Periodic Task Scheduling

Radek Pelánek

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Examples of Periodic Tasks

- sensory data acquisition
- control loops
- action planning
- system monitoring

Simplifying Assumptions

- constant period T_i
- all instances (jobs) of a task have the same computation time C_i
- no precedence relations, no resources
- preemption
- (deadline is equal to period $D_i = T_i$)
- (no aperiodic jobs)

Example

	τ_1	τ_2	τ_3
C_i	2	4	3
T_i	6	10	12

- find schedule
- think about possible scheduling algorithms

Outlook

- notions: jitter, processor utilization, schedulable utilization
- three basic approaches: static scheduling, dynamic priorities (EDF), fixed priorities (rate monotonic)
- examples
- discussion

Jitter

- deviation of the start/finishing time of consecutive instances of some task
- relative, absolute jitter
- for some applications it is important to minimize the jitter
- we do not deal with this issue in detail

Processor Utilization Factor

Definition (Processor utilization factor)

Given a set Γ of n periodic tasks, processor utilization factor U is the fraction of processor time spent in the execution of the task set:

$$U = \Sigma_{i=1}^{n} \frac{C_i}{T_i}$$

Example: $U = \frac{2}{6} + \frac{4}{10} + \frac{3}{12} = \frac{59}{60}$

Note: $U > 1 \Rightarrow$ not schedulable

Schedulable Utilization

Definition (Schedulable Utilization)

schedulable utilization U_S of a scheduling algorithm – the algorithm can feasibly schedule any set of periodic tasks o with the total utilization of the tasks is $\leq U_S$

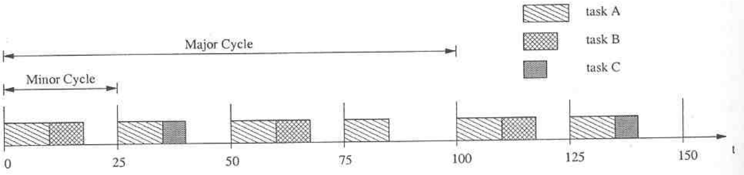
used to easily verify the schedulability of a task set

Cyclic Scheduling

- an approach, rather than algorithm
- (timeline scheduling, clock-driven scheduling)
- static schedule, constructed off-line
- schedule specifies exactly when each job executes
- *minor cycle* = greatest common divisor of periods
- *major cycle* = time after which the schedule repeats itself

Example

$$T_A = 25,\ T_B = 50,\ T_C = 100$$



Aperiodic Jobs

- spare capacities in the static schedule can be used for handling aperiodic jobs
- aperiodic jobs can be scheduled e.g. by deadline based algorithm

Advantages and Disadvantages

- advantages:
  - simple, efficient (precomputed)
  - can deal with complex requirements, precedence constraints, ...
  - special requirements can be taken into account (e.g., minimizing jitter or context switches)
- disadvantages:
  - inflexible, difficult to modify and maintain
  - fragile (overrun may cause whole schedule to fail)
  - not very suitable for systems with both periodic and aperiodic tasks

Suitable for systems which are rarely modified once built (e.g., small embedded controllers).

Earliest Deadline First

- dynamic priority assignment
- selects tasks according to absolute deadline
- does not depend on periodicity; can be directly used for periodic+aperiodic tasks

Schedulability Analysis

Schedulable utilization of EDF is 1.

Theorem

A set of periodic tasks is schedulable with EDF if and only if

$$\Sigma_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Extensions

- deadlines less than periods, aperiodic jobs
- the algorithm works directly for both extensions
- schedulability analysis is more complex (not covered)

Rate Monotonic Scheduling

- priority based algorithm: tasks scheduled according to priorities
- fixed-priority assignment: priorities assigned before the execution (all jobs of one task have the same priority)
- priorities according to periods: shorter period - higher priority
- preemptive

Example

	τ_1	τ_2	τ_3
C_i	2	4	3
T_i	6	10	12

Optimality

- in general RM is not optimal
- RM is optimal among fixed-priority algorithms

Theorem

If a task set can be scheduled by fixed-priority algorithm then it can be scheduled by Rate Monotonic algorithm.

Schedulable Utilization

For arbitrary set of periodic tasks, the schedulable utilization of the RM scheduling algorithm is

$$U_S = n(2^{1/n} - 1)$$

n	1	2	3	4	5
U_S	1.00	0.82	0.78	0.76	0.74

For high values of n the schedulable utilization converges to

$$U_{lub} = \ln 2 \sim 0.69$$

Optimality for Simply Periodic Tasks

- a set of periodic tasks is simply periodic if for every pair of tasks: $T_i < T_j \Rightarrow T_j$ is an integer multiple of T_i

Theorem

A system of simply periodic tasks is schedulable according to the RM algorithm if and only if its utilization factor is ≤ 1.

Deadline Monotonic

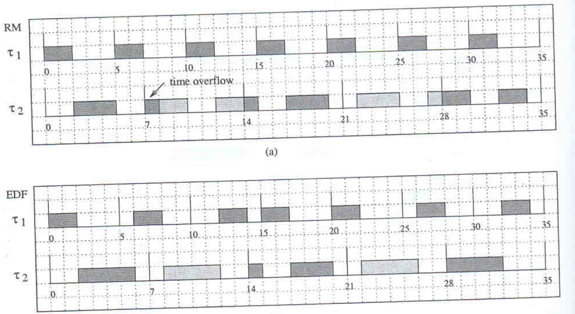
- deadlines less than period
- priorities assigned according to inverse of relative deadlines

Example

	τ_1	τ_2
C_i	2	4
T_i	5	7

- What is the utilization factor?
- Is the task set schedulable?
- What is the schedule produced by EDF/RM?

Example



(a)

(b)

$$U = \frac{2}{5} + \frac{4}{7} = \frac{34}{35} \sim 0.97$$

Example 2

	τ_1	τ_2	τ_3
C_i	2	2	2
T_i	6	8	12

- What is the utilization factor? Is the task set schedulable?
- What is the schedule produced by RM?

Example 3

	τ_1	τ_2	τ_3
C_i	1	2	3
T_i	4	6	8

- What is the utilization factor? Is the task set schedulable?
- What is the schedule produced by RM/EDF?

Comparison: RM vs EDF

	RM	EDF
implementation	multi-level priority queue, $O(1)$	heap, $O(\log n)$
processor utilization	guarantee only for 0.69, practice 0.88	full utilization
context switches	many	few
guarantee test	nontrivial	simple
predictability	good	bad

in practice: fixed-priority schedulers

Note on Predictability

- overload condition (processor utilization factor > 1), which tasks will not meet deadlines?
  - EDF – unpredictable
  - RM – predictable (tasks with the longest period)
- reminder: Apollo 11 landing
  - processor overload
  - RM algorithm used $\Rightarrow$ predictable behaviour $\Rightarrow$ decision possible

Aperiodic Jobs in Priority-driven Systems

- real systems – combination of periodic and aperiodic tasks
- main approaches:
  - fixed priority servers: scheduling of periodic tasks done by fixed priority algorithm (typically RM)
  - dynamic priority servers: scheduling of periodic tasks done by dynamic priority algorithm (typically EDF)

Assumptions and Remarks

- periodic tasks scheduled by a fixed priority algorithm (specifically rate monotonic)
- all periodic tasks start simultaneously at time $t = 0$, deadline = period
- arrival times of aperiodic tasks are unknown beforehand
- preemption
- goal: meet deadlines of periodic tasks, minimize response time of aperiodic tasks
- ordering of aperiodic tasks not discussed (done by some aperiodic scheduling algorithm, we will use FIFO)

Example

Periodic jobs:

	τ_1	τ_2
C_i	1	2
T_i	4	6

Aperiodic jobs:

	J_1	J_2	J_3	J_4
a_i	2	8	12	19
C_i	2	1	2	1

Background Scheduling

- aperiodic tasks scheduled in background (when no periodic task is running)
- schedule of periodic tasks is not changed
- major problem: high periodic load $\Rightarrow$ poor response times for aperiodic tasks

Example

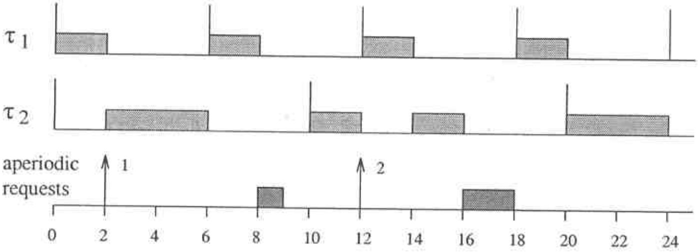


Figure 5.1 Example of background scheduling of aperiodic requests under Rate Monotonic.

Realization

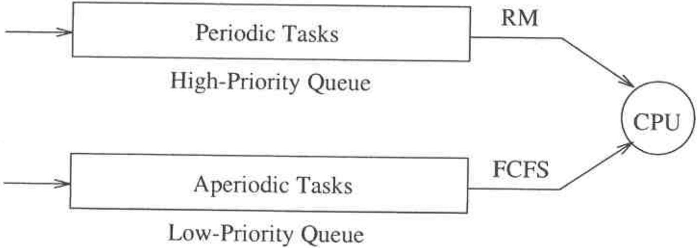


Figure 5.2 Scheduling queues required for background scheduling.

Server for Aperiodic Tasks

- periodic task whose purpose is to service aperiodic requests
- period T_S, computation time C_S (*capacity*)
- scheduled in the same way as periodic tasks
- note: selection of T_S, C_S – total utilization factor must remain ≤ 1

Polling Server

- the simplest variant of server
- when active: serve pending aperiodic requests within its capacity
- no aperiodic requests are pending ⇒ suspend

Polling Server: Example

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

$C_s = 2$

$T_s = 5$

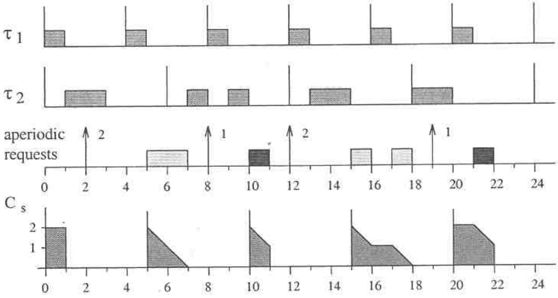


Figure 5.3 Example of a Polling Server scheduled by RM.

Improving Polling Server

- how can we improve the performance?
- (consider the previous example)

Deferrable Server

- similar to polling server
- if no aperiodic requests are pending:
  - suspend itself
  - preserve capacity until the end of the period
  - if aperiodic request arrives later during the period: it is served
- at the beginning of the period capacity is fully replenished

Deferrable Server: Example

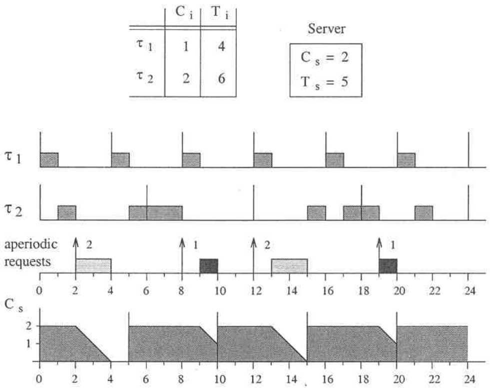


Figure 5.5 Example of a Deferrable Server scheduled by RM.

Deferrable Server: Properties

- deferrable server provides better responsiveness than polling server
- schedulability analysis more complicated
  - defferable server is not equivalent to periodic task

Deferrable Server: Example 2

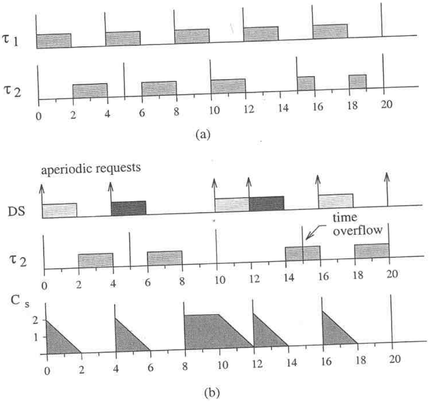


Figure 5.7 DS is not equivalent to a periodic task. In fact, the periodic set $\{\tau_1, \tau_2\}$ is schedulable by RM (a); however, if we replace τ_1 with DS, τ_2 misses its deadline (b).

Priority Exchange

- periodic server with high priority
- preserves capacity by exchanging it for the execution time of a lower-priority task:
  - at the beginning of the period: replenish the capacity
  - aperiodic requests are pending: serve them
  - no aperiodic requests are pending: exchange execution time with the active periodic task with the highest priority
- the priority exchange is performed repeatedly

Example

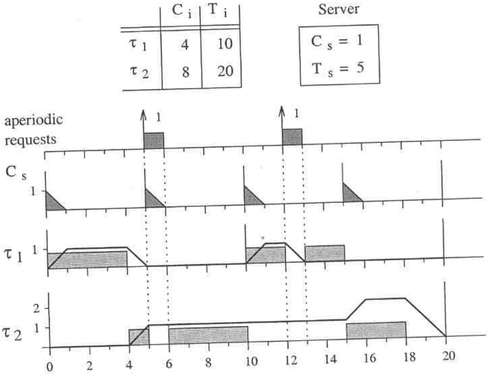


Figure 5.12 Example of aperiodic service under a PE server.

Slack Stealing

- no periodic server; passive task *Slack Stealer*
- slack $= d_i - t - c_i(t)$
- main idea:
  - no benefit in early completion of periodic tasks
  - when aperiodic request arrives: steal available slacks
- better responsiveness, more complicated schedulability analysis

Example

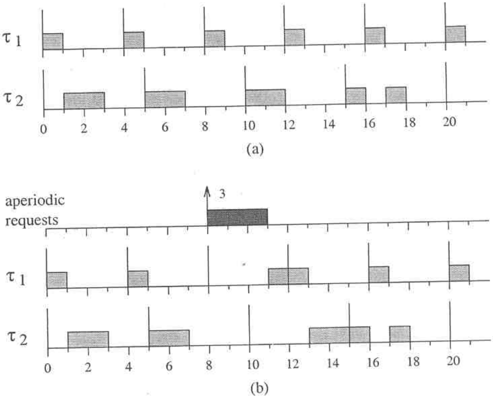


Figure 5.20 Example of Slack Stealer behavior: **a.** when no aperiodic requests are pending; **b.** when an aperiodic request of three units arrives at time $t = 8$.

Non-existence of Optimal Servers

Theorem

For any set of periodic tasks ordered on a given fixed-priority scheme and aperiodic requests ordered according to a given aperiodic queueing discipline, there does not exist any valid algorithm that minimizes the response time of every soft aperiodic request.

Similarly for average response time.

Evaluation of Fixed Priority Servers

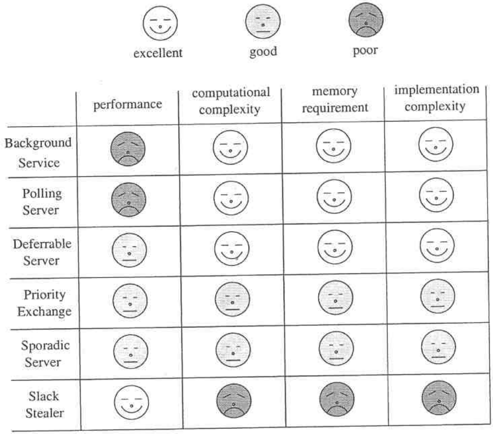


Figure 5.26 Evaluation summary of fixed-priority servers.

Example

periodic tasks

	τ_1	τ_2
C_i	1	2
T_i	5	8

aperiodic tasks

	J_1	J_2	J_3
a_i	2	7	17
C_i	3	1	1

Create schedules and determine response times:

- background scheduling
- polling server with intermediate priority
- deferrable server with the highest priority

Summary of the Lecture

- scheduling periodic tasks
  - cyclic scheduling (static schedule)
  - rate monotonic scheduling (static priorities)
  - earliest deadline first scheduling (dynamic priorities)
- processor utilization factor, schedulable utilization
- aperiodic tasks in periodic systems: fixed priority servers