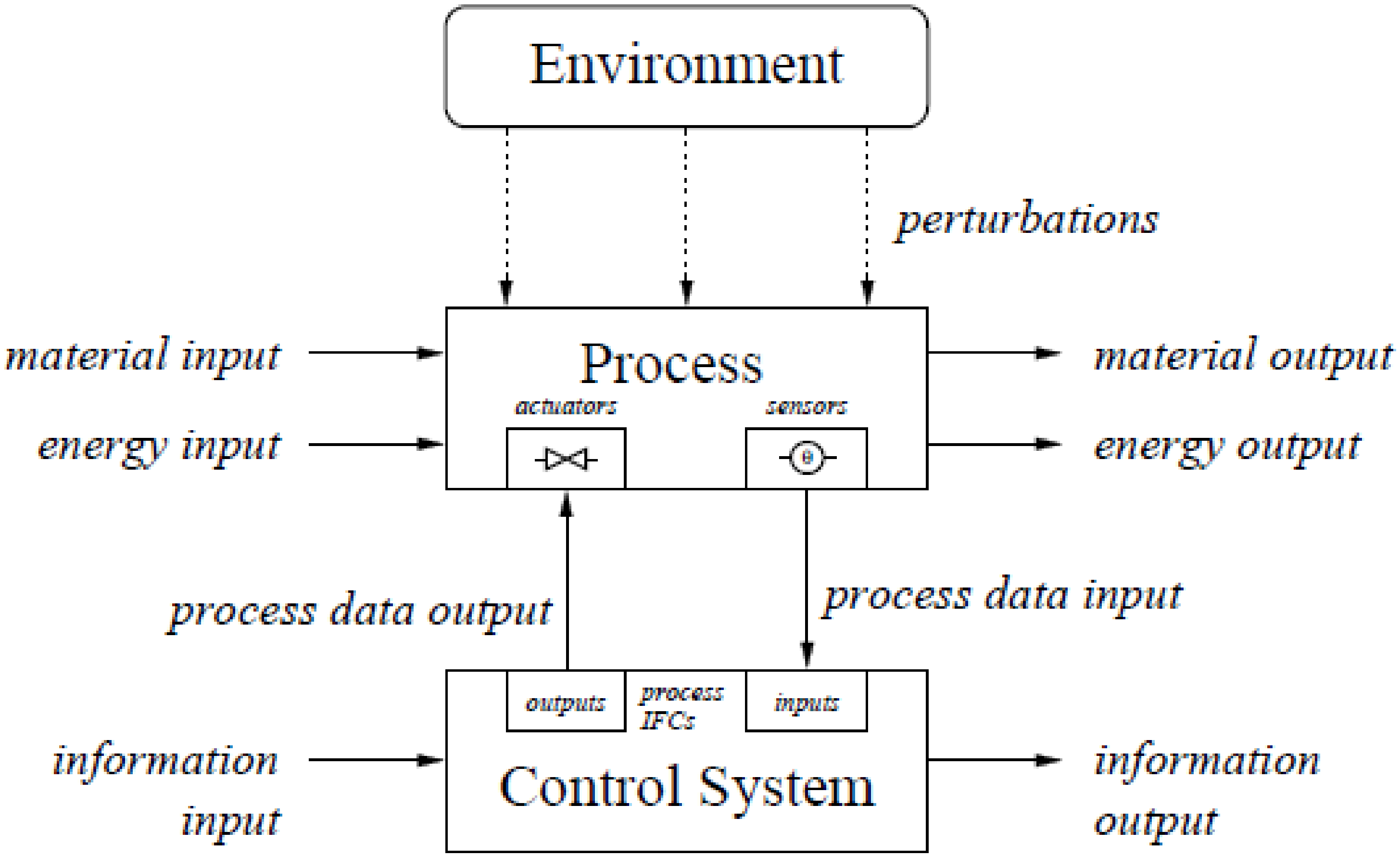
FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Real-Time Scheduling

Motivation

Ulrich Schmid
s@ecs.tuwien.ac.at

Embedded Control Systems (I)



Embedded Control Systems (II)

- Intelligent embedded control systems allow to increase
  - process efficiency
  - process operation time
- Example control system tasks:
  - Feedback control
  - Diagnosis and optimization
- Challenges:
  - Real-time control [closed loop]
  - Critical processes
  - Spatial distribution

Real-Time Computing (I)

- Central problem: Real-time scheduling of
  - tasks on processors
  - messages on communication channels
  - occupancy of mutual exclusive resources
- Example algorithms:
  - Rate Monotonic Scheduling (RM)
  - Earliest Deadline First (EDF)
- Goals:
  - Guaranteed response times
  - Good utilization of resources

Task Characteristics (n Tasks $\tau_1, \ldots, \tau_n$)

- Task τ_i execution properties:
  - Worst-case execution time C_i (WCET)
  - Criticality level
  - Preemptive / non-preemptive execution
  - Task can / cannot be suspended during execution

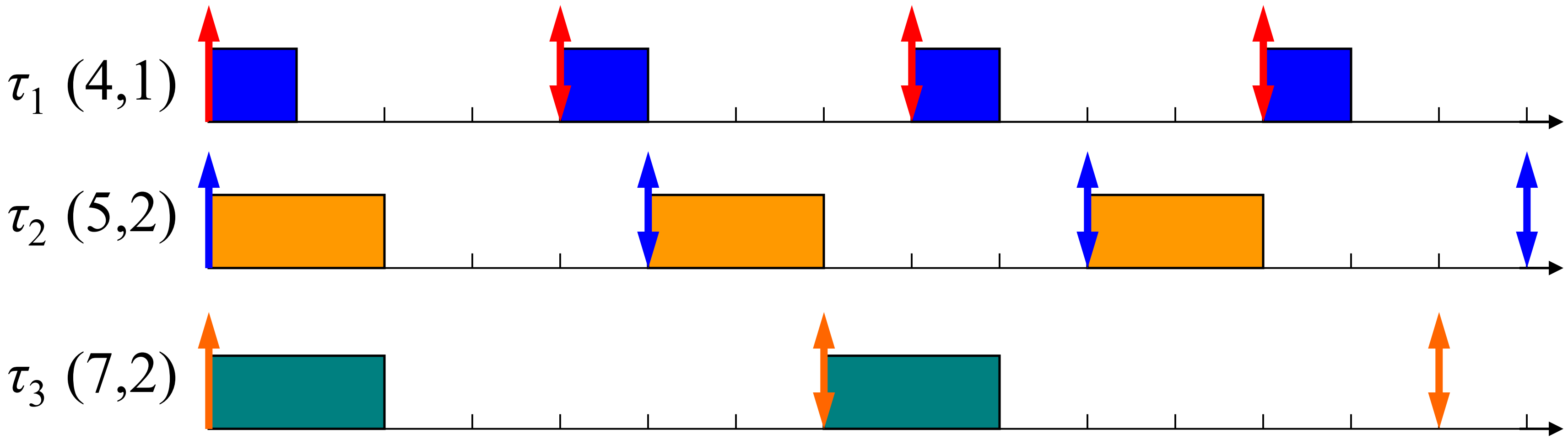
- Task τ_i completion constraints:
  - Relative deadline D_i:
    - Hard: Completion by the deadline mandatory
    - Firm: Completion by the deadline or no execution
    - Soft: Best-effort completion
  - Jitter: Completion within some time interval

- Task τ_i release constraints:
  - Periodic tasks: Released periodically with period T_i
  - Sporadic tasks: Minimum inter-release time T_i
  - Aperiodic tasks: No constraints

- Task τ_i dependencies:
  - Precedence relations among tasks
  - Resource sharing during task execution (shared data, communication links, …)

Schedulability

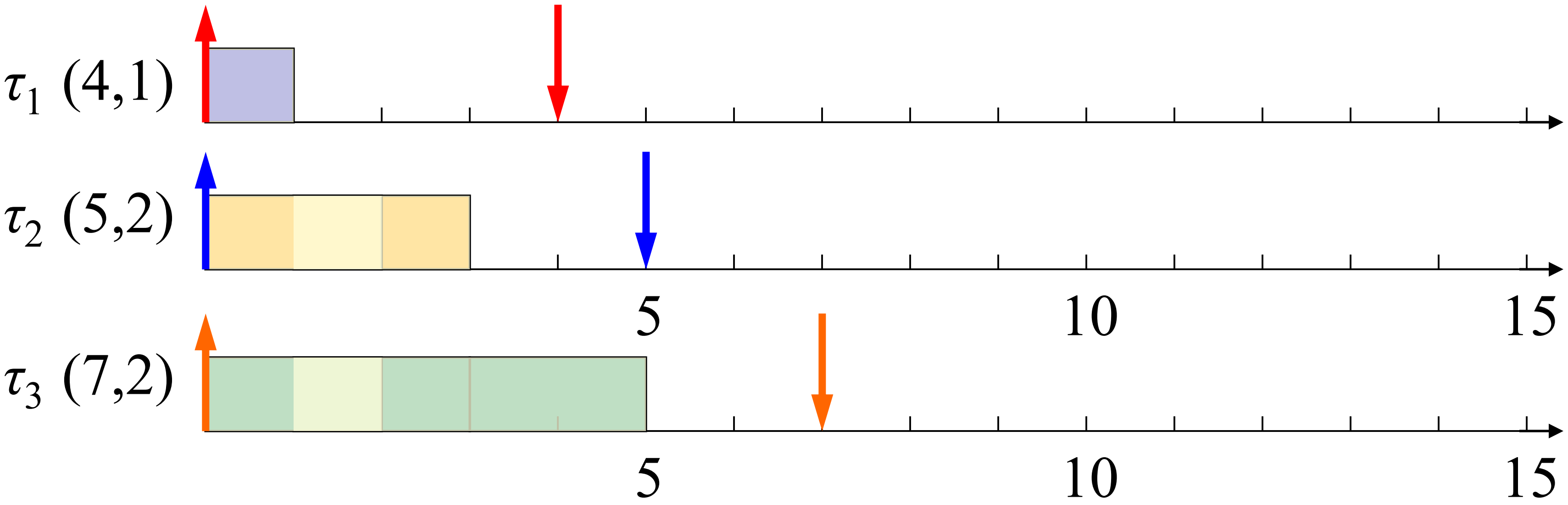
- Property indicating whether a real-time system (a set of real-time tasks) can meet all deadlines
- Example: n=3 periodic tasks $\tau_i\,(T_i, C_i)$ with $D_i = T_i$, $1 \leq i \leq 3$



Can we schedule this feasibly on a single processor ?

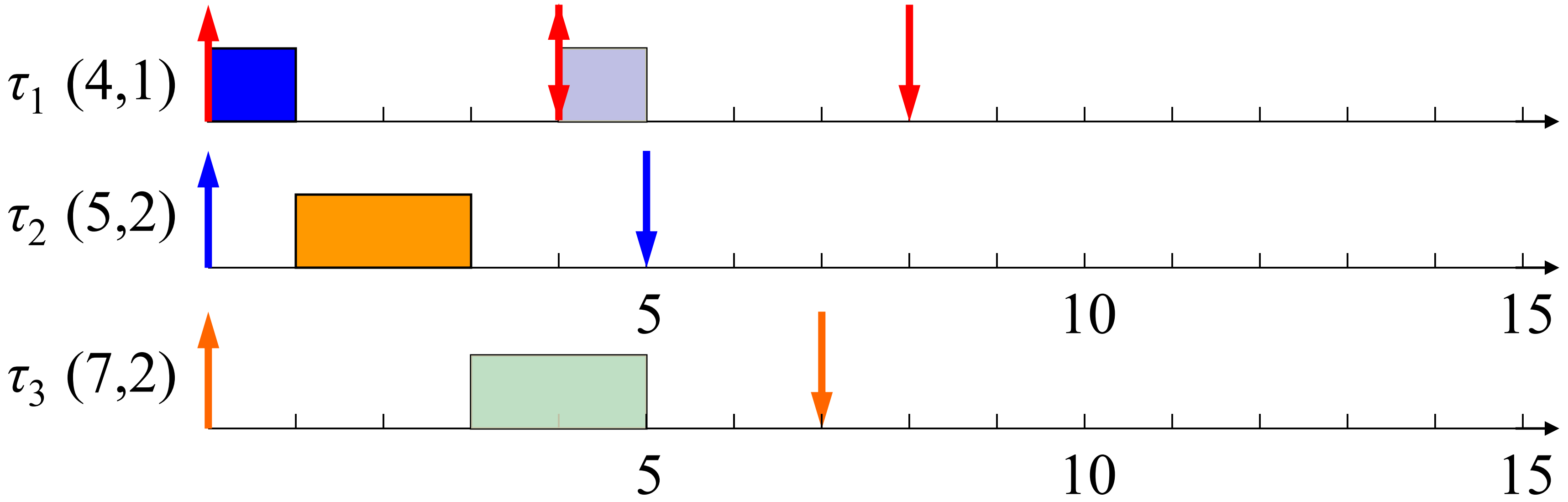
RM (Rate Monotonic) Scheduling

- SPS with priority assigned according to period:
  - A task with a shorter period has a higher priority
  - Always execute pending job with the shortest period
- Optimal static-priority scheduling algorithm



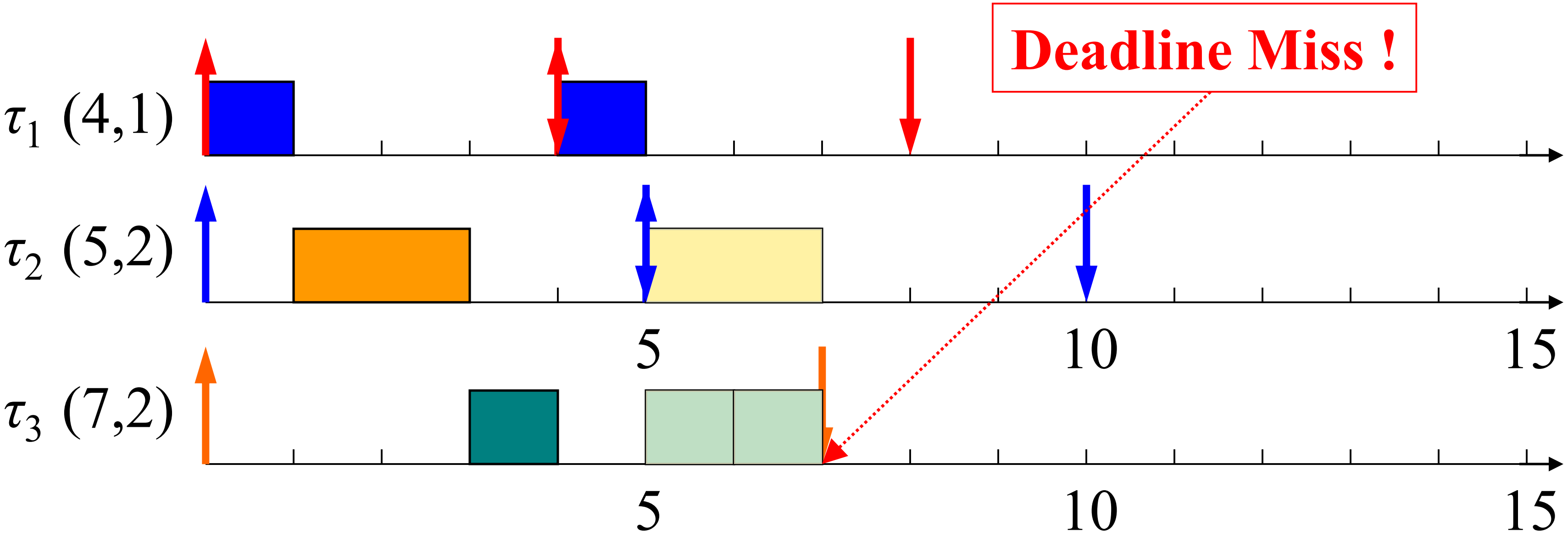
RM (Rate Monotonic) Scheduling

- SPS with priority assigned according to period:
  - A task with a shorter period has a higher priority
  - Always execute pending job with the shortest period
- Optimal static-priority scheduling algorithm



RM (Rate Monotonic) Scheduling

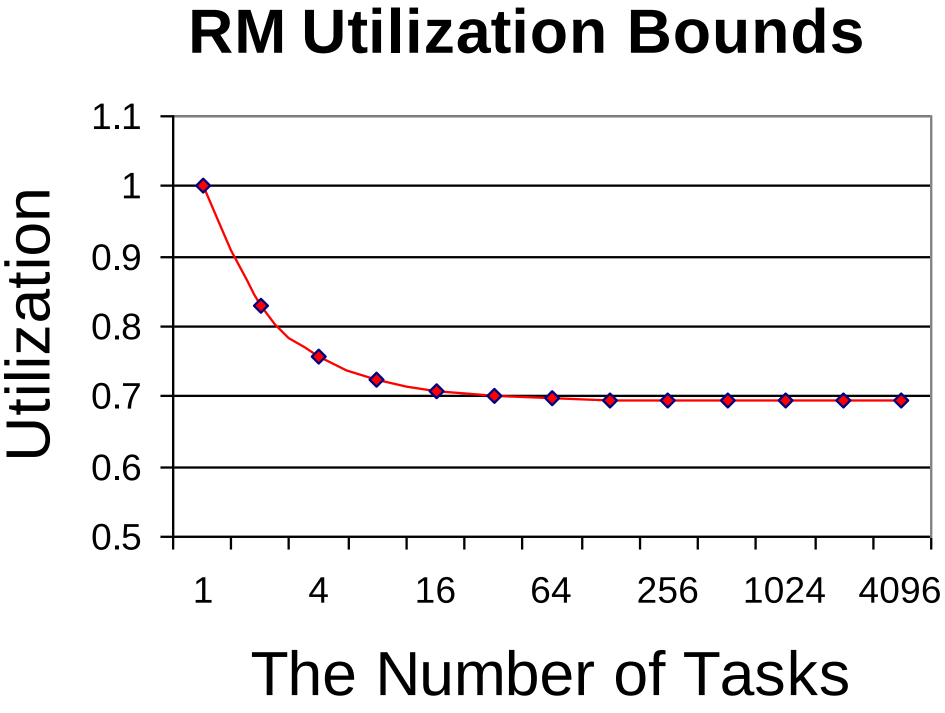
- SPS with priority assigned according to period:
  - A task with a shorter period has a higher priority
  - Always execute pending job with the shortest period
- Optimal static-priority scheduling algorithm



RM – Utilization Bound [LL73]

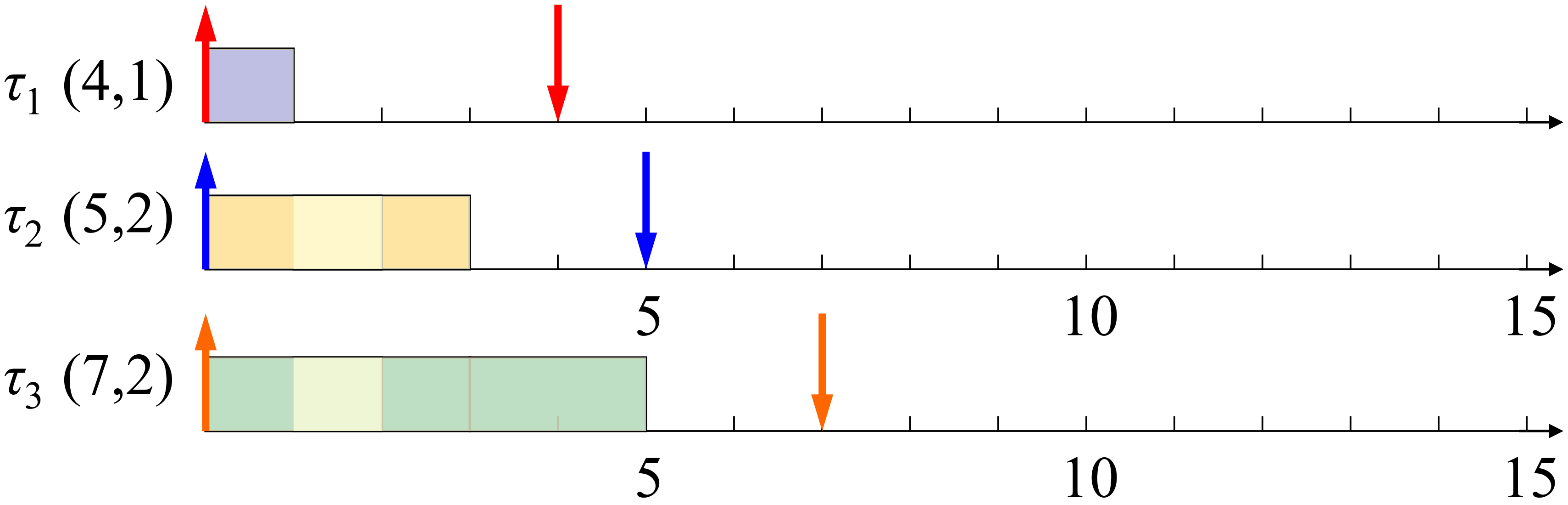
- A set of periodic/sporadic tasks is schedulable under RM if

$$\sum C_i/T_i \leq n\,(2^{1/n}-1)$$



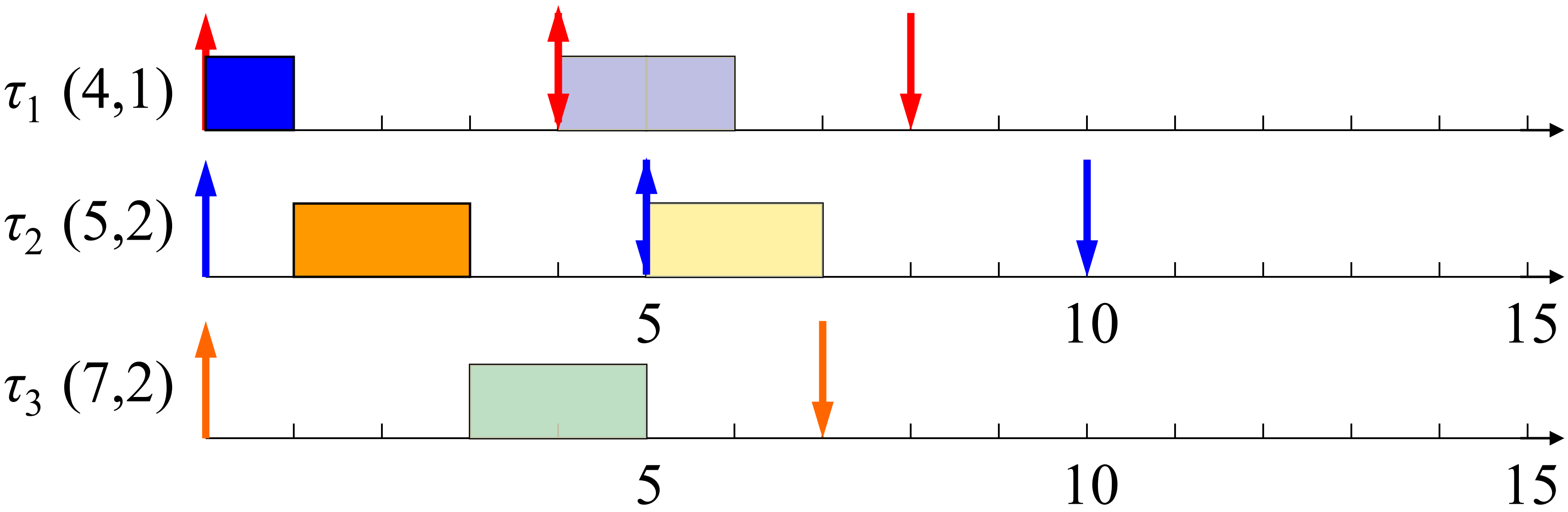
EDF (Earliest Deadline First) Scheduling

- Dynamic priority algorithm:
  - A task with a shorter deadline has a higher priority
  - Always executes pending task with the earliest deadline
- Optimal algorithm (except under overload)



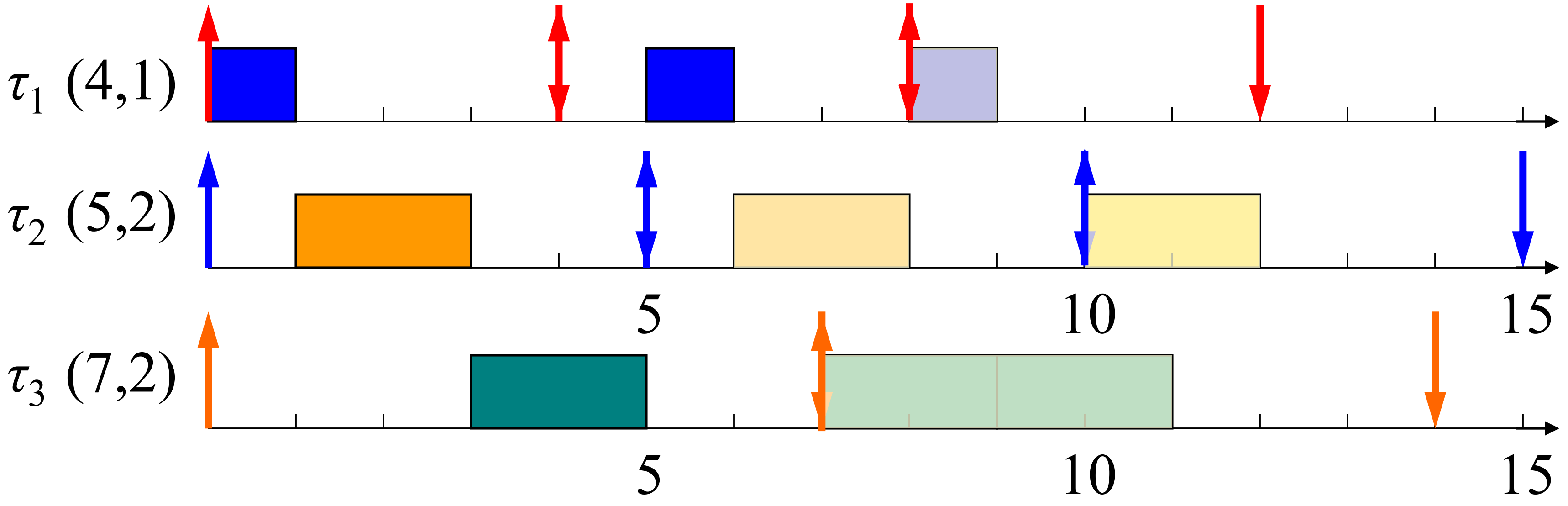
EDF (Earliest Deadline First) Scheduling

- Dynamic priority algorithm:
  - A task with a shorter deadline has a higher priority
  - Always executes pending task with the earliest deadline
- Optimal algorithm (except under overload)



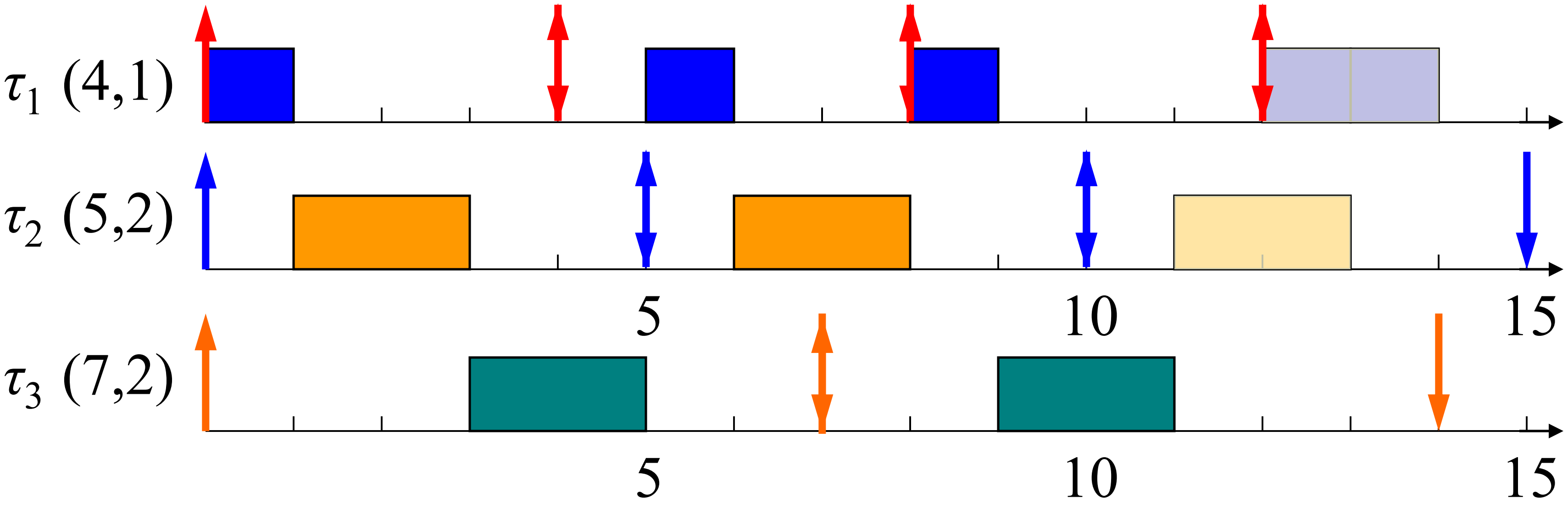
EDF (Earliest Deadline First) Scheduling

- Dynamic priority algorithm:
  - A task with a shorter deadline has a higher priority
  - Always executes pending task with the earliest deadline
- Optimal algorithm (except under overload)



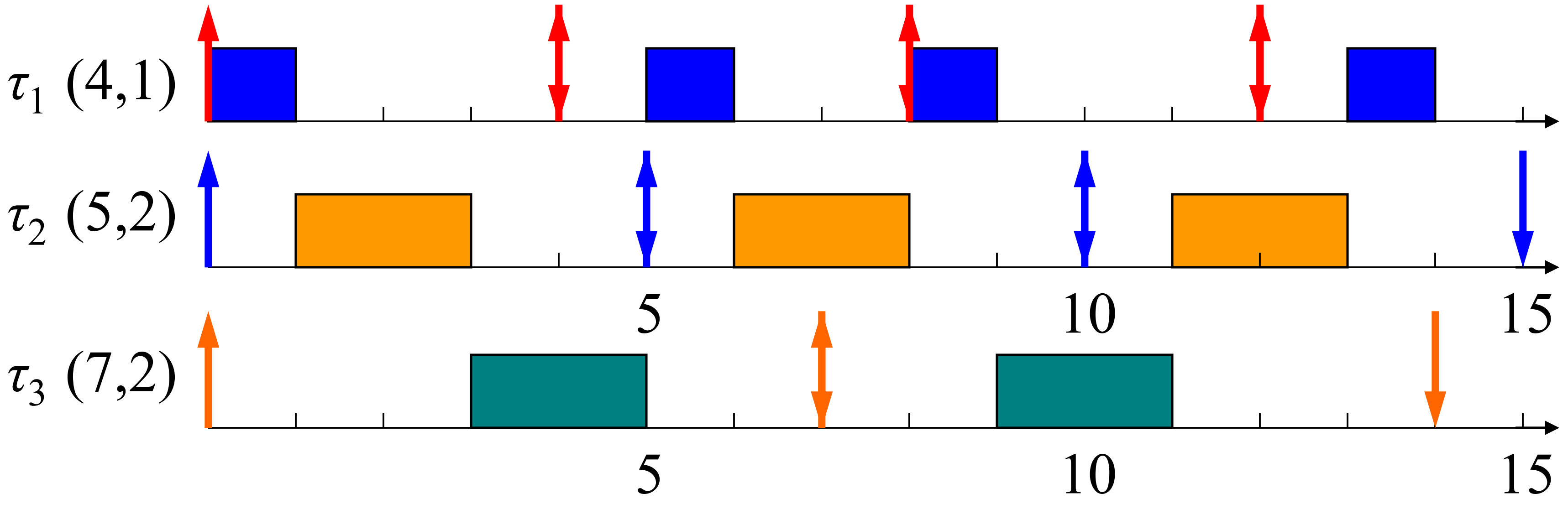
EDF (Earliest Deadline First) Scheduling

- Dynamic priority algorithm:
  - A task with a shorter deadline has a higher priority
  - Always executes pending task with the earliest deadline
- Optimal algorithm (except under overload)



EDF (Earliest Deadline First) Scheduling

- Optimal scheduling algorithm
  - if there is a feasible schedule for a set of real-time tasks, EDF can schedule it.



EDF – Utilization Bound [LL73]

- A set of periodic/sporadic tasks with $C_i = T_i$ is schedulable under EDF if and only if

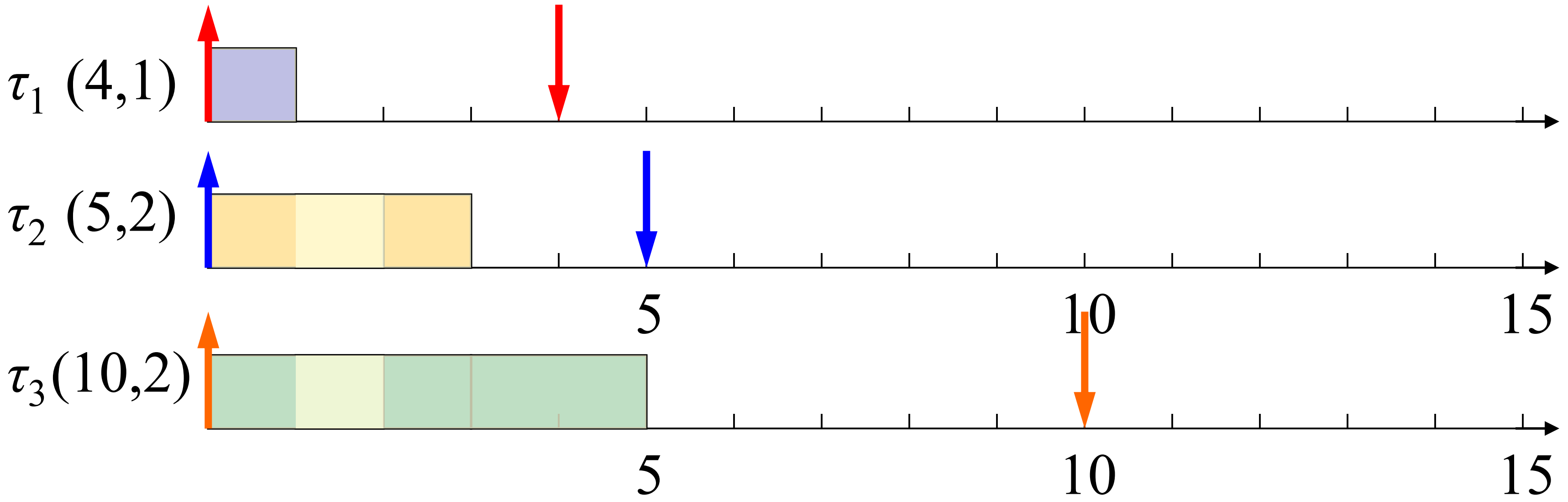
$$\sum C_i/T_i \leq 1$$

- Note: Tasks with
  - critical sections
  - precedence constraints

  can also be handled by means of EDF (using modified deadlines).

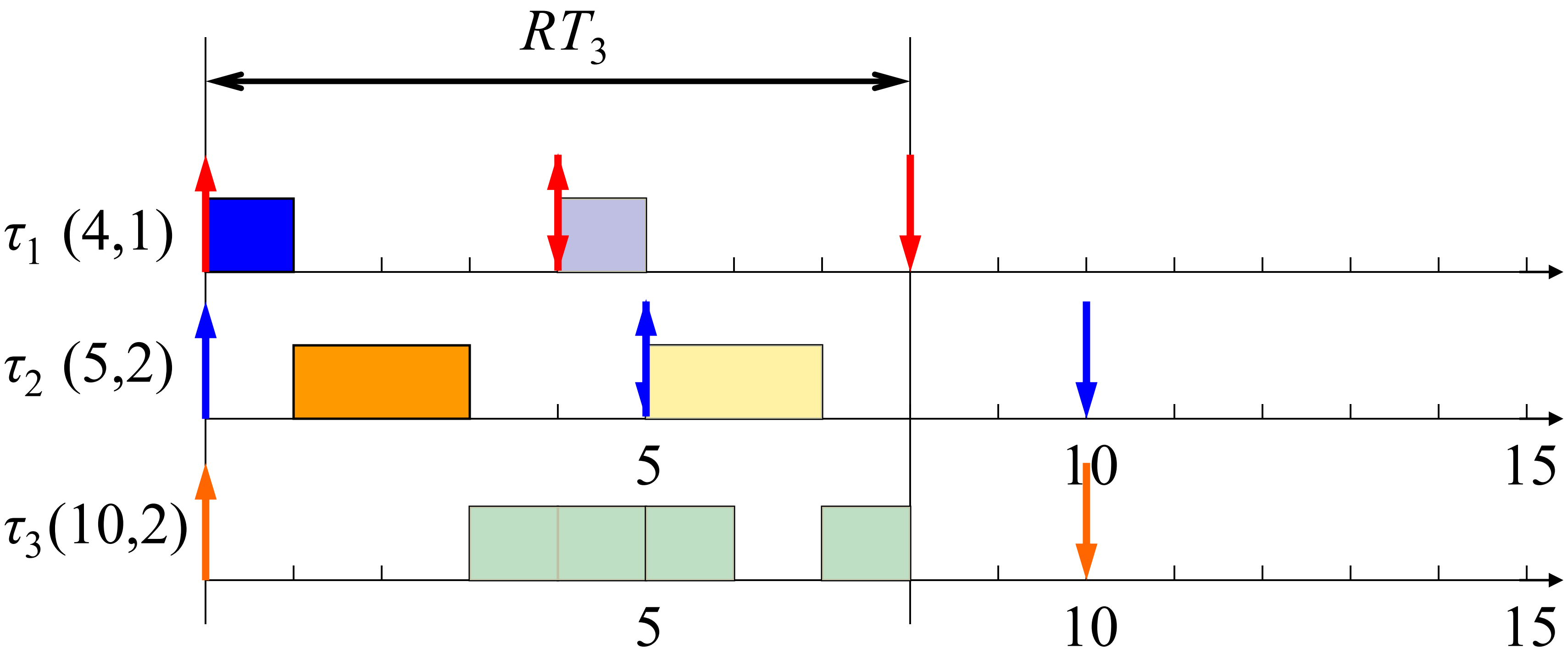
Response Time Analysis

- Response time RT_i
  - Duration from release time to finishing time of task τ_i



Response Time Analysis

- Response time RT_i
  - Duration from release time to finishing time of task τ_i



Response Time Analysis

- Response Time RT_i for RM [ABRT93]

$$RT_i = C_i + \sum_{\tau_k \in HP(\tau_i)} \left\lceil \frac{RT_i}{T_k} \right\rceil \cdot C_k$$

HP(τ_i) : a set of higher-priority tasks than τ_i

- A similar (more complex) formula exists for EDF

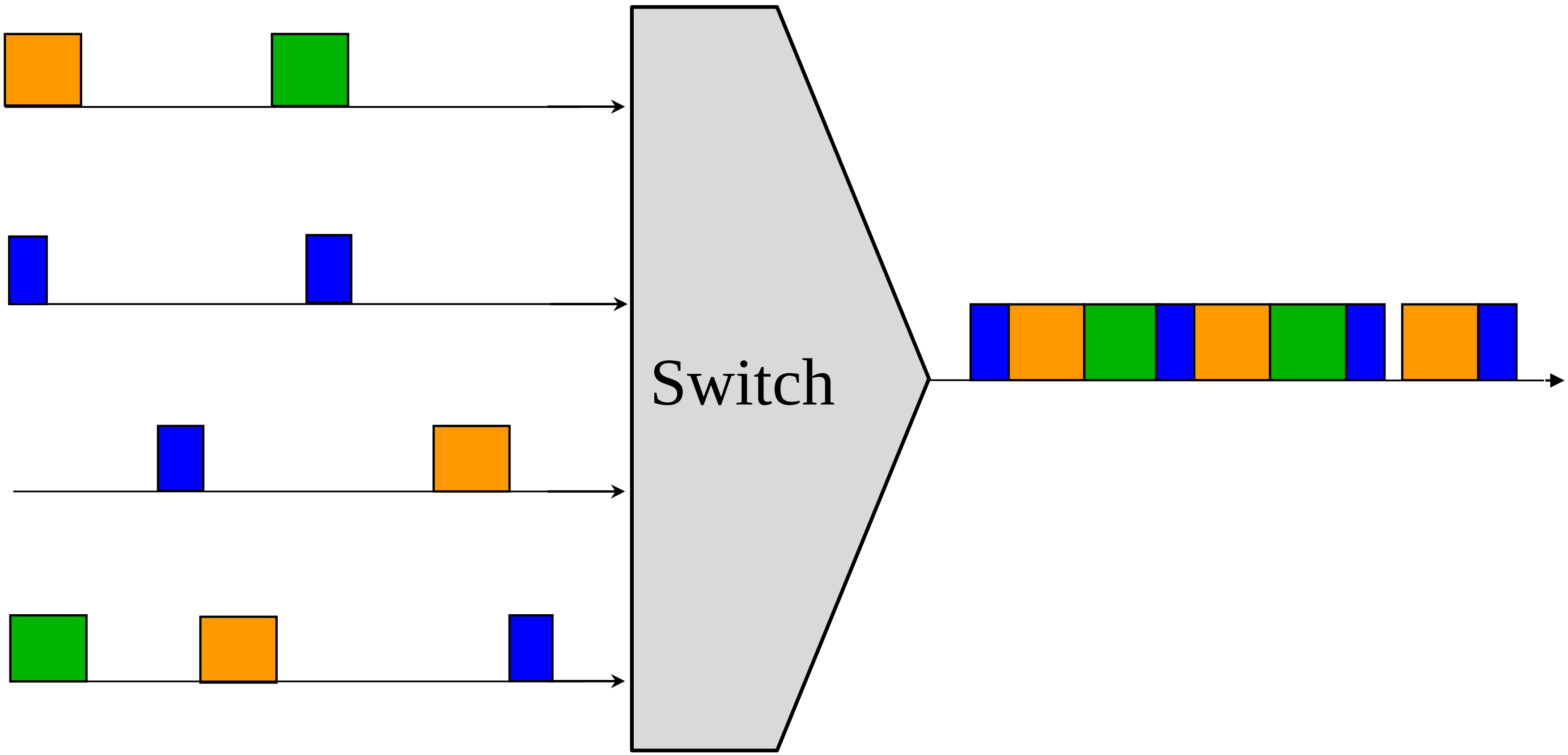
Distributed Real-Time Systems?

- Response time analysis can be extended to incorporate release jitter:
  - Additional (but non-cumulative) delay between release of a task and the time it can be scheduled first
  - Can be used to model variability of message delivery delays at receiving processor, as caused by
    - sending processor CPU scheduling
    - message scheduling in the network
- **Holistic Schedulability Analysis**
  - Static priority scheduling: [TC94]
  - EDF: [Spu96]

Real-Time Scheduling under Overload

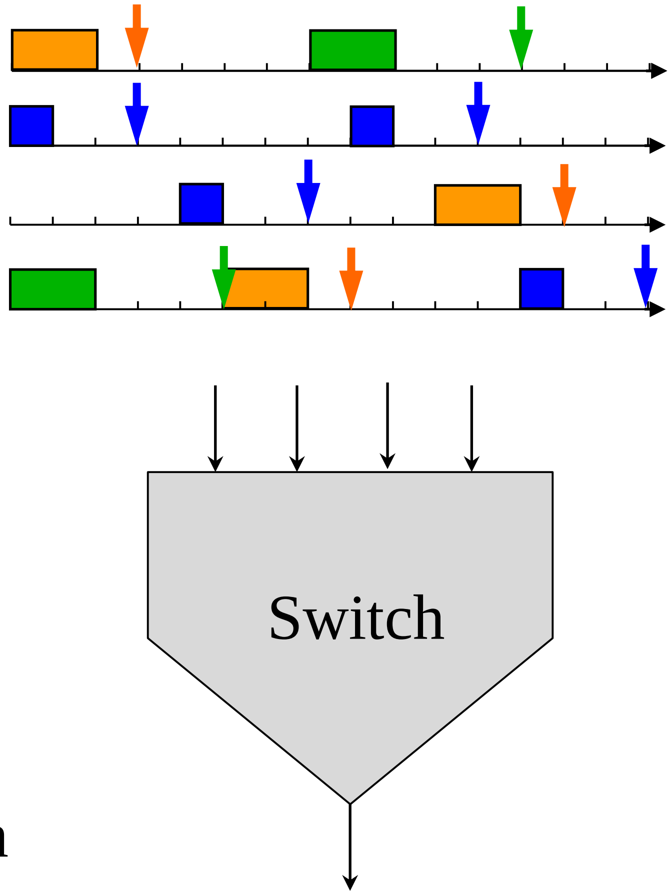
Example

Packet scheduling in network switch/concentrator:

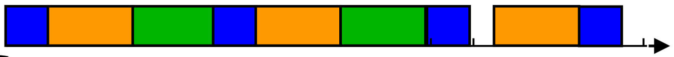


Real-Time Scheduling Problem

- Switch may also be overloaded:
  - Multiple incoming data streams
  - Single outgoing data stream, may drop packets



- n different classes of packets $\tau_1, \ldots, \tau_n$, with τ_i characterized by:
  - Packet duration C_i
  - Utility value V_i
  - Firm relative deadline D_i,after which packet is useless

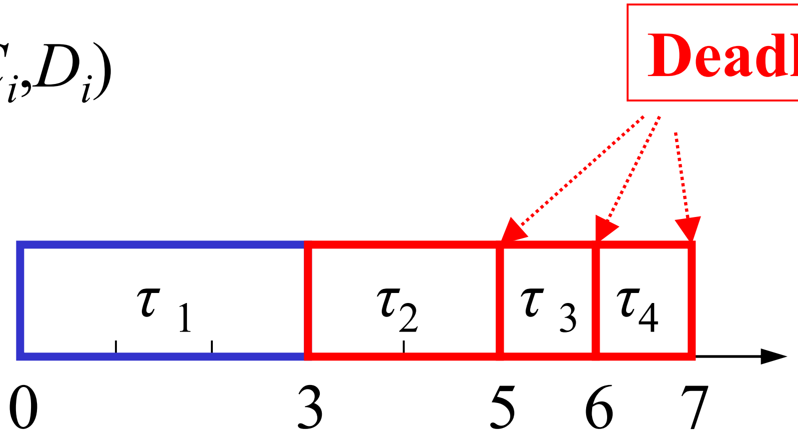


- Packets can be split into fragments (in unit-time slots)

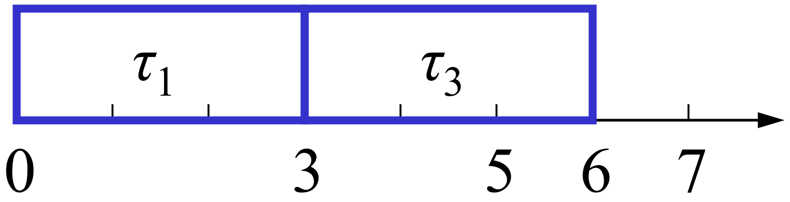
EDF under Overload ?

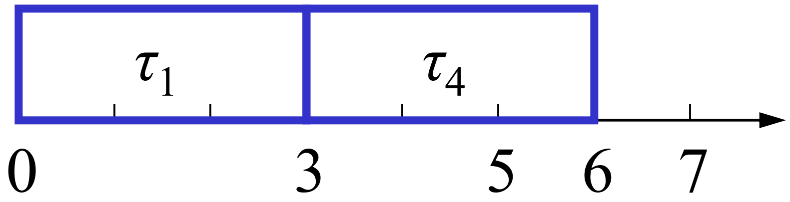
- Domino effect during overload conditions
  - Example: $\tau_1(3,4)$, $\tau_2(3,5)$, $\tau_3(3,6)$, $\tau_4(3,7)$ all pending at $t = 0$

$\tau_i(C_i,D_i)$

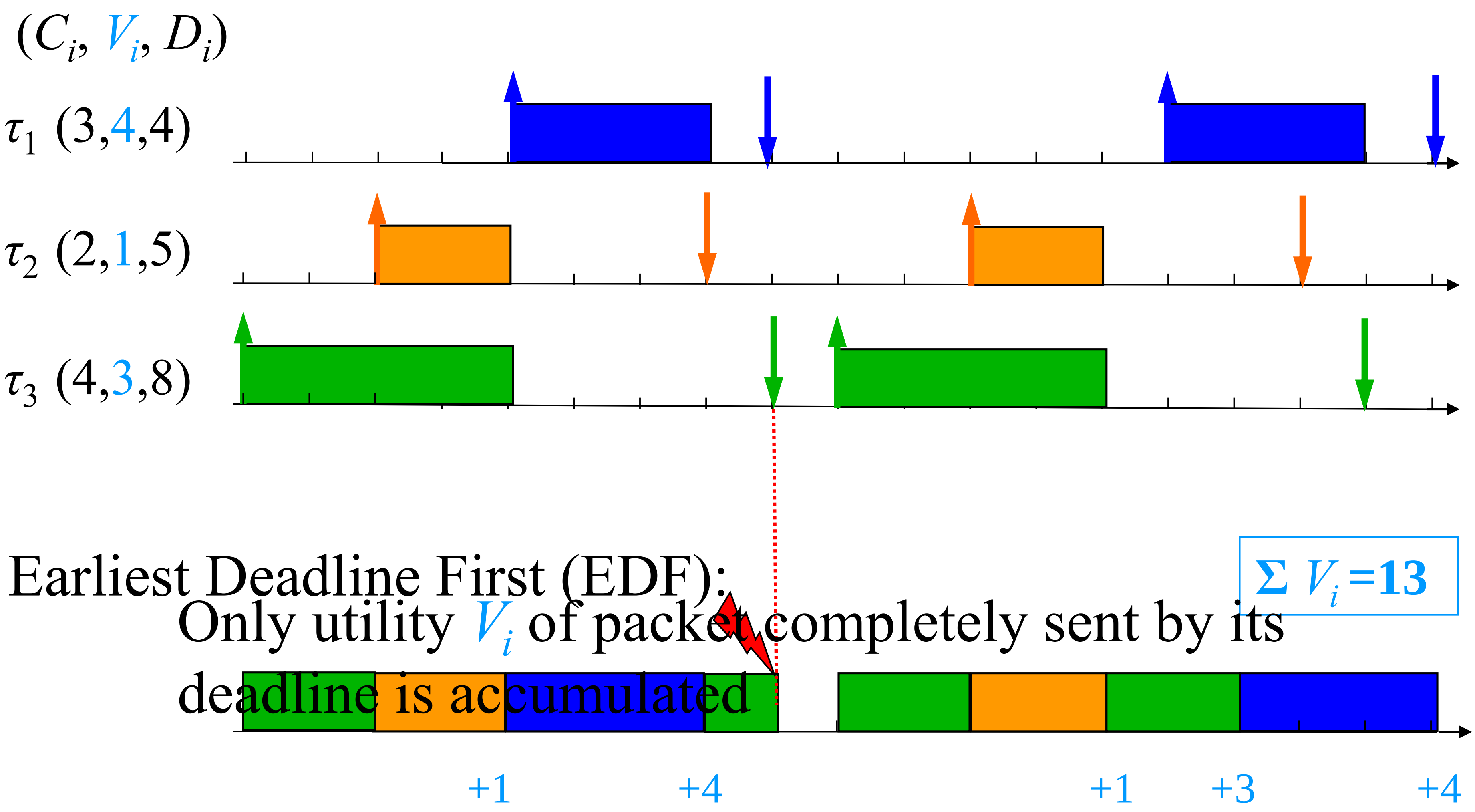


Better schedules :





Goal: Maximize Cumulated Utility (I)

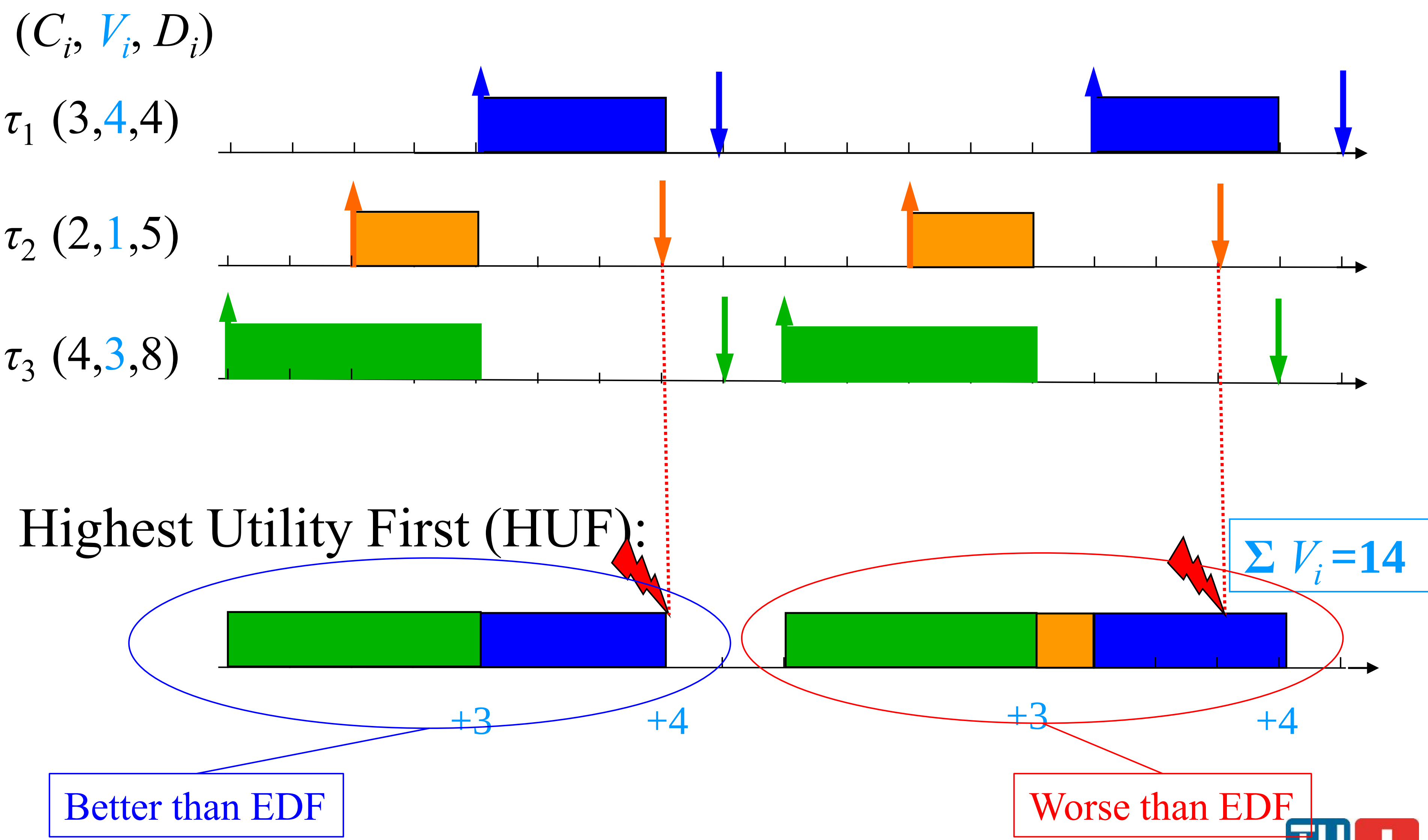


Earliest Deadline First (EDF):

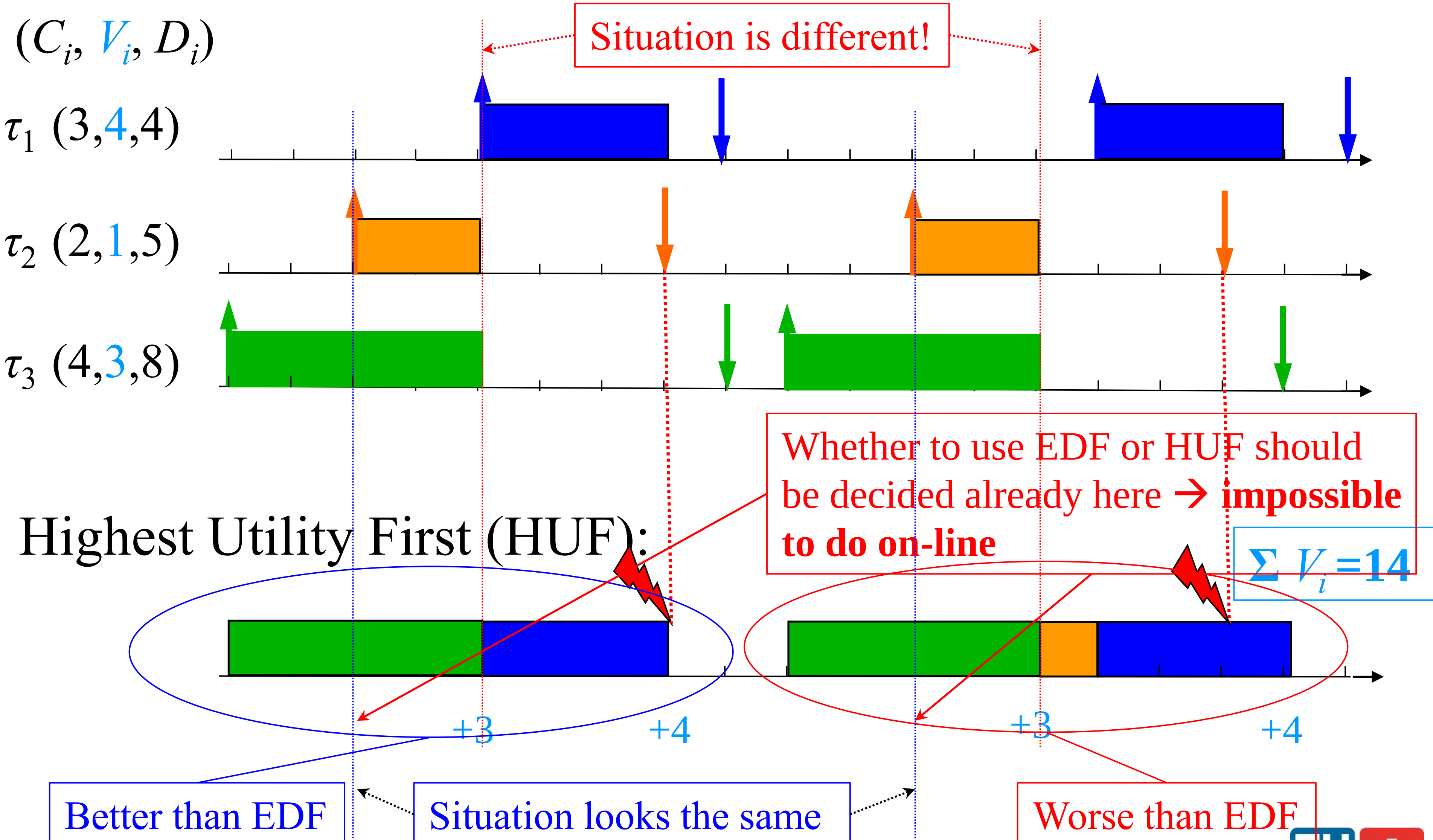
Only utility V_i of packet completely sent by its deadline is accumulated

$\Sigma V_i = 13$

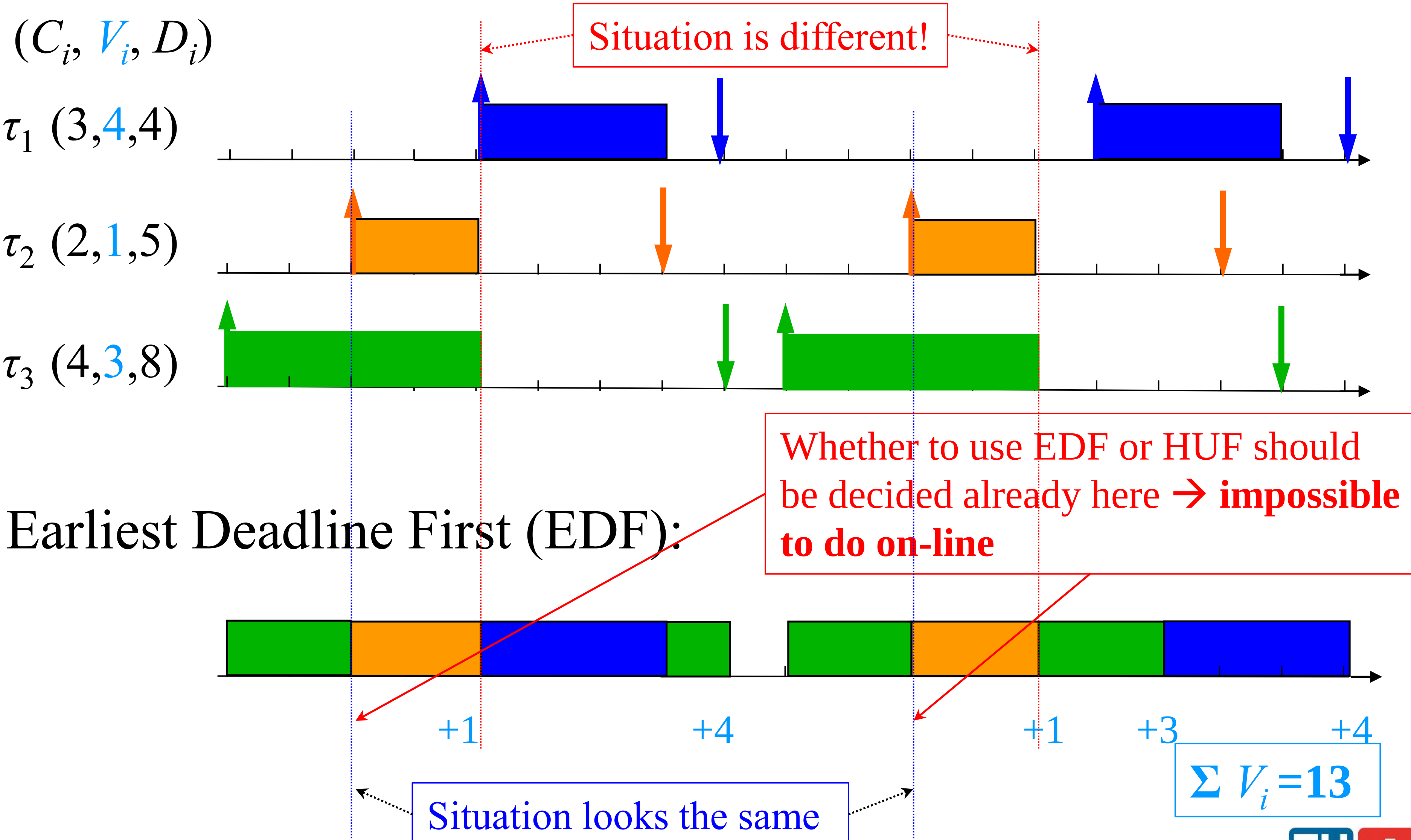
Goal: Maximize Cumulated Utility (II)



Goal: Maximize Cumulated Utility (III)



Goal: Maximize Cumulated Utility (IV)



RT Scheduling Performance ?

- Infeasible:
  - There cannot be an optimal deterministic on-line algorithm (i.e., one that does not know the future)
  - Only a hypothetical clairvoyant algorithm $\mathfrak{C}$ (i.e., one that does know the future) could maximize the cumulated utility
- (Principally) feasible:
  - Computing minimal cumulated utility $\inf_\sigma \Sigma V_i^{\mathfrak{A}}$ for a given on-line algorithm $\mathfrak{A}$ (over all possible input scenarios σ)
  - Compute competitive ratio $\varphi_{\mathfrak{A}}$ of a given on-line algorithm $\mathfrak{A}$

$$\varphi_{\mathfrak{A}} = \inf_\sigma (\Sigma V_i^{\mathfrak{A}} / \Sigma V_i^{\mathfrak{C}})$$

  - Compute competitive ratio φ of optimal on-line algorithm

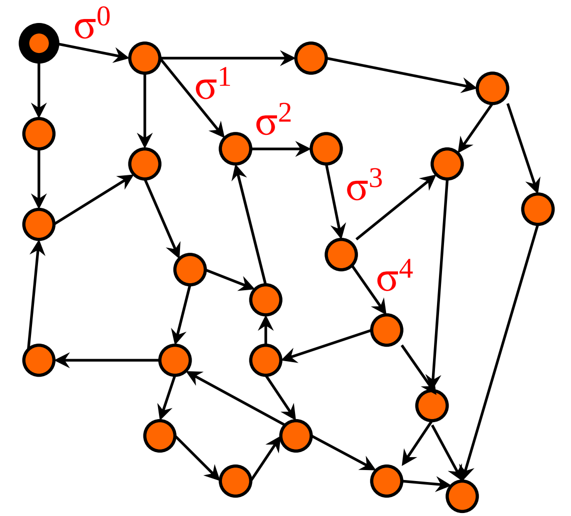
Game Modeling of RT Scheduling

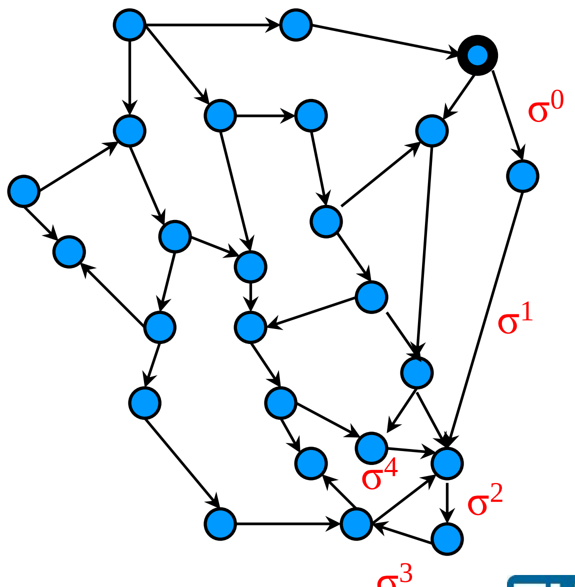
- Player 1 (on-line algorithm):
  - Determines task to be executed in current slot
  - Strategy $\pi \in \Pi^M$: deterministic and memoryless
  - Reward: V_i if task τ_i is completed by its deadline in current slot, or 0 otherwise

- Player 2 (adversary):
  - Determines set of (new) task instances arriving in current slot
  - Strategy $\sigma \in \Sigma$: deterministic but not memoryless

Competitive Analysis Principle [CKS13]

- Competitive analysis reduced to partial observation game
- Player 1: On-line algorithm
  - Performs transitions of on-line algorithm $\mathfrak{A}$
  - Cannot observe moves of $\mathfrak{C}$
- Player 2: Adversary
  - Determines task arrivals per slot ($\sigma^0\sigma^1\sigma^2\ldots$)
  - Performs transitions of clairvoyant algorithm $\mathfrak{C}$





Extensions [CPKS14]

- Compute competitive ratio for a given algorithm, by finding multi-cycles in multi-objective graphs
- Additional constraints on adversary
  - Safety objectives: Certain states never reached in play
    - Restrictions on input task sequences (sporadicity etc.)
    - Energy-constrained workload
  - Büchi objectives: Certain states reached infinitely often
    - Ensure finite periods of overloads
    - Analyzing semi-online algorithms (certain tasks eventually executed)
  - Multi-dimensional objectives: Multiple mean-payoff
    - Constraining average load
    - Additional objectives for scheduling algorithms, like minimizing energy consumption

The End
(not quite …)

References

- [ABRT93] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling, Software Engineering Journal, September 1993.
- [But05] Giorgio C. Buttazzo. Rate monotonic vs. EDF: Judgement Day. Journal Real-Time Systems 29(1), January 2005.
- [CKS13] Krishnendu Chatterjee, Alexander Kößler, Ulrich Schmid: Automated Analysis of Real-Time Scheduling using Graph Games. Proceedings of the 16th ACM international conference on Hybrid Systems: Computation and Control (HSCC '13), 2013.
- [CPKS14]] Krishnendu Chatterjee, Andreas Pavlogiannis, Alexander Kößler, Ulrich Schmid: A Framework for Automated Competitive Analysis of On-line Scheduling of Firm-Deadline Tasks. To appear in Proc. Real-Time Systems Symposium, 2014.
- [LL73] C. L. Liu, James Layland. Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of the ACM 20 (1): 46-61, 1973.
- [SAAC04] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, “Real time scheduling theory: A historical perspective,” *Real-Time Systems Journal, vol. 28, no. 2/3, pp. 101–155, 2004.*
- [Spu96] Marco Spuri: Holistic Analysis for Deadline Scheduled Real-Time Distributed Systems. INRIA Rapport de Recherche 2873, 1996.
- [SSRB98] John A. Stankovic, Marco Spuri, Kriti Ramamritham, Giorgio C. Buttazzo: *Deadline Scheduling for Real-Time Systems*, Kluwer Academic Publishers (now Springer Verlag), 1998, ISBN 0-7923-8269-2