Handling Sporadic Tasks in Off-Line Scheduled Distributed Real Time Systems

Damir Isović & Gerhard Fohler

Department of Computer Engineering

Mälardalen University, Sweden

Presented by : Aseem Lalani

Outline

- Introduction
- Paper Contribution
- System Description & Task Model
- Slot Shifting Method for integration of Online & Offline Scheduling
- Offline Guarantee Acceptance Test
- Offline Feasibility Test for Sporadic Tasks
- Example
- Conclusion

Introduction

- Acquisition of temporal aspects of application is either difficult or impossible to gather due to high costs or unavailability.
- Knowledge of only partial information (such as minimum inter-arrival time between tasks) of the controlled environment.
- Online Systems provide no countermeasures for sporadic task sets which are rejected.

Paper Contribution

- Method for off-line feasibility test for sporadic tasks on top of off-line scheduled distributed periodic set.
- Ability to re-schedule or re-design upon test failure.

System Description

- System is distributed viz. consists of several processing and comm. nodes
- Discrete Time Model with task periods and deadlines defined in terms of slot-length.

Task Model

- Number of slots defined by LCM of involved periods.
- Periodic Task T_P characterized by Max. Execution Time (MAXT), Period (P), and relative deadline (Dl).
- Hard Aperiodic tasks characterized by arrival time (a), maximum execution time and relative deadline.
- No deadline constraints for soft aperiodic jobs.

Task Model (cont'd)

- Sporadic tasks arrive at random points in time with defined minimum inter-arrival times between two consecutive invocations.
- Arrival order pattern not known but Max. Frequency of arrival of sporadic tasks is known.
- Sporadic Task T_S characterized by relative deadline, minimum inter-arrival time (λ) and Max. Execution Time.
- Additional On-line information available about sporadic tasks include arrival time of k^{th} invocation is its arrival time and its absolute deadline.

Slot Shifting for Integrated off-line and on-line scheduling

- Efficient method to provide on-line guarantee of scheduling aperiodic tasks on top of a distributed schedule with task constraints.
- Re-claims unused resources from off-line schedule to schedule other feasible tasks.
- Off-line preparations include

  -> Allocation of tasks to nodes, resolving precedence constraints by ordering task execution.

  -> Creating schedule tables listing start and end times of task executions.

Slot Shifting for Integrated off-line and on-line scheduling (cont'd)

-> Creating disjoint intervals for each node with tasks having the same deadline constituting one interval.

-> Calculating spare capacity for interval I_i as:

$$sc(I_i) = |I_i| - \sum_{T \in Ii} MAX(T) - \min(sc(I_{i+1}), 0)$$

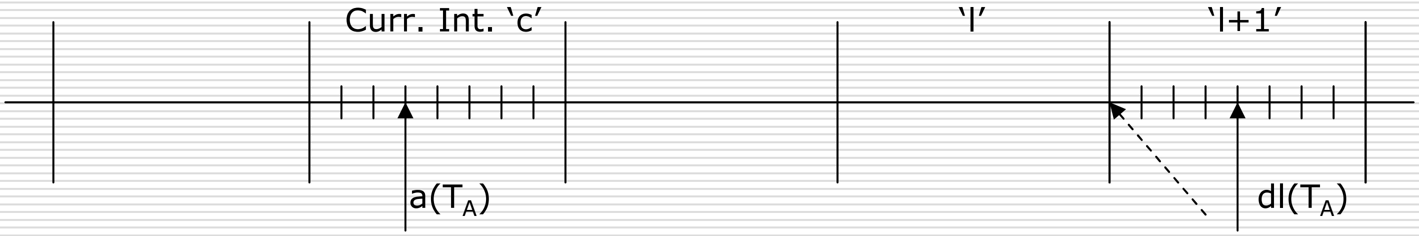
Guarantee Algorithm for Aperiodic Tasks

- At each slot, guarantee algorithm is performed on arriving aperiodic tasks.
- For each aperiodic task T_A, find

  -> A = $sc(I_c)_t$: Spare remaining capacity of current interval.

  -> B = Positive Spare capacities of full intervals between t and $dl(T_A)$.

  -> C = Min(sc of last interval, execution need of T_A before its deadline in this interval).
- $(A + B + C) > MAXT(T_A)$ guarantees acceptance of T_A



On-line Scheduling

- $sc(I_C) > 0$ => Apply EDF to set of Ready Tasks.
- $sc(I_C) = 0$ => Guaranteed task has to execute else task deadline violation will occur.
- Soft Aperiodic Tasks execute immediately when $sc(I_C) > 0$
- After each scheduling decision, update spare capacities of affected intervals.

Acceptance Test for Sporadic Tasks

- Feasible set is defined to schedule all tasks in the sporadic set such that no periodic task misses its deadline.
- The test includes:

  -> Creating an off-line schedule for periodic tasks analyzed analyzed for slot-shifting.

  -> Fitting sporadic tasks by investigating critical time slots.

  -> Re-design the system upon failure of test to manage the sporadic tasks.

More on Sporadic Tasks

- Sporadic tasks have been proven to behave like periodic tasks for worst case analysis when successive tasks arrive at minimum inter-arrival time.
- Guaranteeing this worst case load pattern at critical time slots guarantees acceptance of all sporadic tasks with greater inter-arrival times.

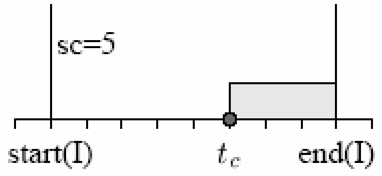


Figure 1. Example of a critical slot.

Critical slot is defined as the time slot when the execution of sporadic tasks can be delayed maximally.

Why Critical Slots are important to investigate?

- If the sporadic set can be fitted within the periodic set upon arrival at critical slot then it can be guaranteed to fit upon arrival at any other slot.

  Δ = Difference between spare capacities for sporadic task T_S at critical slot t_c and any other slot t.

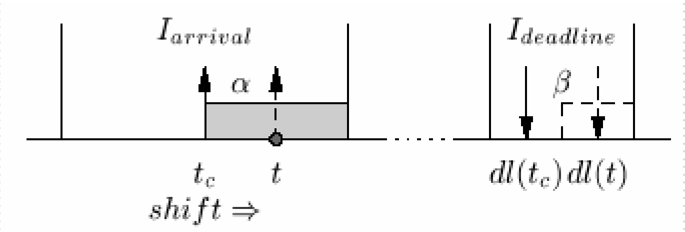
  α = Difference in spare capacity of the arrival caused by shifting arrival time T_S from t_c to t.

  β = Difference in spare capacity of the deadline interval caused by shifting the deadline of T_S..

$$\Delta = \alpha + \beta$$

Critical Slot Investigation

$t > t_c$ => T_S arrives after C.S.

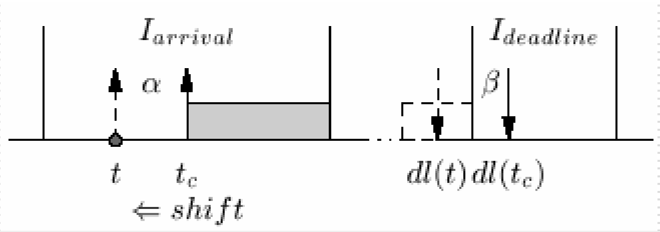


$\alpha = 0$

$\beta >= 0$

$\Delta = (\alpha + \beta) >= 0$

$t < t_c$ => T_S arrives before C.S.



$\beta_{worst} = - \alpha$

$|\beta_{opt}| < \alpha$

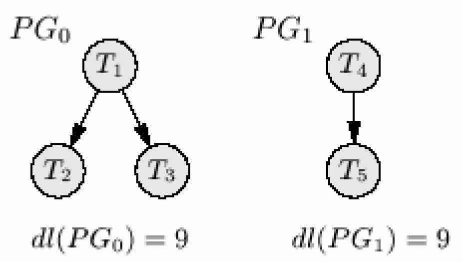
$\Delta = (\alpha + \beta) >= 0$

Contradiction !

Off-line Feasibility Test for Sporadic Tasks

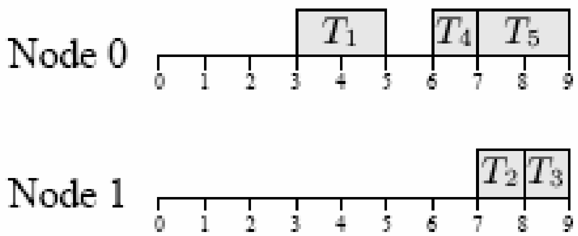
1: Investigate every critical slot.
2: No slots reserved yet.
3: Guarantee every sporadic task T_S in the set.
4: Guarantee every invocation T^n_s of T_S.
5: Calculate sc available for T_S from its arrival until its deadline. It is equal to the sum of sc for all full intervals between $I_{arrival}$ and the $I_{deadline}$ of T^n_s , increased by
6: the remaining sc of the $I_{deadline}$ available until dl(T^n_s),decreased by
7: the amount of sc reserved for other, previously guaranteed sporadics that intersect with T^n_s.
8: If the available sc is greater or equal to the maximum execution time of T_S, then
9: reserve slots needed for T^n_s as close to its dl as possible, and continue.
10: If not enough spare capacity, abort the guarantee algorithm and report that the guaranteeing failed.

Example

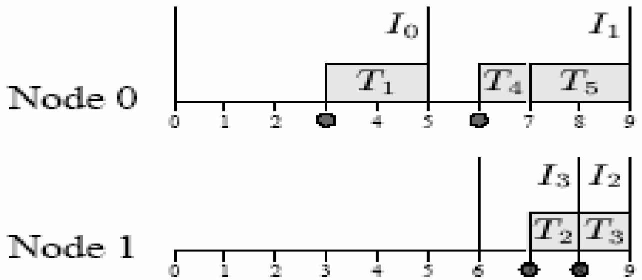


Task	MAXT	dl
T_1	2	5
T_2	1	8
T_3	1	9
T_4	1	9
T_5	2	9

(a) Precedence Graph & Task Description



(b) Task Execution at nodes



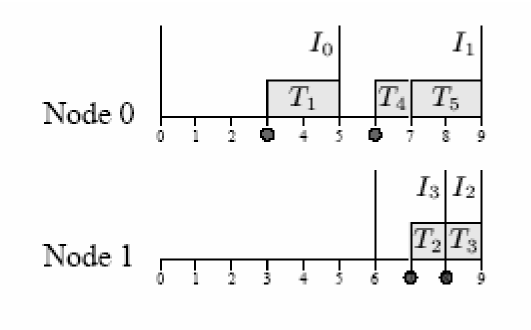
(c) Critical Slots and Intervals

Interval	*Node*	*start*	*end*	*sc*	t_c
I_0	0	0	5	3	3
I_1	0	5	9	1	6
I_2	1	6	8	1	7
I_3	1	8	9	0	8

(d) Schedule Table

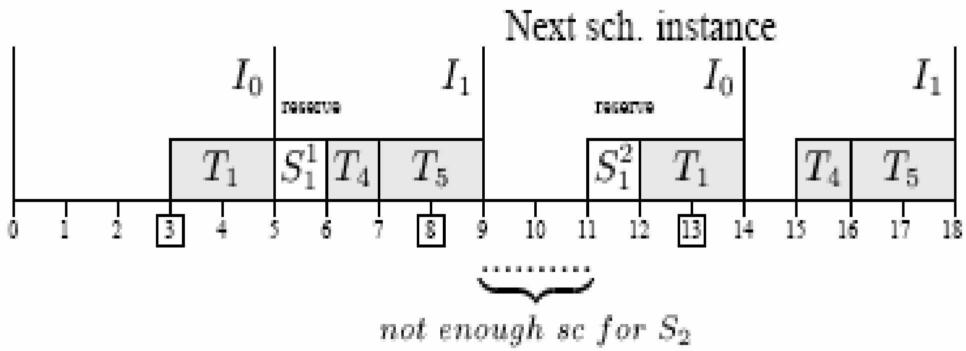
Example (cont'd)

S = {S1(1; 5); S2(3; 10)}



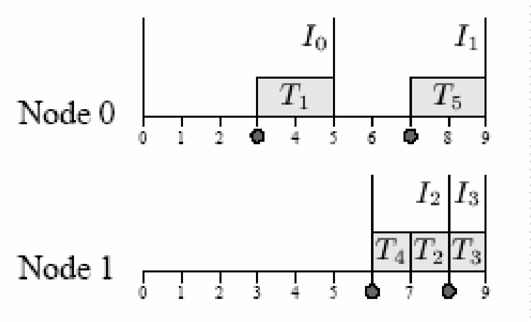
Steps in Guarantee Algorithm

1	3	4	5	8	9	10
$\mathbf{t_c}$	$\mathbf{T_S}$	**Inv.**	$\mathbf{sc_a}$	$\geq$ **MAXT(?)**	**R**	
3	S_1	1	1	$\geq 1 \Rightarrow \top$	$\{5\}$	
		2	3	$\geq 1 \Rightarrow \top$	$\{5,11\}$	
	S_2	1	2	$\geq 3 \Rightarrow \bot$		abort

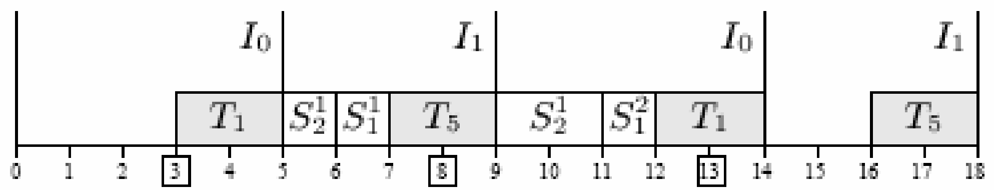


Critical Slot 3

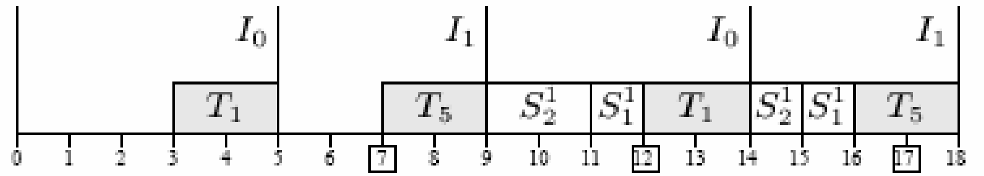
Example (cont'd)



1	3	4	5	8	9
$\mathbf{t_c}$	**Task**	**Inv.**	$\mathbf{sc_a}$	$\geq (?)$	**R**
3	S_1	1	2	$\geq 1 \Rightarrow \top$	$\{6\}$
		2	3	$\geq 1 \Rightarrow \top$	$\{6,11\}$
	S_2	1	3	$\geq 3 \Rightarrow \top$	$\{5,6,9,10,11\}$
7	S_1	1	3	$\geq 1 \Rightarrow \top$	$\{11\}$
		2	2	$\geq 1 \Rightarrow \top$	$\{11,15\}$
	S_2	1	3	$\geq 3 \Rightarrow \top$	$\{9,10,11,14,15\}$



Critical Slot 3



Critical Slot 7

Guaranteeing after Re-design

S = {S1(1; 5); S2(3; 10)}

Example (cont'd)

1. $\mathcal{R}(t) = \{\}$: There are no tasks ready to be executed, the CPU remains idle.

2. $\mathcal{R}(t) \neq \{\} \wedge \exists T_A$, T_A soft aperiodic:

   (a) $sc(I)_t > 0 \wedge \exists T_S \in \mathcal{R}(t), T_S$ sporadic $\Rightarrow$ execute T_S.

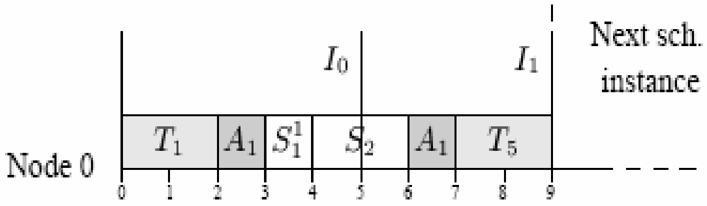
   (b) $sc(I)_t > 0 \wedge \neg\exists T_S \in \mathcal{R}(t) \Rightarrow$ execute T_A.

   (c) $sc(I)_t = 0$: a periodic task from ready set has to be executed. Zero spare capacities indicate that adding further activities will result in a deadline violation of the guaranteed task set.

3. $\mathcal{R}(t) \neq \{\} \wedge \neg\exists T_A, T_A$ soft aperiodic: The task of ready set with the shortest deadline is executed.

On-line Mechanism

t	$\mathcal{R}(t)$	case	exe.	sc
0	$\{T_1, T_5\}$	3	T_1	unchanged
1	$\{T_1, T_5\}$	3	T_1	unchanged
2	$\{T_5, A_1\}$,	2b	A_1	$sc(I_0)$ decreased
3	$\{T_5, S_1^1, S_2, A_1\}$	2a	S_1^1	$sc(I_0)$ decreased
4	$\{T_5, S_2, A_1\}$	2a	S_2	$sc(I_0)$ decreased
5	$\{T_5, S_2, A_1\}$	2a	S_2	$sc(I_1)$ decreased
6	$\{T_5, A_1\}$	2b	A_1	$sc(I_1)$ decreased
7	$\{T_5\}$	3	T_5	unchanged
8	$\{T_5, S_1^2\}$	2c	T_5	unchanged



On-line Execution at Node 0 with Aperiodic task A_1 (2,2)

Conclusion

- Sporadic Tasks are guaranteed during design time allowing re-scheduling or re-design in case of failure.
- Efficient method since major part of preparation is off-line and on-line mechanisms are simple.
- Slot shifting algorithm allows reclaim of unused resources allowing high resource utilization.