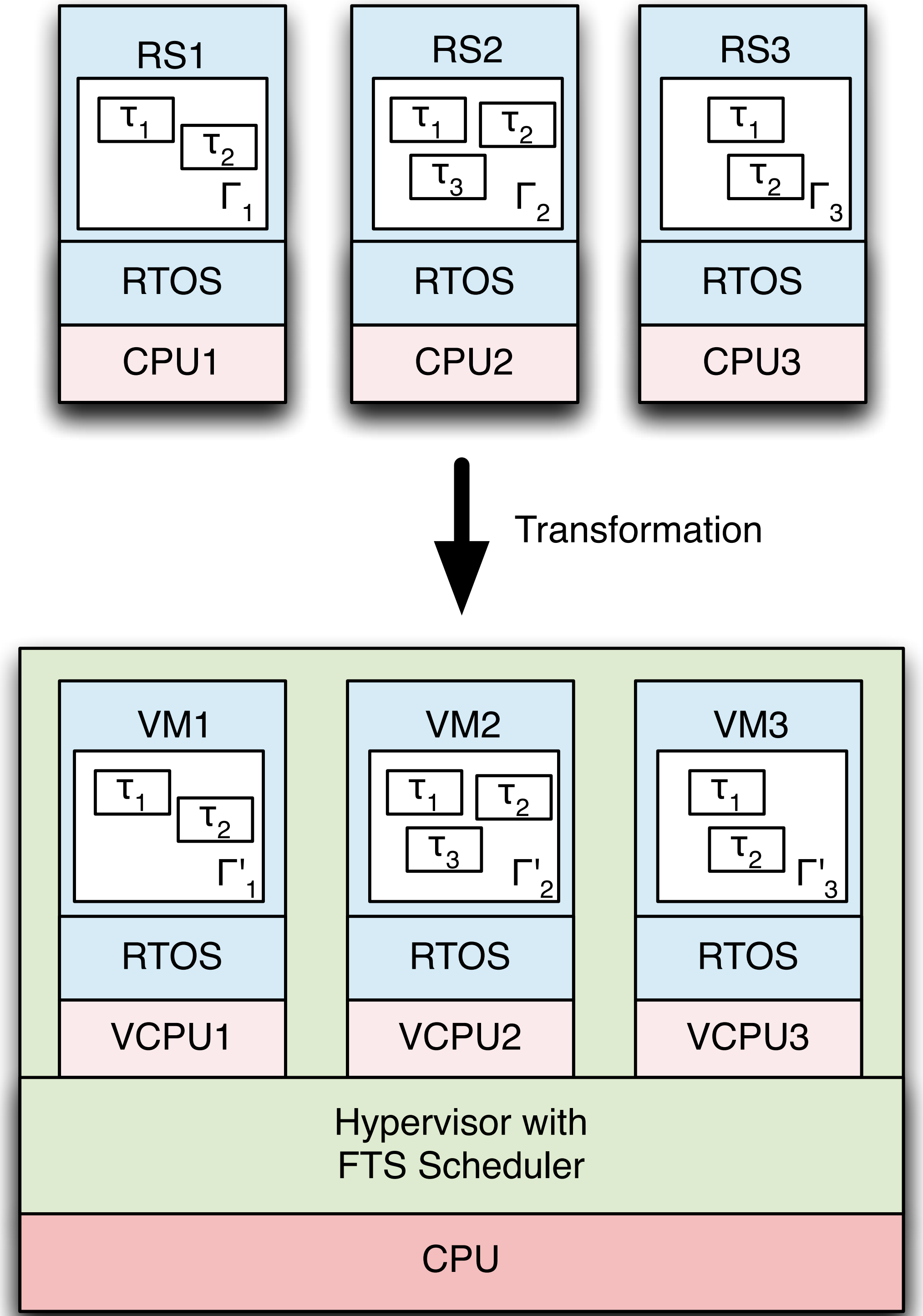
$$P = \gcd(\{T_k \mid \tau_k \in \bigcup_{i=1}^{n} \Gamma_i'\})$$

Summary

- **Goal: Execution of $RS_1,\ldots,RS_n$ as virtual machines $VM_1,\ldots,VM_n$ on a single CPU**
  - Question 1: What CPU to use for the virtual real-time system?
    - Normalization on slowest given RS.
    - Speedup based on EDF/RM
    - Transformation of $\Gamma_1,\ldots,\Gamma_n$ into $\Gamma'_1,\ldots,\Gamma'_n$
    - Theorem 1 ensures the needed allocation!
    - Assumption: Infinitesimal time slicing!
  - Question 2: How to schedule the virtual machines while preserving full virtualization?
    - Usage of single time slot periodic partitions

$$P = \gcd(\{T_k \mid \tau_k \in \bigcup_{i=1}^{n} \Gamma_i'\})$$

      - P may get very small
      - Tradeoff between interrupt latency and switching overhead

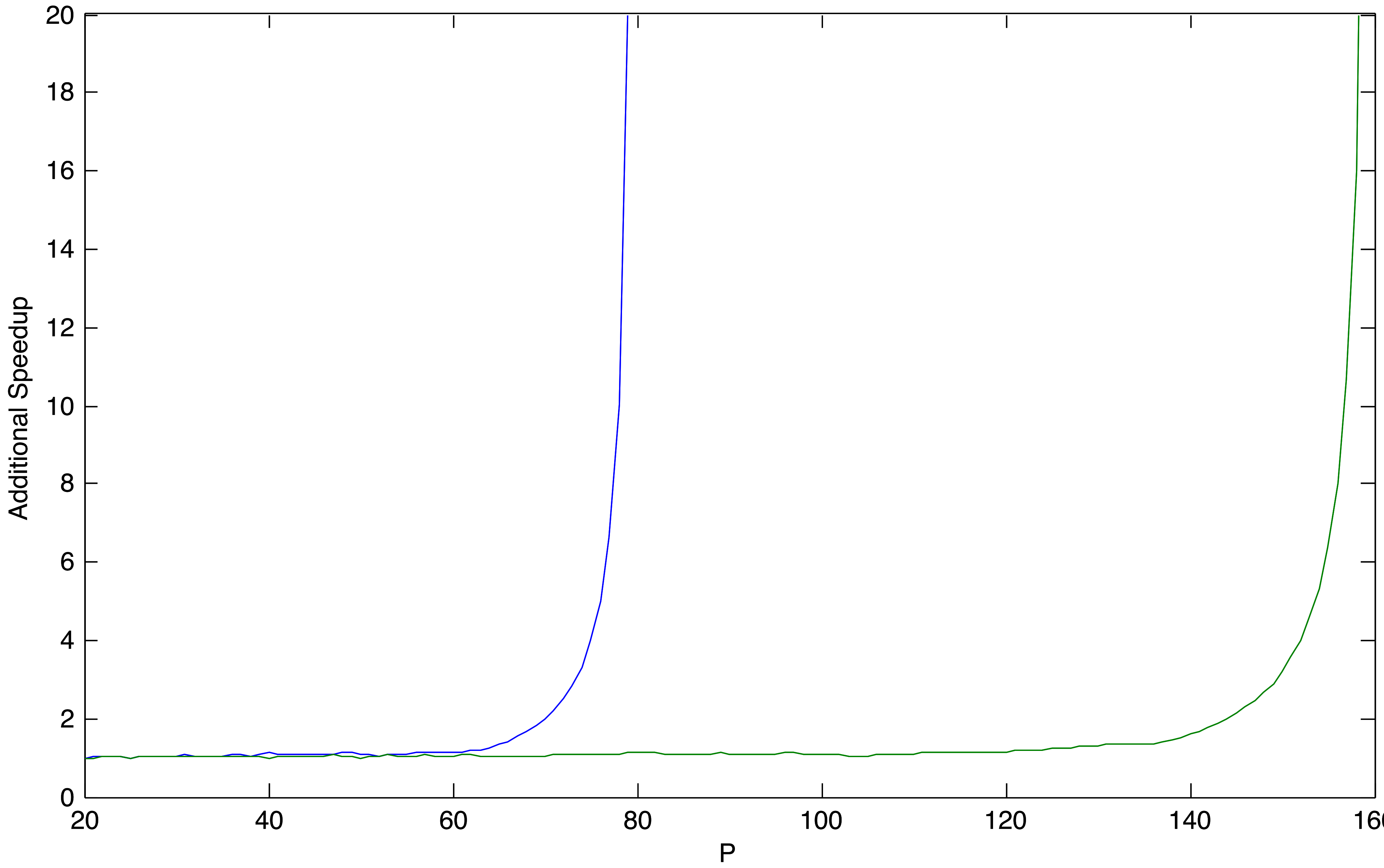


Reducing the switching overhead

HEINZ NIXDORF INSTITUTE
University Paderborn
Design of Parallel Systems
Prof. Dr. rer. nat. Franz J. Rammig

- **Small values of P lead to a larger weight of the switching overhead**
- **Is it possible to increase the value of P without missing deadlines?**
- **Idea:**
  - Choose a P larger than the GCD of all deadlines
  - Calculate for every VM at its tasks deadlines within the hyperperiod
    - the needed computation time N(t)
    - the assigned computation time Z(t)
  - The fraction N(t)/Z(t) is the necessary speedup to fulfill the requested computation time
  - The maximum fraction within the hyperperiod is the speedup to apply the choosen STSPP length P

Reducing the switching overhead

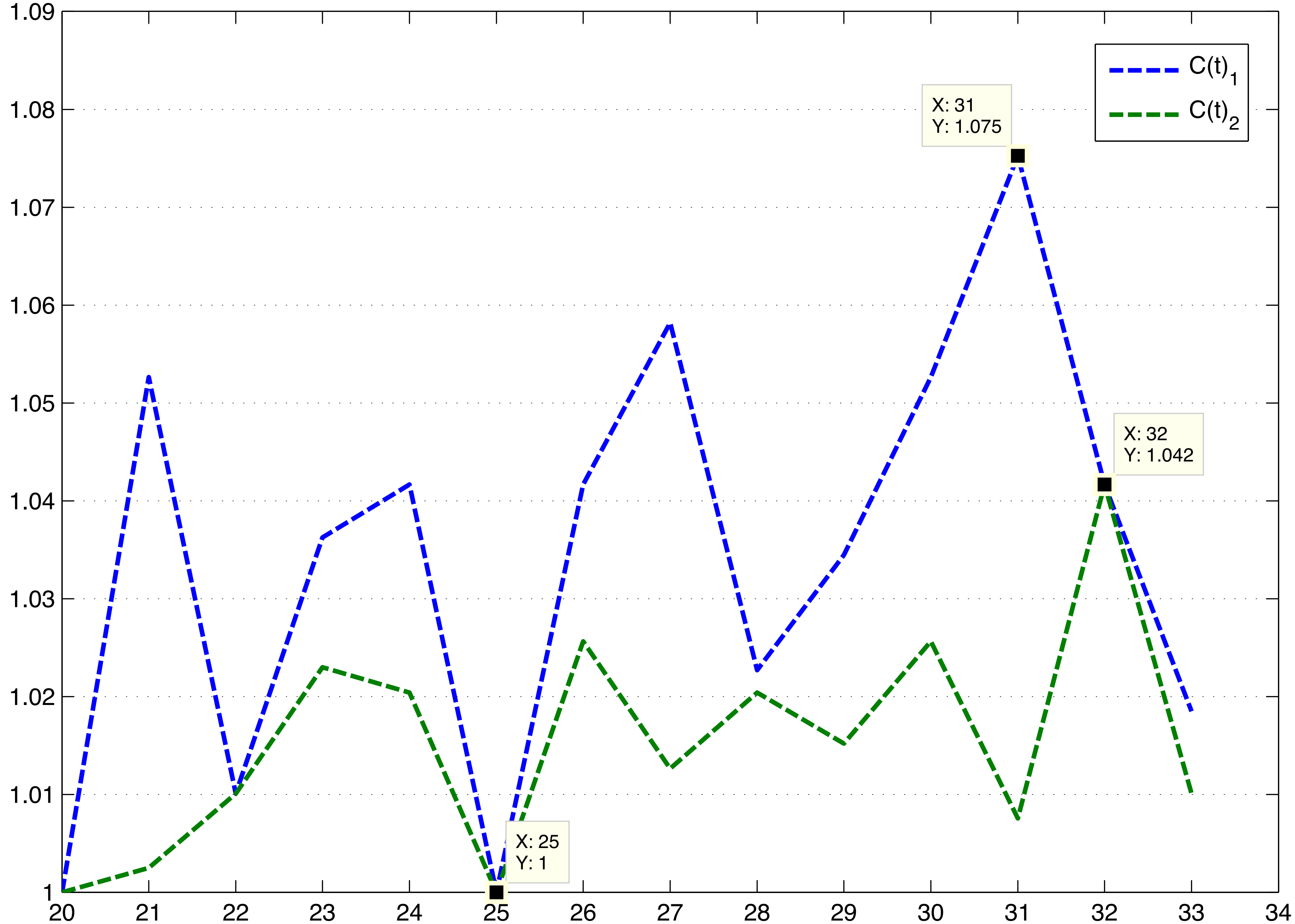


$$VM_1 = \{(40,10),(100,25)\}$$
$$VM_2 = \{(80,16),(200,60)\}$$

$$P = GCD(\{40,80,100,200\}) = 20$$

$$STSPP_1 = \{(0,10),20\}$$
$$STSPP_2 = \{(10,20),20\}$$

Reducing the switching overhead



$VM_1 = \{(40,10),(100,25)\}$

$VM_2 = \{(80,16),(200,60)\}$

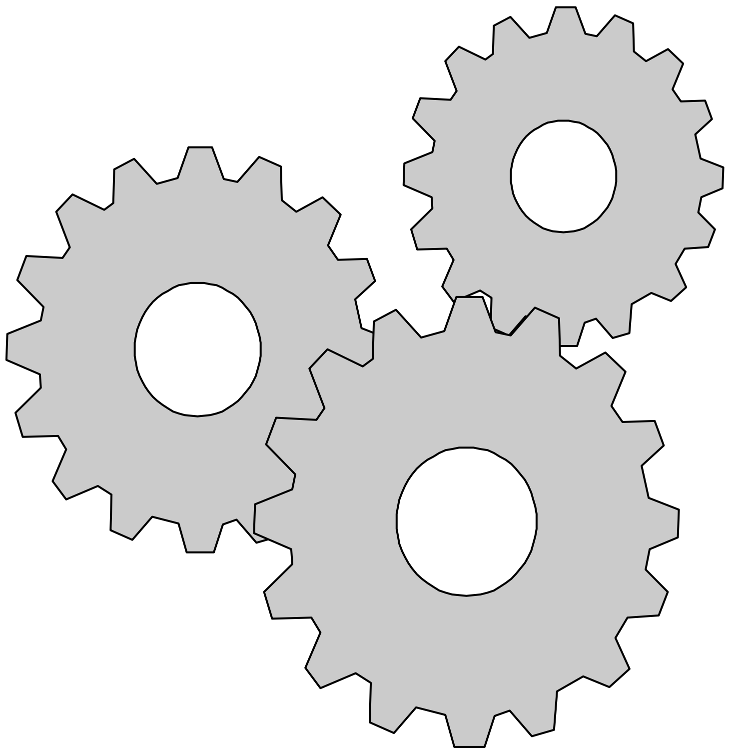
$P = GCD(\{40,80,100,200\}) = 20$

$STSPP_1 = \{(0,10),20\}$

$STSPP_2 = \{(10,20),20\}$

Conclusion

- Goal:
  - Transform given real-time systems into a virtualized system
    - Performance of Host CPU?
    - Full virtualization
    - VM Scheduling using FTS
- We developed a simple methodology to derive
  - CPU speedup
  - STSPPs
  - FTS Schedule
- Switching overhead can be severe with a small P
  - Reduction by additional speedup
  - Analysis to determine the required speedup for a given P



Conclusion

- Goal:
  - Transform given real-time systems into a virtualized system
    - Performance of Host CPU?
    - Full virtualization
    - VM Scheduling using FTS
- We developed a simple methodology to derive
  - CPU speedup
  - STSPPs
  - FTS Schedule
- Switching overhead can be severe with a small P
  - Reduction by additional speedup
  - Analysis to determine the required speedup for a given P

Thank you for your attention.