Slack-stealing scheduling

1

Introduction

Contextualization and
Motivation

Contextualization

- Scheduling tasks in real time systems

- (Example) Robotics:
  - Sensor for monitoring the environment
  - Send data to server

- Run all those tasks without collision within a certain time.

Motivation

- How do we schedule hard-periodic and soft-aperiodic tasks in Fixed-priority Preemptive System?

  (Concepts explained at next section)

- Is it possible to guarantee that all periodic and aperiodic tasks will be completed?

2

Review

Important concepts

LET’S REVIEW SOME CONCEPTS

Periodic Tasks

Tasks represented as a tuple T = <Φ, T, C, D>

Φ → first time task appears

T → Period of the task

C → execution time

D → relative deadline

Aperiodic Tasks

Tasks without deadline or with soft deadline that can start in different times.

Soft Tasks

Tasks with extendable deadlines

Hard Tasks

Tasks that must be completed until the deadline

LET’S REVIEW SOME CONCEPTS

Fixed-priority system

Tasks have a fixed order of execution that must be respected

Dynamic-priority system

The order that the tasks must be executed is calculated during the system execution

Preemptive Systems

The execution of one task can be stopped by the system due to the need of processing other tasks.

3

Main Content

How to address the scheduling problem

Older Standard Approaches

- Always Execute the Periodic Tasks first
- Await for idle time to execute aperiodics
- Or create a periodic task to handle it
- Cons:
  - Larger response time for aperiodics

Example - Using Idle Available Time

Periodic tasks:

T_1 = <Φ_1=0, T_1=4, C_1=1, D_1=1>

T_2 = <Φ_2=0, T_2=6, C_2=3, D_2=6>

Aperiodic tasks(arrival at 0):

p_1 = <C_1=1> → R_1 = 6

p_2 = <C_2=2> → R_2 = 12

task	τ_1	τ_2	τ_2	τ_2	τ_1	p_1	τ_2	τ_2	τ_1	τ_2	p_2	p_2
time	1	2	3	4	5	6	7	8	9	10	11	12

The Slack Stealing Approach

- Idea: “Steal” processing time from periodics without causing time faults.
- How: pre-compute slack time availables.
- Goal: reduce the response time of aperiodic tasks (a.k.a decrease the finish time)

Example - Using Slack-Stealing

Periodic tasks:

$T_1 = <\Phi_1=0,\ T_1=4,\ C_1=1,\ D_1=1>$

$T_2 = <\Phi_2=0,\ T_2=6,\ C_2=3,\ D_2=6>$

Aperiodic tasks(arrival at 0):

$p_1 = <C_1=1> \rightarrow R_1 = 2 < 6$

$p_2 = <C_2=2> \rightarrow R_2 = 8 < 12$

task	τ_1	p_1	τ_2	τ_2	τ_1	τ_2	p_2	p_2	τ_1	τ_2	τ_2	τ_2
time	1	2	3	4	5	6	7	8	9	10	11	12

The Slack Stealing - Basic Scenario

- All periodic tasks variables are already know.
- Tasks do not stop themselves nor sync with each other.
- Any task can be instantly preempted.
- Scheduling overhead are assumed to be zero.
- Aperiodic tasks has soft deadlines (but can be hard with some adjustments)

The Slack Stealing - Algorithm

1. Pré-runtime configurations
   a. Define a fixed priority algorithm for the periodic tasks
      (ex: Earliest Deadline (EDS), Rate Monotonic (RTS))
   b. Compute the Hyperperiod H (LCM of periods)
   c. Define a priority for the aperiodic tasks
      (ex: Earliest Arrival Time)

The Slack Stealing - Algorithm

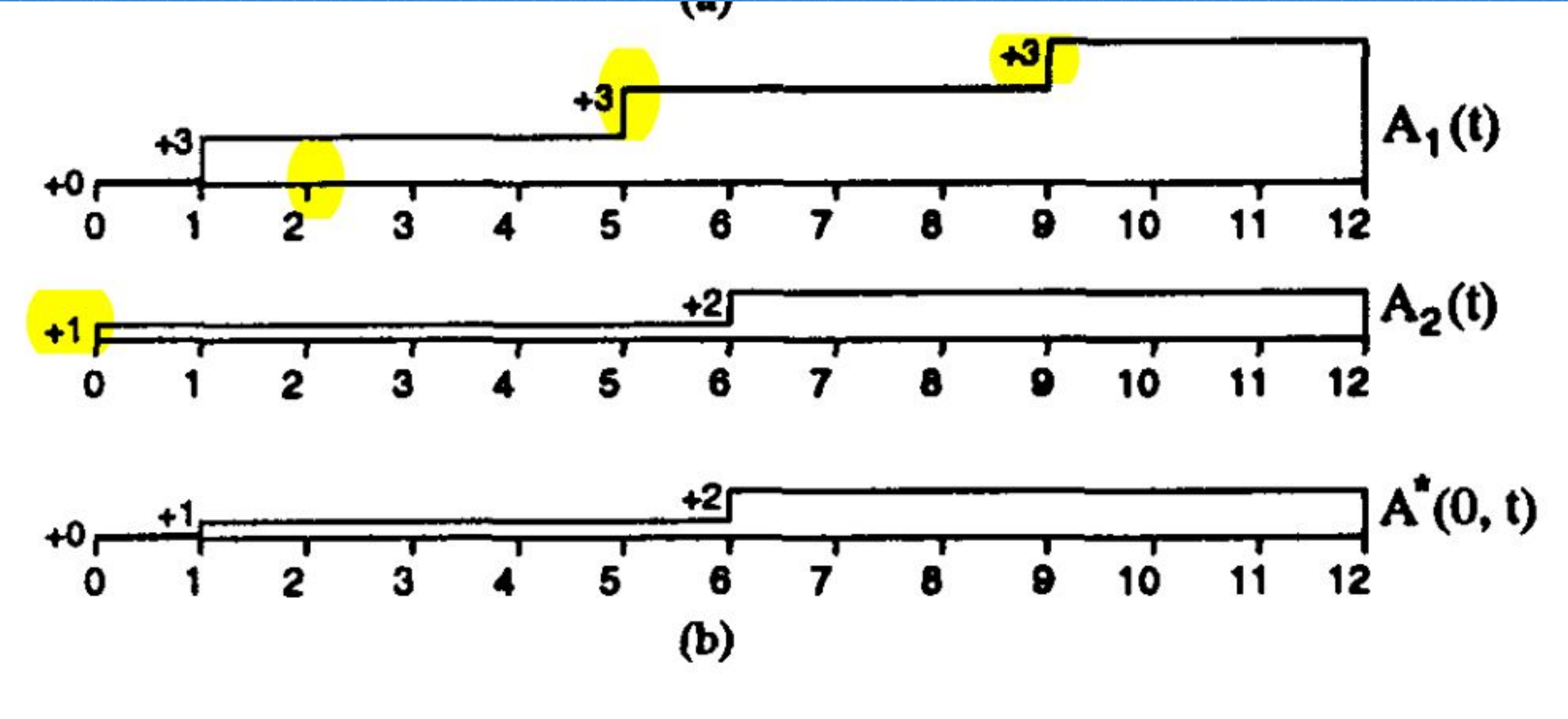
1. Pré-runtime configurations
   a. Compute Periodic Ready Work for all tasks

      $P_i(t) \rightarrow$ qtd of processing done or waiting to be done by periodic tasks from [0, t] at level i or higher.

   b. Compute the Maximum Aperiodic Processing Available.

      $A_i(t) \rightarrow$ max aperiodic work in [0, t] at level i or higher with all τ_i deadline meet

The Slack Stealing - Algorithm



The Slack Stealing - Algorithm

1. Pré-runtime configurations
   a. Initialize Accumulators

      $A(s) \rightarrow$ max aperiodic work done in [0, s] with all τ_i deadlines meet (this was a horrible variable name choice)

      $I_i(s) \rightarrow$ Accumulate Inactive time at level i or higher

The Slack Stealing - Algorithm

1. Scheduler Algorithm
    a. The scheduler has 2 queues:
        i. The Periodic Tasks Queue
        ii. The Aperiodic Tasks Queue
    b. And can be in one of these three states
        i. Idle → Increase all $I_i(s)$ accumulators
        ii. Running τ_i → Increase $I_j(s)$ for $j > i$
        iii. Running p_i → Increase $A(s)$

The Slack Stealing - Algorithm

1. Scheduler Algorithm - Scenarios
   a. If τ_i is running and $\tau_{j\ (j<i)}$ arrives, τ_j start running
   b. If a periodic ends or a aperiodic arrives we
      i. Check if the aperiodic queue is not empty
      ii. Check if we have slack time available to use by computing the following

         $A_i(s, t) \rightarrow$ max aperiodic work available in [s, t] at level i or higher with all τ_i deadline meet

         $A^*(s, t) \rightarrow$ max aperiodic work in [s, t] with all τ_i deadline meet

Slack Stealing - Example

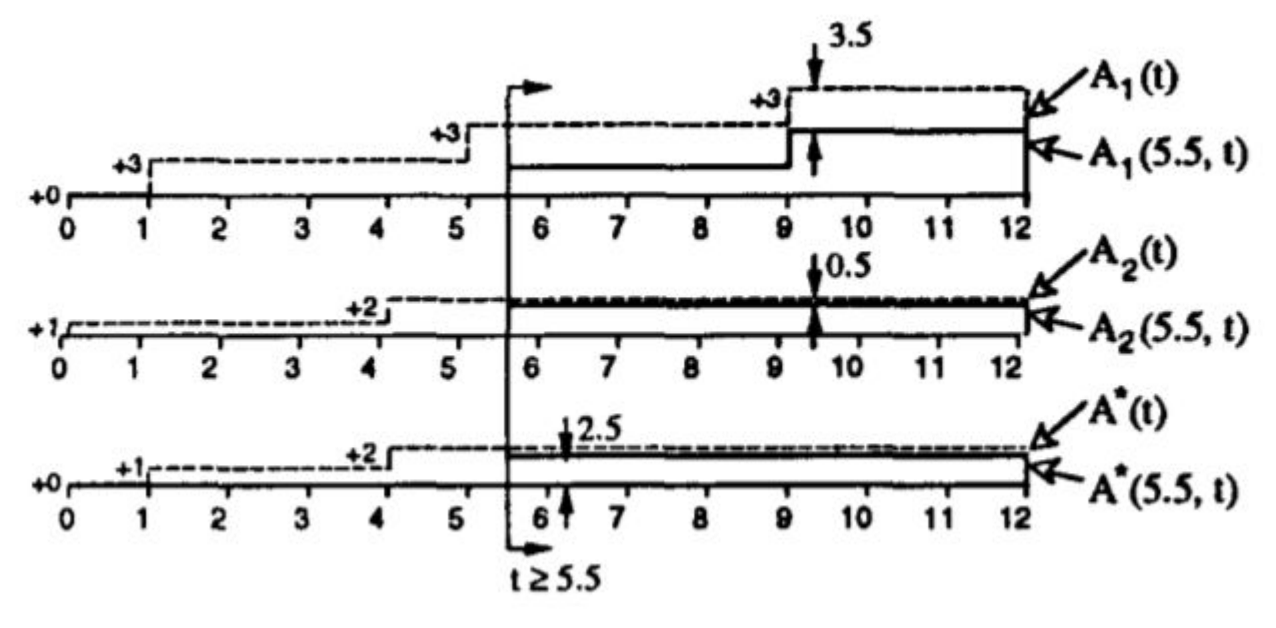


Figure: Example from slack stealer paper

Optimality of the Slack Stealer

Set of periodic tasks scheduled by fixed-priority and aperiodic stream

↓

Slack Stealing min response time of aperiodic and meet deadline of periodic

(mathematical proof on original paper)

4

Illustration

Demo implementation of slack stealing

5

References

An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems

John P. Lehoczky[†] and Sandra Ramos-Thuel[‡]

Department of Statistics[†]
Department of Electrical and Computer Engineering[‡]
Carnegie Mellon University,
Pittsburgh, PA 15213

Abstract

This paper presents a new algorithm for servicing soft deadline aperiodic tasks in a real-time system in

solution of many practical problems which arise in actual real-time systems including task synchronization, transient overload, and simultaneous scheduling of both periodic and aperiodic tasks, among others.

- Jeon, W.; Kim, W.; Lee, H.; Lee, C.-H. Online Slack-Stealing Scheduling with Modified laEDF in Real-Time Systems. Electronics 2019, 8, 1286. doi.org/10.3390/electronics8111286
- J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems," [1992] Proceedings Real-Time Systems Symposium, 1992, pp. 110-123, doi: 10.1109/REAL.1992.242671.
- Real-Time Systems classes from the course
- https://www.geeksforgeeks.org/tasks-in-real-time-systems/

Thanks!

ANY QUESTIONS?

Tiago Marino

Gabriel Van Loon

Rodrigo Arboleda

João Vitor Ramos

Code at Github:
/firewall1011/Slack-Stealing-Scheduling