Real Time Operating Systems

Scheduling

Lecturer:
Prof. William Fornaciari
Politecnico di Milano
fornacia@elet.polimi.it
www.elet.polimi.it/~fornacia

Outline

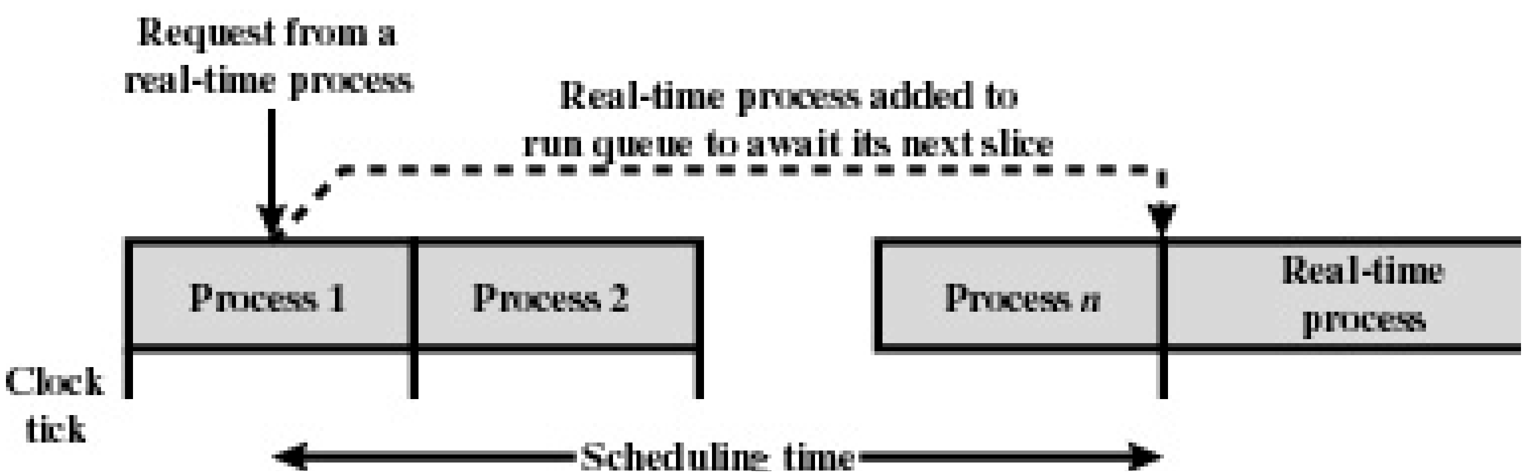
- *Introduction to Real Time Systems*
- *The role of a RTOS*
- *Time constraints*
- The problem of scheduling RT activities
- Classification of scheduling strategies
- Review of RT scheduling approaches
- *VxWork real-time features*
- *Windows CE real-time features*
- *Unix SVR4, Win2000, Linux*

Short-Term Scheduler enables RT

- The heart of a RTOS is the short-term scheduler
  - Fairness and min avg response time are not paramount
  - crucial: all hard-RT tasks must complete (or start) by their deadline and as many as possible soft-RT tasks should also complete (or start) meeting their deadlines
- Most current RTOSs are unable to deal with deadlines
  - they are designed to be as responsive as possible to RT tasks, so that, when deadline approaches, they can be quickly scheduled
  - this approach requires deterministic response time sometimes below milliseconds

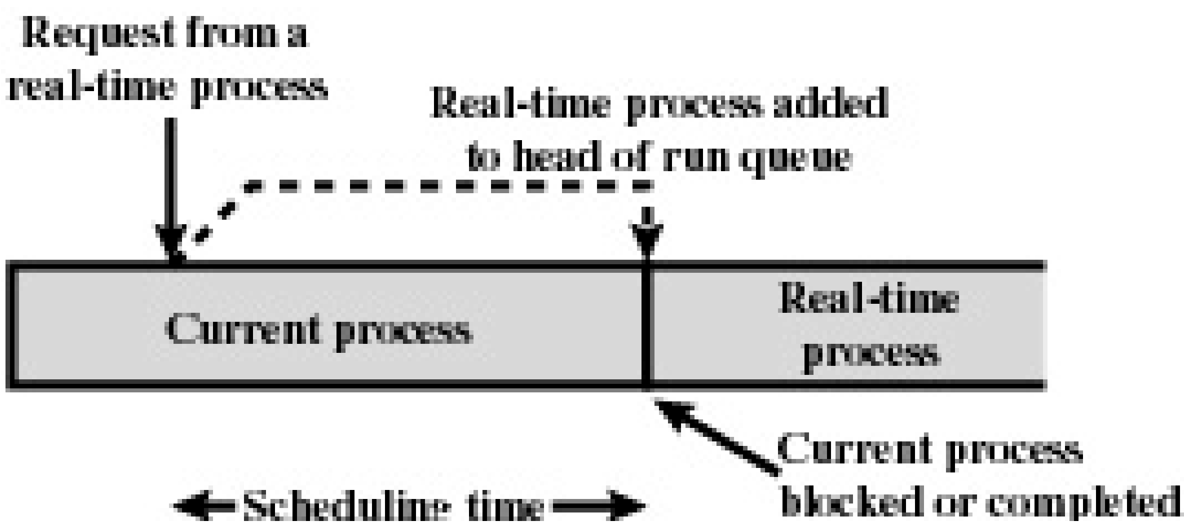
Towards a RT scheduler (1)

- Round Robin preemptive scheduler
  - The RT task is appended to the ready queue to await its next timeslice
  - The delay can be unacceptable for RT applications



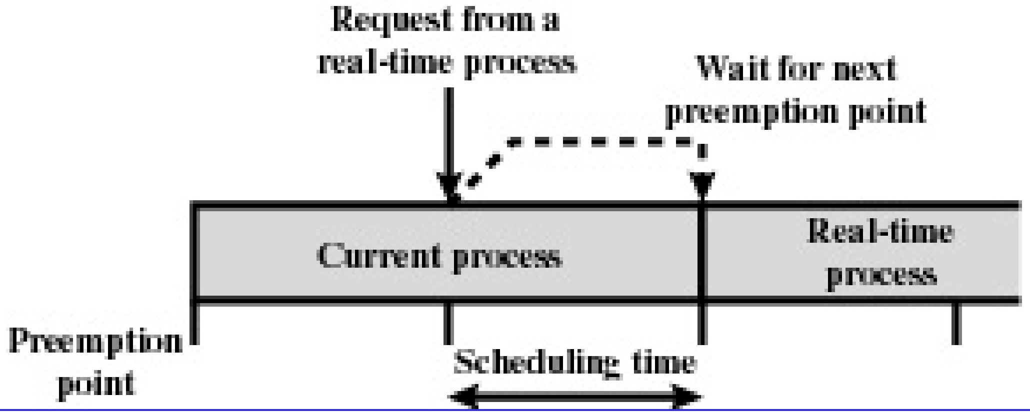
Towards a RT scheduler (2)

- Priority driven nonpreemptive scheduler
  - RT tasks have higher priority
  - A RT task is scheduled when the current P is blocked or runs to completion (even if with low priority)
  - Possible delay of seconds, unacceptable for RT



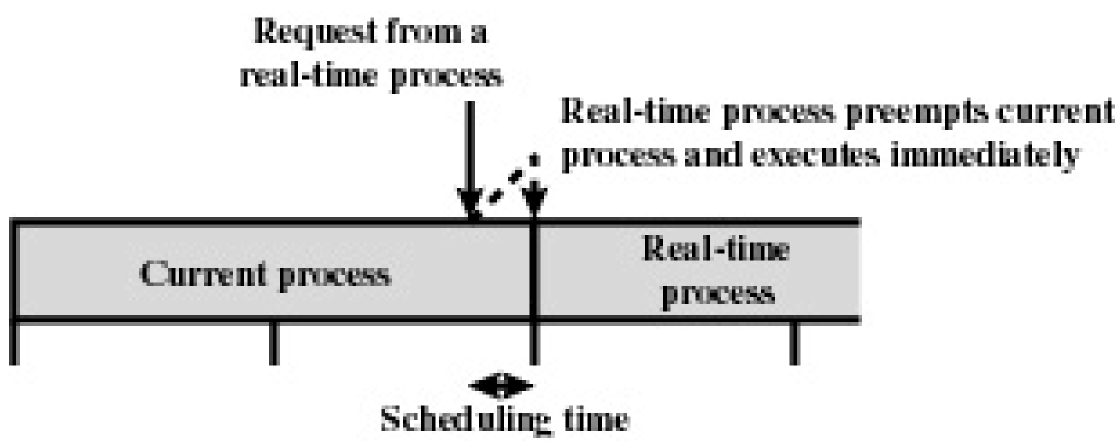
Towards a RT scheduler (3)

- Priority driven preemptive scheduler on preemption points
  - preemption takes place at the end of some regular intervals
  - in those points, the highest priority task is scheduled (including kernel tasks)
  - delays in the order of some ms, adequate for some applications not for more demanding ones



Towards a RT scheduler (4)

- Immediate preemptive scheduling
  - Apart from the case OS is executing a critical region, the service to the interrupt is almost immediate
  - Scheduling delays fall down to 100 µs, or less
  - Good for critical systems



(d) Immediate Preemptive Scheduler

Factors influencing RT scheduling

- The system performs (or not) schedulability analysis
- Static vs dynamic schedulability analysis
- The result of the analysis can be
  - a clear scheduling
  - a strategy (plan) to be followed at run-time for task dispatching
- Classes of algorithms
  - static table-driven
  - static priority-driven preemptive
  - dynamic planning-based
  - dynamic best-effort

Static table driven

- Through a static feasibility analysis of schedule, determines, at run-time, when a task must begin execution
- Applicable to periodic tasks
- Analysis inputs:
  - periodic arrival time
  - execution time
  - periodic ending deadline
  - relative priority of each task
- Predictable but inflexible approach: any change in the requirements of tasks imply the computation of a new schedule
- Example: Earliest-deadline-first

Static priority-driven preemptive

- Uses the traditional priority-driven preemptive scheduler
- Static analysis is performed but no schedule is drawn-up, it is used to assign priorities to task
  - in no RT time-sharing systems, typ priority can change depending on I/O vs CPU bound process nature
  - in RT systems depends on time constraints associated with tasks
- Example: Rate Monotonic assigns static priorities to tasks based on the lengths of their periods

Dynamic planning-based

- Feasibility is determined at run-time rather then offline
- A task is accepted for execution iff it is feasible to meet its time constraints
  - before its execution an attempt is made to create a schedule including previous tasks and the new one
  - the deadline of the extended task set must be met

Dynamic best-effort

- Used in many commercial RT systems, easy to implement
- No feasibility analysis is performed
- Typ the tasks are aperiodic so that no static scheduling analysis is possible
- When a task arrives, the system assign a priority based on its characteristics (e.g. based on earliest deadlines)
- The system tries to meet deadlines and aborts any started process whose deadline is missed
- Unitl the task is completed (or deadline arrives), it is unknow if time constraints will be met

RT Scheduling

Deadline based

Rate monotonic

Deadline scheduling (1)

- Real-time applications are not concerned with speed but with completing tasks
- Priorities provide a crude tool and do not capture the requirement of completion (or initiation) at the most valuable time
- Other deadline related information should be taken into account

Deadline scheduling - infomation used

- Ready time
  - time at which task is ready for execution. For periodic task it is a sequece of times known in advance
- Starting deadline
- Completion deadline
  - typical RT application will have either starting or completion deadlines, but not both
- Processing time
  - in some cases it is supplied, in others OS measures an exponential average
- Resource requirements
  - in addition to microprocessor

Deadline scheduling - infomation used

- Priority
  - measures relative importance of tasks. Hard-RT tasks have “absolute” priority
- Subtask structure
  - task possibly decomposed in mandatory (the only with hard RT deadlines) and optional subtasks

Design issues considesing deadlines

- Which task to schedule next?
  - For a given preemption strategy, using either starting or completion deadlines, scheduling tasks with the earliest deadline minimized the fraction of tasks that miss their deadlines (true for single and multiprocessor configurations)
- What sort of preemption is allowed?
  - When starting deadlines are specified, nonpreemptive scheduler makes sense
  - RT task has the responsability to block itself sfter executing the critical portion, allowing starting other RT deadlines

Executing profile of two periodic tasks

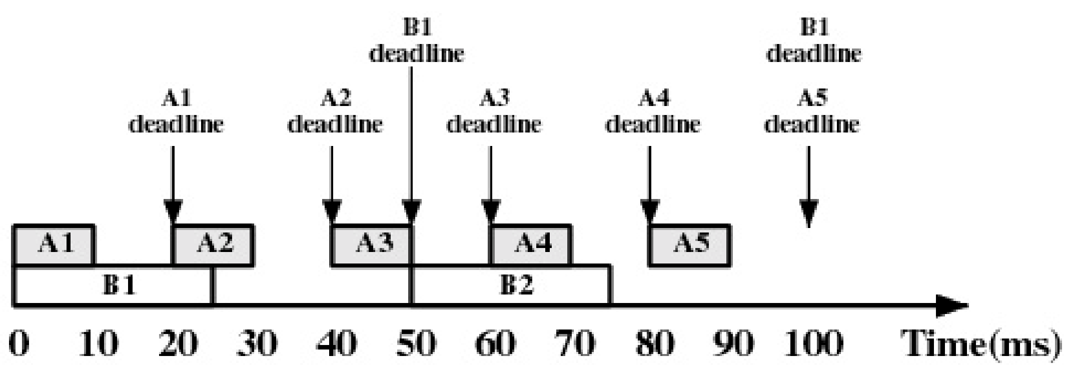
- System collecting and processing data from two sensors, A and B
  - Deadlines: A must be read every 20 ms, B every 50 ms
  - Processing time (including OS overhead): 10 ms for A samples, 25 ms for data from B
  - The computer makes scheduling decision every 10 ms

Example

Executing profile of two periodic tasks

Process	Arrival Time	Execution Time	Ending Deadline
A(1)	0	10	20
A(2)	20	10	40
A(3)	40	10	60
A(4)	60	10	80
A(5)	80	10	100
•	•	•	•
•	•	•	•
•	•	•	•
B(1)	0	25	50
B(2)	50	25	100
•	•	•	•
•	•	•	•
•	•	•	•

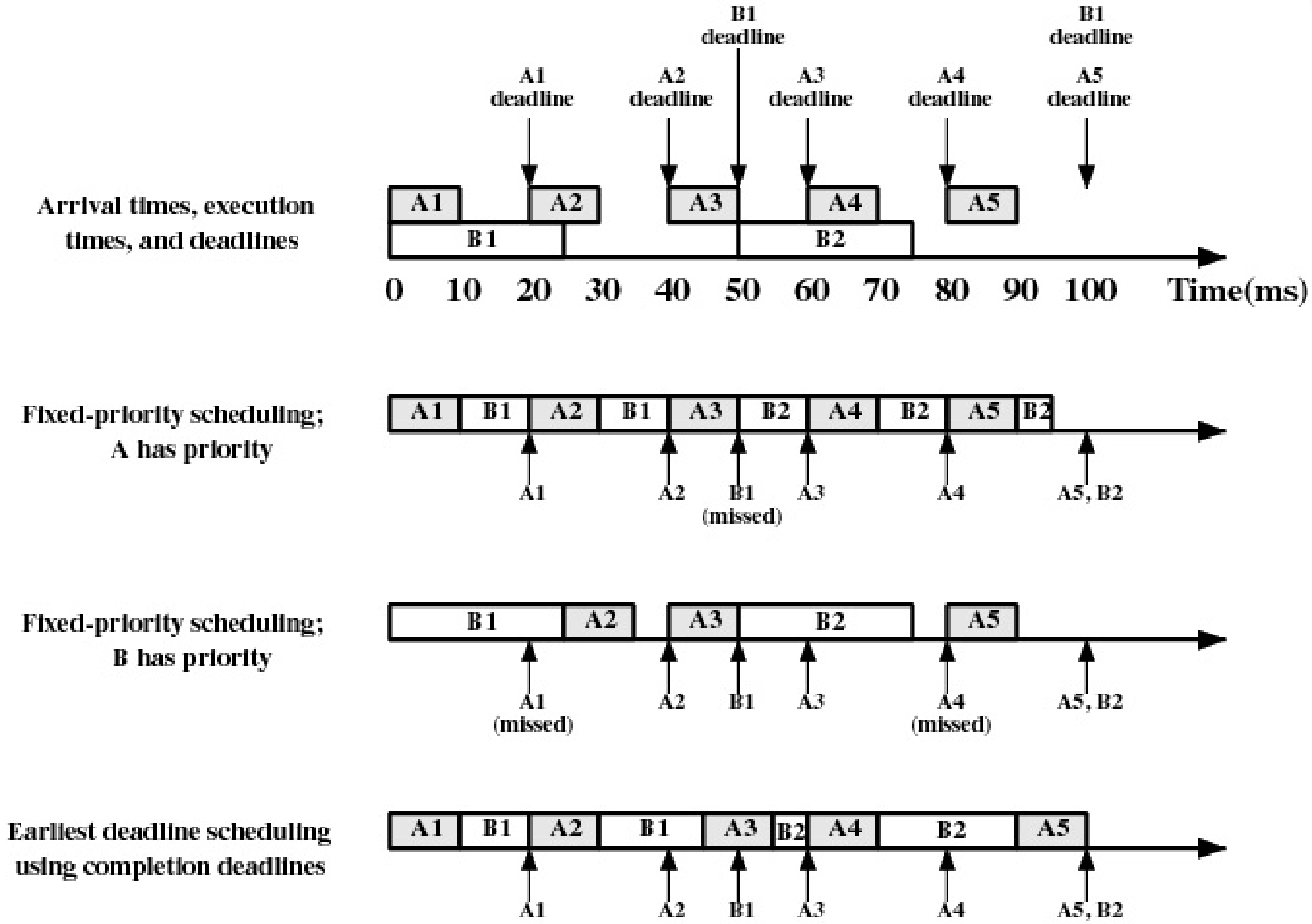
Arrival times, execution times, and deadlines



Scheduling of periodic RT tasks with Completion deadlines (1)

- Priority scheduling
  - A has higher priority -> B fails deadline
  - B has higher priority -> A fails deadline
- Earliest deadline schema
  - the scheduler gives priority at any preemption point to the task with the nearest deadline
  - All the deadlined can be met

Scheduling of PERIODIC RT tasks with COMPLETION deadlines (2)



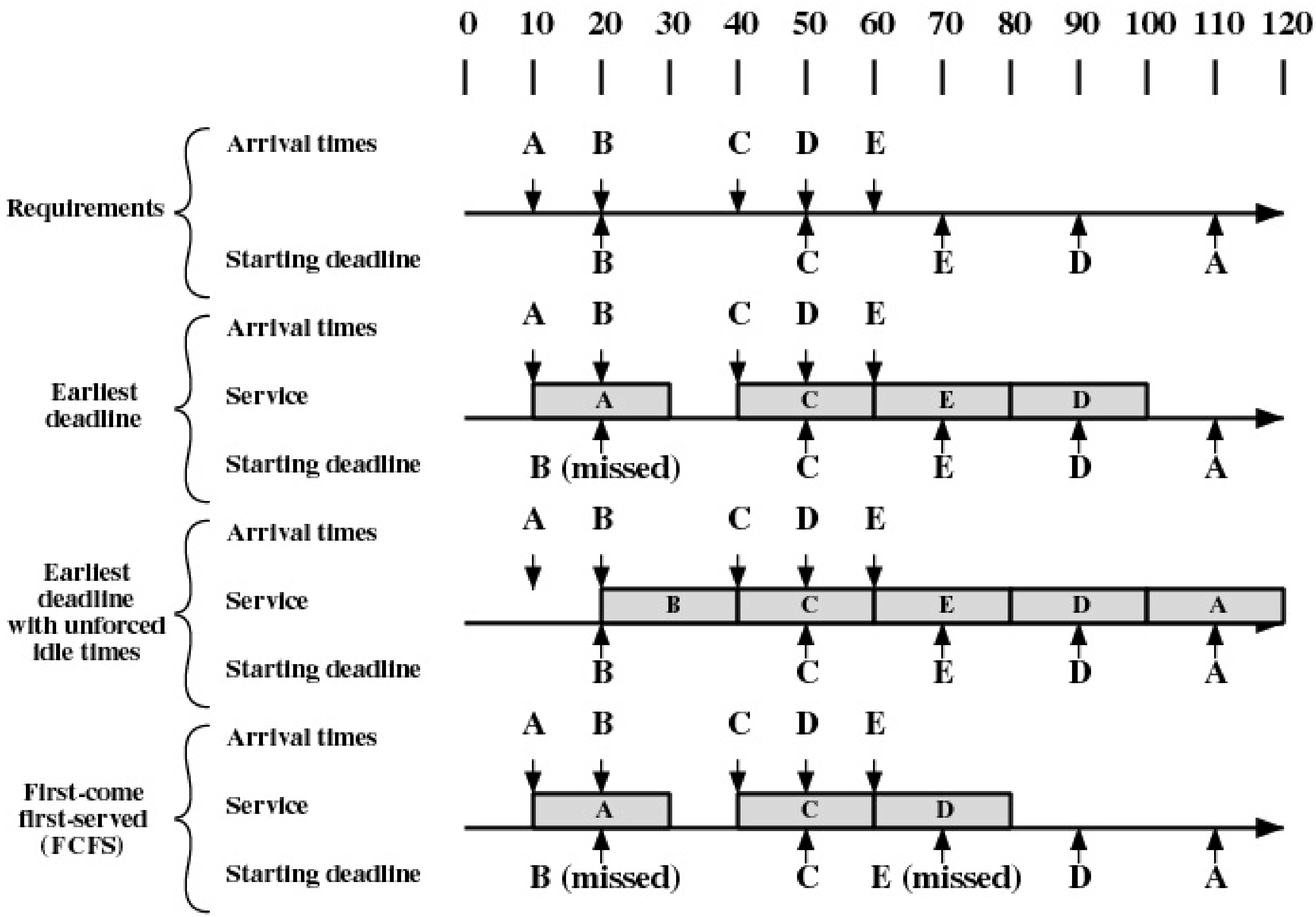
Scheduling of APERIODIC RT tasks with STARTING deadlines (1)

- Five tasks each having execution time of 20 ms
- Earliest deadline
  - schedule the ready task with the earliest deadline and let the task run to completion
  - The immediate service required by B is denied. Typ case of aperiodic task with starting deadline
- FCFS
  - Tasks B and E do not meet their deadlines

Scheduling of APERIODIC RT tasks with STARTING deadlines (2)

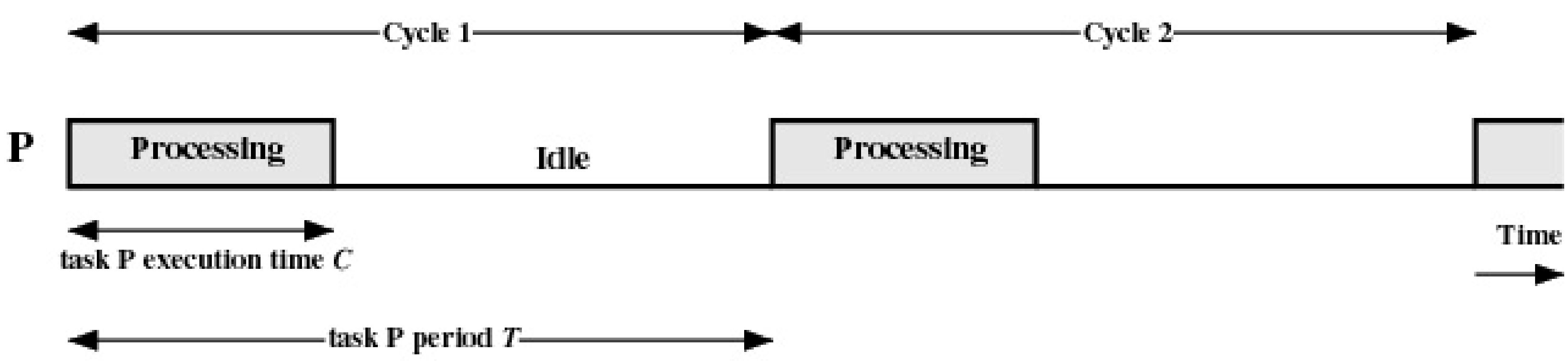
- Earliest deadline with unforced idle times
  - Refinement possible if deadlines can be know in advance of the time when the task is ready
  - Always schedule the *elegible* task with the earliest deadline and let the task run to completion
  - Elegible tasks may not be ready -> processor can remain idle though there are ready tasks
  - All the requirements are met, even with non optimal processor exploitation

Scheduling of APERIODIC RT tasks with STARTING deadlines (3)



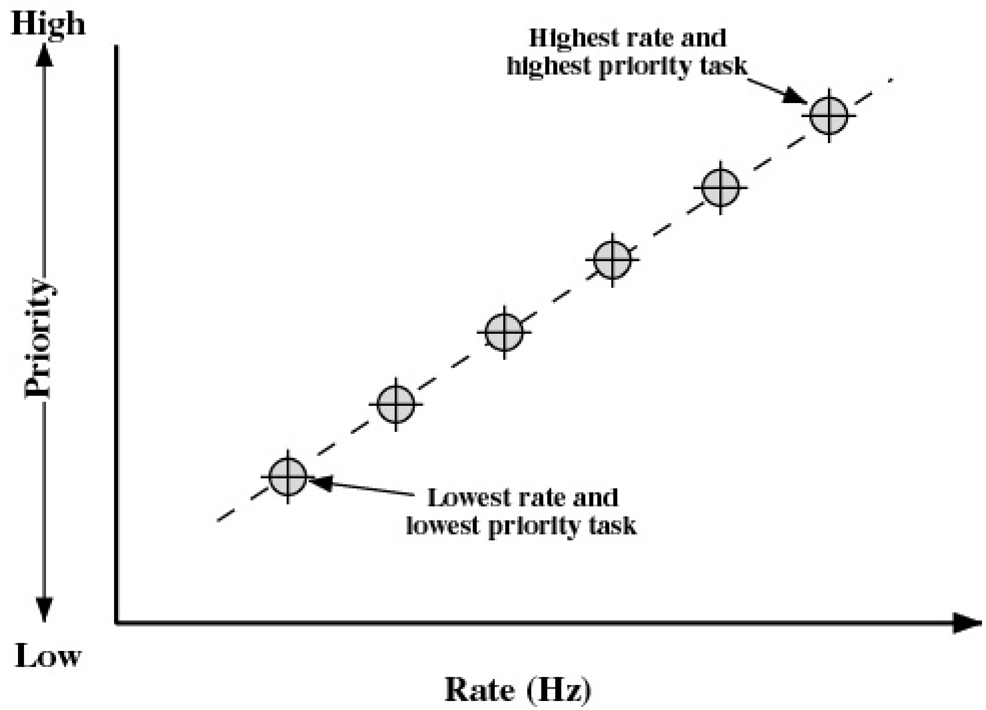
Rate Monotonic Scheduling (RMS)

- Each Task has a period T and an Execution time C for each occurence of the task
- In uniprocessor systems C < T
- Processor Utilization U= C/T (< 100%)



Rate Monotonic Scheduling (RMS)

- Assigns priorities to tasks on the basis of their periods
- Highest-priority task is the one with the shortest period



Scheduling of periodic tasks: evaluation

- Effectiveness measure of a periodic scheduling algorithm: capability to meet deadlines
- n tasks with a fixed period and execution time, to meet all deadlines, the processor utilizations of individual tasks must not exceeds the available computational power

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \ldots + \frac{C_n}{T_n} \leq 1$$

- For RMS, it can be shown

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \ldots + \frac{C_n}{T_n} \leq n(2^{1/n} - 1)$$

RMS - schedulability analysis

- The total utilization of the tasks must be less than the upper bound
- Examples of upper bounds
  - n=2: 0.82
  - n=4: 0.75
  - n=6:0.73
  - …
  - n->∞:0.693 (ln 2)
- The same constraint also holds for Earliest Deadline scheduling
- It is possible to achieve greater processor utilization with the EDF scheduling, nevertheless RMS is widely used for industrial applications…why?

RMS wins over EDF?

- Performance difference is small in practice
  - upper bound is conservative, usage of 90% frequent
- Most Hard RT systems also have soft RT components
  - e.g. built-in self test, displays, … can execute with low priority to absorb processor time left by RMS
- Stability is easier to achieve with RMS
  - in case system problems(e.g. due to transient errors), deadlines of essential tasks must be guaranteed
  - in static priority assignment one only need to ensure that essential tasks have relatively high priorities
  - in RMS is sufficient to organize essential tasks to have short periods or to modify RMS priority
  - in EDF a periodic task’s priority can change from one period to another, difficult to meet deadlines

RT Scheduling - Addendum

Cyclic Scheduling

Deterministic Scheduling

Capacity-Based Scheduling

Dynamic Priority Scheduling

Scheduling Tasks with Imprecise Results

Scheduling

- Cyclic Scheduling
- Deterministic Scheduling
- Capacity-Based Scheduling
- Dynamic Priority Scheduling
- Scheduling Tasks with Imprecise Results

Cyclic Executive

- A cyclic executive is a supervisory control program
- It schedules tasks according to schedule constructed during the system design phase
- The schedule consists of a sequence of action to be taken
- Long-term external conditions are used to choose a schedule for execution

Cyclic Scheduling(1)

- The cyclic executive provides a practical means for executing a *cyclic schedule*
- The cyclic schedule is a timed sequence of computations which is to be repeated indefinitely, in a cyclic manner

Cyclic Scheduling(2)

- Event-based processing can be handled with the cyclic schedule in different ways
  - Higher priority to the processes that are reacting to events
  - Allocate slots in scheduling where aperiodic, event-based processes can be serviced
  - Service aperiodic events in the background

Cyclic Scheduling (3)

- Deterministic scheduling theory may be used to help find schedules
- Optimal deterministic scheduling algorithms require *a priori* knowledge
- Optimal non-preemptive scheduling of computations with timing constraints is NP-Hard

Cyclic Scheduling(4)

- Advantages
  - Simple to implement, efficient, predictable
- Critical issues
  - Design
  - Runtime: the system cannot adapt to a dynamically changing environment
  - Maintenance: the code may reflect job splitting and sequencing details of the schedule

Deterministic Scheduling(1)

- It provides methods for constructing schedules in which the assignment of tasks to processors is known exactly for each point in time
- Information needed a priori:
  - Vector processing time
  - Arrival time
  - Deadline
  - Priority
  - Task splitting (preemption vs non-preemption)

Deterministic Scheduling(2)

- A schedule is an assignment of processors to tasks

- Each task in a schedule has:
  - –Completion time
  - –Flow time
  - –Lateness
  - –Tardiness
  - –Unit penalty

- Schedules are evaluated using:
  - –Schedule length
  - –Mean flow time
  - –Mean weigthed flow time
  - –Maximum lateness
  - –Mean tardiness
  - –Mean weigthed tardiness
  - –Number of tardy tasks

Deterministic Scheduling(3)

- Properties of scheduling:
  - Not only measures for evaluating but also criteria for optimization
- Many of optimization problems are NP-hard
- Approximate solutions are found by relaxing some assumption

Deterministic Scheduling(4)

- Limits of deterministic schedule:
  - The computation times are not known in advance
  - Time to produce a schedule is larger than the time it takes to service the tasks
  - Tasks arrive dynamically, updated schedule is needed

Capacity-Based Scheduling(1)

- Requirements:
  - Information about the amount of computation
  - The amount of computation available
- For a restricted class of real-time activities it's possible to determine if a task is schedulable

Capacity-Based Scheduling(2)

- Assumptions:
  - Each task in the task set must be periodic
  - The deadline is the end of the period
  - The computation time must be constant
  - Neither communication nor synchronization between tasks
  - No critical regions in any of the computation

Capacity-Based Scheduling(3)

- Schedule by using fixed priority preemptive scheduling
  - Order the tasks according to their frequency
  - Assign integer priorities to the tasks
  - The higher priority assigned to highest frequency task
- This is called: Rate Monotonic Priority Assignment

Dynamic Priority Scheduling(1)

- Task priorities may change
- The approach:
  - Define a selection discipline
  - Make on-line scheduling decisions
  - Consider the instantaneous state of the system

Dynamic Priority Scheduling(2)

- Earliest-deadline-first:
  - All tasks are ready at time t=0
  - The computations are non-preemptive
  - If the algorithm can schedule without missing any deadlines it minimizes the maximum lateness in the task set.

Dynamic Priority Scheduling(3)

- Least-slack-time:
  - The slack-times the amount of time that the task can be delayed without missing its deadline
  - Order tasks according to nondecreasing slack-time
  - It maximizes minimum task lateness and minimum task tardiness

Scheduling Tasks with Imprecise Results(1)

- Considerations:
  - A certain amount of processor time will produce a reasonably accurate result.
  - Any additional processing time will increase the accuracy
- Each task is considered to be diveded into:
  - A mandatory subtask
  - An optional subtask

Scheduling Tasks with Imprecise Results(2)

- The schedule guarantees:
  - All mandatory subtasks will be completed by their deadlines.
  - The optinal subtasks are scheduled in the remaining processor time.
- Scheduling algorithms:
  - Mandatory subtasks: traditional deterministic scheduling theory
  - Optional subtasks: various scheduling criteria