Real Time Systems – SS2016

Prof. Dr. Karsten Weronek

Faculty 2

Computer Science and Engineering

Scheduling

The aim for RT-planning

Find a relation between a set of jobs and
the resources to perform these jobs, that

- all required resources (computation time, memory, devices, etc.) are available to the jobs and
- the achievement of all time-related requirements is garanteed

A Schedule

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- A schedule or a timetable, as a basic time-management tool, consists of
  - a list of times at which possible tasks, events, or actions that are intended to take place, or/and
  - of a sequence of events in the chronological order in which such things are intended to take place.

The process of creating a schedule

- deciding how to order these tasks and how to commit resources between the variety of possible tasks –

is called scheduling, and

a person responsible for making a particular schedule may be called a scheduler.

Defintion: Schedule

For RTS:

Definition:

A schedule of a set of jobs is called feasible(viable) when
each job can be completed with its individual Deadline.

To schedule means
to decide, which process will be processed in which time frame.

But how to find such a feasible schedule ?

Scheduling (non-)optimial Algorithm (Strategy)

Definition:

An **scheduling algorithm** is called **optimal** if
it is able to create a feasible schedule
in those cases in which an useful schedule exists.

A **non-optimal** scheduling algorithm may not be able
to create an feasible schedule.

Points in Time, Periods in Time

Points in time means

there is an event K, that is so short in time, that it can be assumed that the event take no time and can be defined by

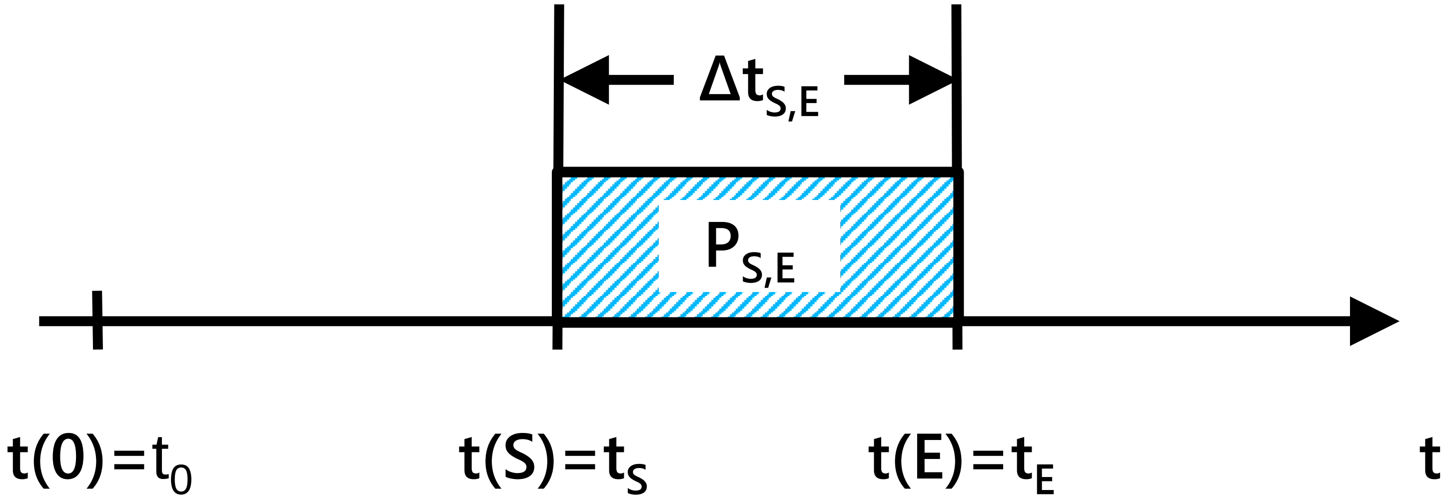
a single number t(K) on the **time bar t.**

points in time, periods in time

Periods in time means

- there is a defined time frame P, that
- takes some time Δt, and can be defined by
- a start point t(S) and an end point t(E) on the time bar.
- The period is defined by: $P_{S,E} = P[\ t(S)\ |\ t(E)\ ] = P(\ t_S\ |\ t_E\ )$
- The duration of the time frame P can then be calculated by

$$\Delta t(P_{S,E}) \quad = \quad t(P_E) - t(P_S) \quad = \quad t_E - t_S \quad = \quad \Delta t_{S,E}$$



What is the issue?

Starting from

$$\Delta t(P_{S,E}) \quad = \quad t(P_E) - t(P_S) \quad = \quad t_E - t_S \quad = \quad \Delta t_{S,E}$$

and define $t(P_S) \quad = \quad t_S \quad = \quad 0$ (Start point is set to 0) will become

$$\Delta t(P_{0,E}) \quad = \quad t(P_E) - 0 \quad = \quad t_E - 0 \quad = \quad \Delta t_{0,E}$$

leads to

$$\Delta t(P_E) \quad = \quad t(P_E) \qquad \text{and:} \quad t_E \quad = \quad \Delta t_E$$

In a lot of documentation (books, scripts, journals, etc.)

there is no difference between point in time and periods in time.

You have to keep in mind what is meant!

Develop picture on whiteboard

Coming soon …

Points in time and time intervals (all named be time!)

Points in time:

Release time	t_r	time when a job is ready to run
Starting time	t_s	time when job really starts
Completion Time	t_c	time when job is done
Deadline	t_d	time until job needs to be completed

Points in time and time intervals (all named be time!)

Time intervals:

Execution time	t_{exec}	„CPU-time“	$t_c - t_s$
Response time	t_{resp}	„reaction time“	$t_c - t_r$
Tardiness	t_{tard}	„too late“	$t_c - t_d$; for $t_c < t_d = 0$
Slack time	t_{slack}	„time to idle“	$t_d - t_c$ for $t_c < t_d$

Warning: t_d is based on t_r

Computation-time request

The physical process creates a request for a computation task. This task is identified by a letter (e.g. **u**).

The computation-time request needs a certain a mount of CPU time and has a certain deadline, until the computation must deliver a proper result.

Peridic Requests

The release time is the point in time when the computation-time request appeares.

Release Time: $t_r(u)$ or $t_{r,u}$

In periodically apearing processes the Release Times of requests of the same type can be numbered:

$t_{r,u,1}$, $t_{r,u,2}$, $t_{r,u,3}$, $t_{r,u,4}$, ...

Process Time/Period; Rate

The time difference between two requests of the same type is called Process Time or Process Period.

The process time: $t_{p,u}$

The process may vary between: $t_{pmin,u} \leqslant t_{p,u} \leqslant t_{pmax,u}$

For RTS t_{pmax} is not relevant.

The inverse of the period is called rate:

For RTS the maximal Rate is: $r_{max,u} = \frac{1}{t_{pmin,u}}$

Maximum Reaction Time/Deadline

The Deadline is the maximum time until a computation-time request needs to be accomplished (i.o.w. needs to be completed in the way that it could be initiated a second time.

In RTS there is a minimal reaction time: t_{dmin}

And a maximum reaction (Deadline): t_{dmax}

Phase

The phase reflects the minimal time delay between a computation-time request and the the time reference point.

Phase: $t_{\Phi min} = \min \{ u(t) - t_0 \}$

In most cases it is assumed to be zero:

$$t_{\Phi min} = 0$$

Execution Time/Computation Time, Runtime,

The execution time is the time that
a computation-time request
is consuming Computer-time.

Is the request for CPU-time only it is equivalent to the CPU-time.

However owing to some latency issues it is slightly higher.

The Execution varies between a
Best Case Execution Time (BCET) and a
Worst Case Execution Time (WCET)

Response Time

The Response Time is defined as the time between
the arrival of a computation-time request (Release Time) and
the completion time of the computation.

$$t_{resp} = t_u - t_c$$

Don‘t confuse the Response Time and the waiting time. The waiting is the between the reuquest and the start of execution. So the response includes the waiting time already.

Latency Times

Latency time are the time that is needed for computer internal processing (operation system and processor internal) and leads to couple of time delay between an initiation and the start of the requested task.

There are

- Interrupt-Latency
- Task-Latency
- Kernel-Latency
- Preemption Delay (Verdrängungszeit)

Latencies

Interrupt Latency

Time between an interrupt request and
the start of the related interrupt routine.

Task Latency

Time between the initiation of a task and
the start of the task.

Kernel Latency

are OS-internal latencies
owing to blocking sequeces of certain hardware areas.

Preemptive Delay

The delay for stopping and/or removing a currently
running code sequence, to load and start a new one (context switch)

Simplification

For the scheduling the time will be devided evenly into reasonable time-slices.

For RTS scheduling you aim to describe your problem as a periodic problem to have a finite problem to solve.

If you have a solution for one period you have it for all the time.

For the beginning let assume that all tasks will be independent.

Easy Task model

- Non periodic/aperidoc (three parameters)

  - A: arriving time (of the request)
  - C: computing time
  - D: deadline (relativ to arrival)

Remark:

A $= t_A = t_r$ is the release time

C $= t_{exec} = t_s - t_c$

D $= t_{execmax}$ = WCET

Then a computation request/task can be defined as u = (A, C, D)

A schedule than may be u_1 , u_s , u_1, ... u_n

Scheduling Problems

Given a set of tasks (ready queue)

- Check if the set is schedulable
- If yes, construct a schedule to meet all deadlines
- If yes, construct an optimal schedule
  e.g. minimizing response times

Tasks with the same arrival time

Assume a list of tasks

(A1, C1, D1), (A2, C2, D2) . . . (An, Cn, Dn)

Assume: A1 = A2 = . . . = An = 0   (same arrival time at 0sec)

The you can leave off A an the task list is

→ (C1, D1), (C2, D2) . . . (Cn, Dn)

→ Is there a feasible schedule?

→ How to find a feasible schedule?

→ May be, there are many feasible schedules!

Earlist Due Date fist (EDD) by Jackson 1955

EDD: order tasks with nondecreasing deadlines.

EDD is a simple form of EDF (earliest deadline first).

Example: (1,10)(2,3)(3,5)

Schedule is (2,3)(3,5)(1,10)

EDD is optimal

if EDD can‘t find a feasible schedule

EDD: Schedulability test

IF $C1+C2+\ldots+Ck \leq Dk$ for all $k \leq n$

for the schedule with nondecreasing ordering of deadlines,

then the task set is schedulable

Response time for task i: $Ri = C1 + \ldots + Ci$

Wenn

für einen (Ablauf-)Plan mit

nicht-fallender Sortierung nach Deadlines,

$C1+C2+\ldots+Ck \leq Dk$ für alle $k \leq n$

dann ist der Plan ausführbar.

Die Antwortzeit für den task i: $Ri = C1 + \ldots + Ci$