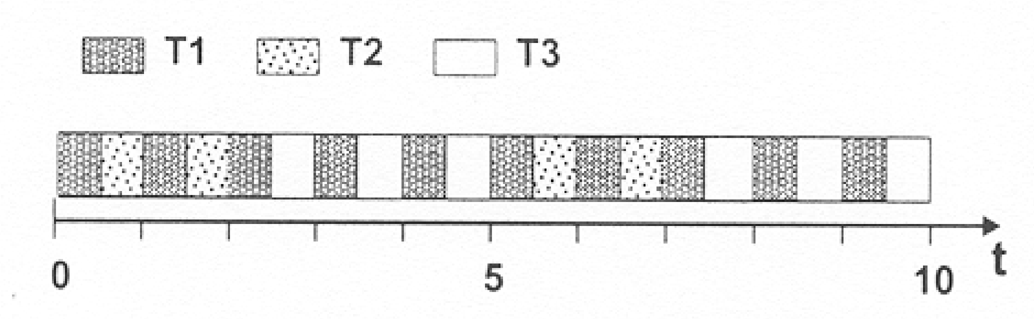
Real-Time (Paradigms) (28)

Example of a rate-monotic schedule



Heuristics for dealing with sporadic tasks:

- modeling them as *pseudo-periodic* by defining $T_R = T_{Rmin}$
  Main drawback: most of the periods are empty --> very low processor utilization
- Adding a periodic server task with high priority to serve the pending sporadic requests (*sporadic server)*

Note:

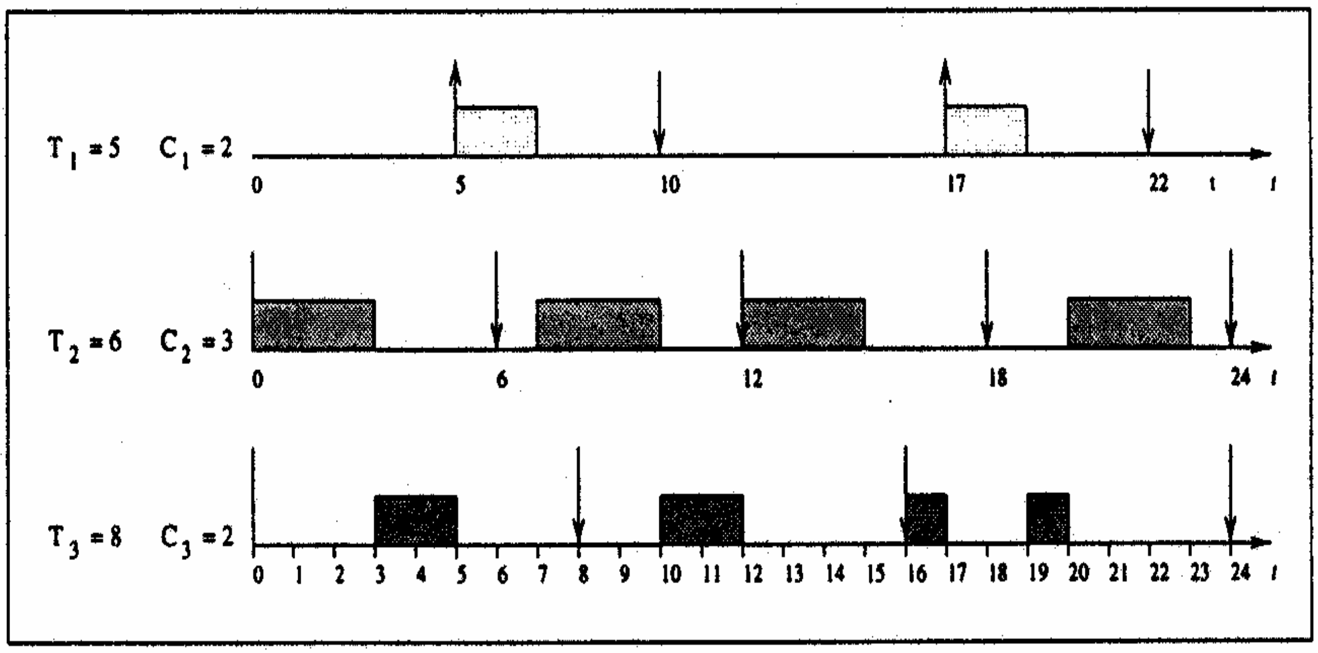
Since RM is optimal among all static assignments, an improvement of the bound for U can be achieved only by using dynamic scheduling algorithms.

Real-Time (Paradigms) (29)

Earliest Deadline First Scheduling Algorithm (EDF)

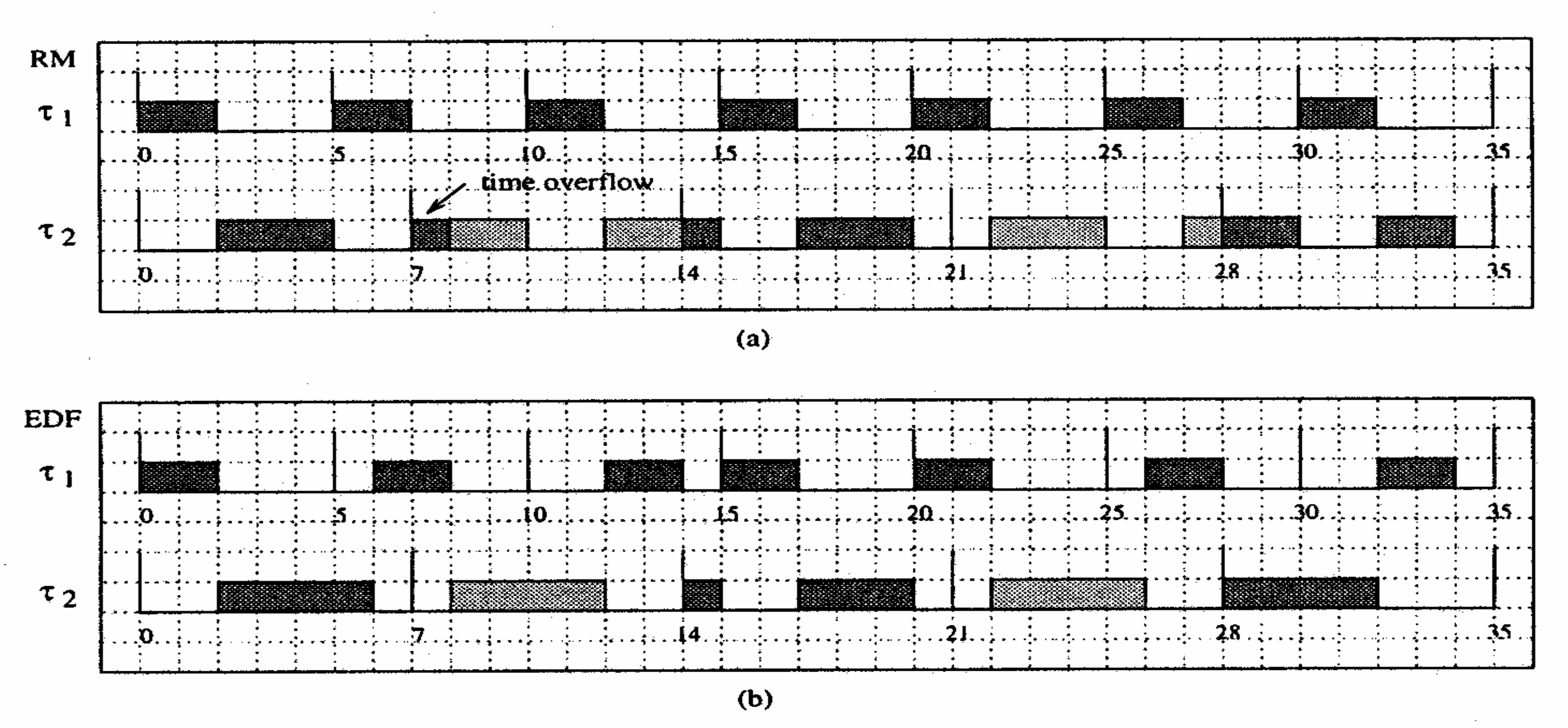
- designed for static and dynamic scheduling of independent periodic and sporadic tasks
- it is preemptive and based on dynamic priorities
- the task´s priority is inversely related to its absolute deadline ---> tasks with shorter deadlines have higher priorities
- It is optimal among all priority-based algorithms
- If used for static scheduling, U<= 1 is a sufficient condition for the schedulability test

Example of an earliest deadline first - schedule



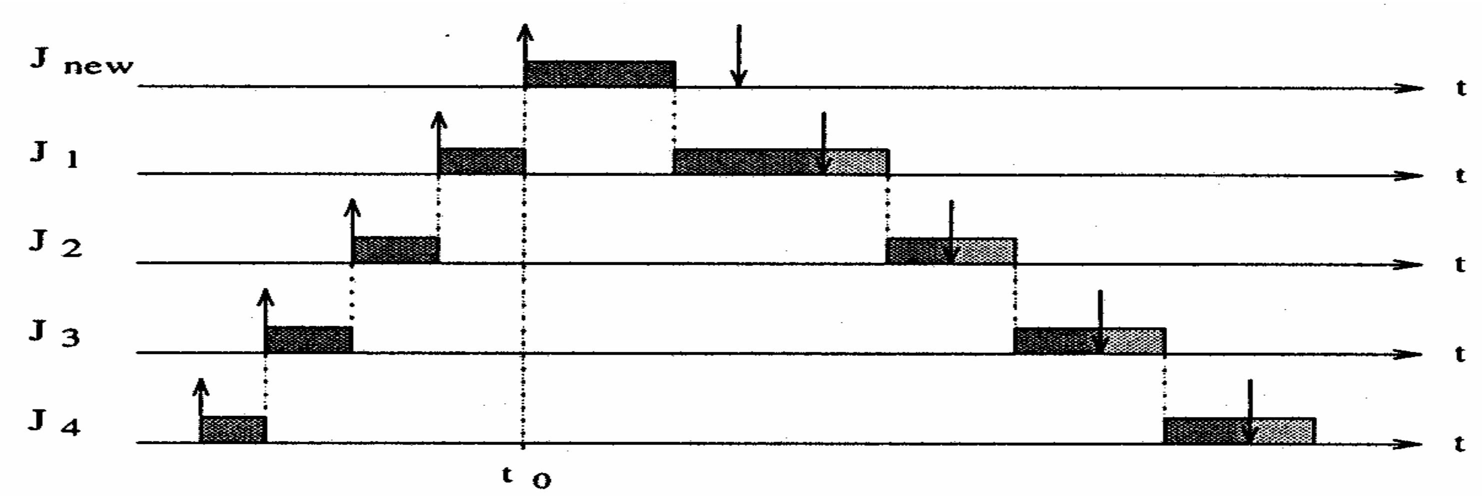
Real-Time (Paradigms) (30)

Comparison EDF <---> RM by means of an example



Real-Time (Paradigms) (31)

Example of the domino effect



The last example constitutes a best-effort approach
 ---> no feasibility checking is done
---> no individual task deadline can be guaranteed
---> provides no predictability

Classification of scheduling policies

Several scheduling policies exist, depending on whether

- a system performs schedulability tests at all (if not, only best-effort (no real-time) approach)
- if so, when it is done (on-line versus off-line)
- what type of schedule is produced as a result of the analysis (priority list or calendar)
- whether robustness or fault-tolerance is considered

Estimating Task Execution Times (1)

The Execution Time (ET) of a task depends on the characteristics of

- the hardware architecture
- the operating system
- the programming language

1.1 DMA (Direct Memory Access)

DMA is a technique used by many peripheral devices to directly transfer data between the device and main memory

Purpose: to relieve the CPU of the task of controlling the I/O transfer

---> CPU and the respective I/O device share the memory bus

---> need for conflict resolution

Most common method: *Cycle stealing*

Idea: in case of a conflict, the I/O device always gets priority

---> CPU ET of a task cannot be precisely determined

Possible solution: *time-slice method*

Idea: each memory cycle is split into two adjacent time slots

Estimating Task Execution Times (2)

1.2 Cache

The cache is a fast memory inserted as a buffer between the CPU and the main memory (RAM)

Purpose: to reduce the bad effects on the speed of processes´execution stemming from the wide disparity between processor and memory cycle times

Problem: It is difficult to determine the success of a cache access (cache hit versus cache miss) since the cache contents are not easy to predict. Even extensive code analysis does not solve the problem.

Main reason: The existence of conditional branches and/or task preemption's

Solution approaches:
- Worst-case analysis resulting in a disabled cache
- Strategic Memory Allocation for Real Time (SMART)