Fixed Priority Scheduling

Real-Time Operative Systems Course

Paulo Pedreiras, DETI/UA/IT

October 17, 2022

1. Preliminaries
2. Online scheduling with fixed priorities
3. Schedulability tests based on utilization
4. Response time analysis

Last lecture

- The concept of temporal complexity
- Definition of schedule and scheduling algorithm
- Some basic scheduling techniques (EDD, EDF, BB)
- The static cyclic scheduling technique

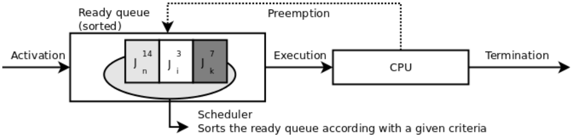
Agenda for today

Fixed-priority online scheduling

- Rate-Monotonic scheduling
- Deadline-Monotonic and arbitrary priorities
- Analysis:
  - The CPU utilization bound
  - Worst-Case Response-Time analysis

Online scheduling with fixed priorities

- The schedule is built while the system is operating normally (**online**) and is based on a **static criterium** (priority)
- The ready queue is **sorted by decreasing priorities** . Executes first the task with highest priority.
- If the system is preemptive, whenever a task job arrives to the ready queue, if it has higher priority than the one currently executing, it starts executing while the latter one is moved to the ready queue
- Complexity: $O(n)$



Online scheduling with fixed priorities

Pros (with respect to Static Cyclic Scheduling)

- Scales
- Changes on the task set are immediately taken into account by the scheduler
- Sporadic tasks are easily accommodated
- Deterministic behavior on overloads
  - Tasks are affected by priority level (lower priority are the first ones)

Cons (with respect to Static Cyclic Scheduling)

- More complex implementation (w.r.t. static cyclic scheduling)
- Higher execution overhead (scheduler + dispatcher)
- On overloads (e.g. due to SW errors or unpredicted events) an higher priority tasks may block the execution of lower priority ones

Online scheduling with fixed priorities

Rules for priority assignment to tasks

- Inversely proportional to period (RM – Rate Monotonic)
  - Optimal among fixed priority scheduling criteria if D=T
- Inversely proportional to deadline (DM – Deadline Monotonic)
  - Optimal if $D \leq T$
- Proportional to the task importance
  - Typically **reduces the schedulability** – not optimal
  - But very common is industry cases – e.g. automotive CAN

Online scheduling with fixed priorities

Schedulability tests

- As the schedule is built online it is fundamental to know a priori if a given task set is schedulable (i.e., its temporal requirements are met)
- There are two basic types of schedulability tests:
  - Based on CPU utilization rate
  - Based on response time

1. Preliminaries
2. Online scheduling with fixed priorities
3. **Schedulability tests based on utilization**
4. Response time analysis

RM Scheduling

Schedulability tests for RM based on task utilization

- Valid with preemption, n independent tasks, D=T
- Liu&Layland's (1973), **Least Upper Bound (LUB)**

$$U(n) = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n \cdot (2^{\frac{1}{n}} - 1) \Rightarrow \text{One execution per period guaranteed}$$

- Bini&Buttazzo&Buttazzo's (2001), **Hyperbolic Bound**

$$\prod_{i=1}^{n} (\frac{C_i}{T_i} + 1) \leq 2 \Rightarrow \text{One execution per period guaranteed}$$

RM Scheduling

Interpretation of the Liu&Layland test

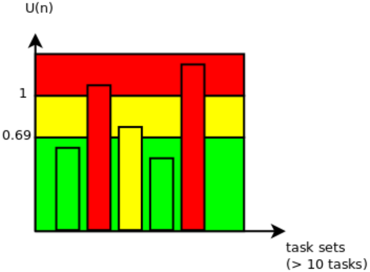
$U(n) > 1 \Rightarrow$ task set not schedulable (overload) – **necessary condition**

$U(n) \leq U_{lub} \Rightarrow$ task set is schedulable – **sufficient condition**

$1 \geq U(n) \geq U_{lub} \Rightarrow$ the test is **indeterminate**

Some U_{lub} values

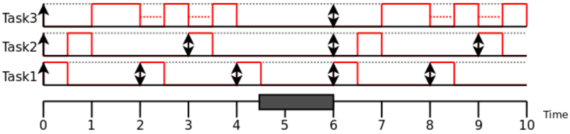
- $U_{lub}(1) = 1$
- $U_{lub}(2) = 0.83$
- $U_{lub}(3) = 0.78$
- ...
- $U_{lub}(n) \rightarrow ln(2) \approx 0.69$



RM Scheduling – example 1

Task properties

τ	C	T
1	0.5	2
2	0.5	3
3	2	6



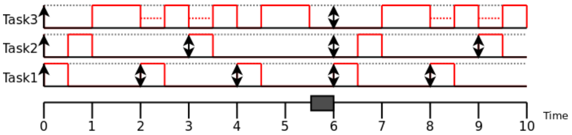
- $U = \frac{0.5}{2} + \frac{0.5}{3} + \frac{2}{6} \approx 0.75$
- $U_{lub}(3) = 0.78$. As $0.75 < 0.78$ one execution per period is guaranteed

RM Scheduling – example 2

Task properties

Same task set, but C_3 increases from 2 to 3 tu.

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

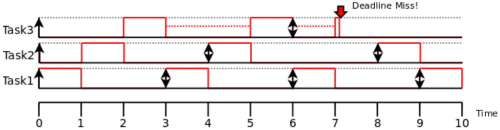


- $U = \frac{0.5}{2} + \frac{0.5}{3} + \frac{3}{6} \approx 0.92$
- $U_{lub}(3) = 0.78$. As $0.92 > 0.78$ one execution per period is **NOT guaranteed**
- But the task set is schedulable (see Gantt chart)

RM Scheduling – example 3

Task set properties

τ	C	T
1	1	3
2	1	4
3	2.1	6

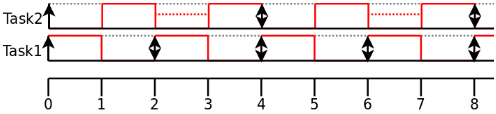


- $U = \frac{1}{3} + \frac{1}{4} + \frac{2.1}{6} \approx 0.93$
- $U_{lub}(3) = 0.78$. As $0.93 > 0.78$ one execution per period is **NOT guaranteed**
- And the task set indeed is **not schedulable** (see Gantt chart)

RM Scheduling – Harmonic Periods

Particular case: if the task **periods are harmonic** then the task set is schedlable iif $U(n) \leq 1$

- E.g. $\Gamma = \{(1,2);(2,4)\}$



Deadline Monotonic Scheduling DM)

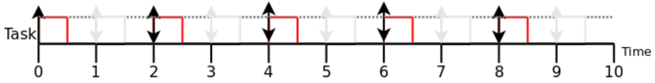
Schedulability tests for DM

- In some cases tasks may have large periods (low frequency) but require a short response time.
- In these cases we assign a deadline shorter than the period, and the scheduling criteria is the deadline.
- It is possible to use utilization-based tests.
  - The adaptation is simple, but the test is **very pessimistic** .

Utilization-based test

$$\sum_{i=1}^{n}\left(\frac{C_i}{D_i}\right) \leq n(2^{\frac{1}{n}} - 1)$$

E.g. $\{C = 0.5, T = 2, D = 1\}$ (Assumed utilization is doubled)



Response-time analysis

- For arbitrary fixed priorities, including RM, DM and others, the **Response Time Analysis** is an **exact** test (i.e., a necessary and sufficient condition) in the following conditions:
  - full preemption, synchronous release, independent tasks and $D \leq T$
- Worst-case response time (WCRT, Rwc, R,...) = maximum time interval between arrival and finish instants.
  - $Rwc_i = max_k(f_{i,k} - a_{i,k})$

Schedulability test based on the WCRT

$$Rwc_i < D_i, \forall i \Leftrightarrow \text{task set is schedulable}$$

Response-time analysis

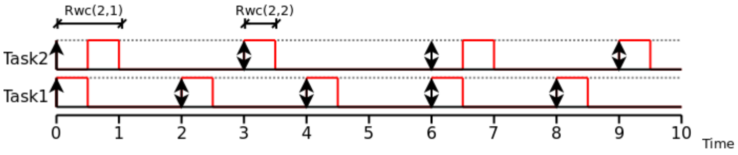
- The WCRT of a given task occurs when the task is activated at the same time as all other high-priority tasks (critical instant)
- I_i – interference caused by the execution of higher priority tasks

Computing Rwc_i

$$\forall i, Rwc_i = I_i + C_i$$

$$I_i = \sum_{k \in hp(i)} \left\lceil \frac{Rwc_i}{T_k} \right\rceil \cdot C_k$$

$\left\lceil \frac{Rwc_i}{T_k} \right\rceil$ represents the number of activations of hp task "k"



Response-time analysis

The equation is solved iteratively. Stop conditions are:

- A **deadline is violated**
  - $Rwc_i(m) > D_i$
- **Convergence** (two successive iterations yield the same result)
  - $Rwc_i(m+1) = Rwc_i(m)$

Algorithm

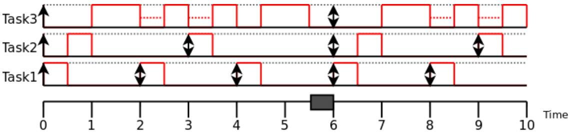
1. $Rwc_i(0) = \sum_{k \in hp(i)} (C_k) + C_i$
2. $Rwc_i(1) = \sum_{k \in hp(i)} \lceil \frac{Rwc_i(0)}{T_k} \rceil \cdot C_k + C_i$
3. ...
4. $Rwc_i(m) = \sum_{k \in hp(i)} \lceil \frac{Rwc_i(m-1)}{T_k} \rceil \cdot C_k + C_i$

Repeat Step 3 until convergence or deadline violation

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

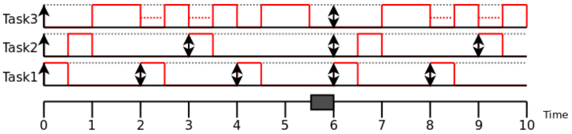


- $Rwc_1 = ?$

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

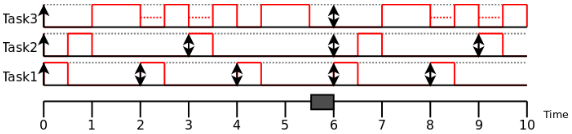


- $Rwc_1 =?$
  - $Rwc_1(0) = C_1 = 0.5tu$
- $Rwc_2 =?$

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

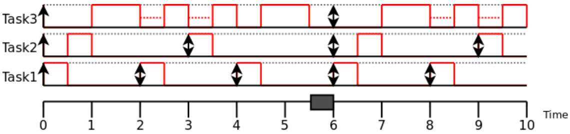


- $Rwc_1 = ?$
  - $Rwc_1(0) = C_1 = 0.5tu$
- $Rwc_2 = ?$
  - $Rwc_2(0) = C_1 + C_2 = 0.5 + 0.5 = 1tu$
  - $Rwc_2(1) = \sum_{k \in \{1\}} \lceil \frac{Rwc_i(0)}{T_k} \rceil \cdot C_k + C_2 = \lceil \frac{1}{2} \rceil \cdot 0.5 + 0.5 = 1tu$
  - **Converged** , thus $Rwc_2 = 1tu$

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

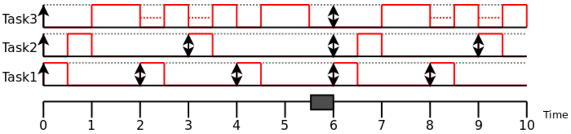


- Rwc_3=?

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

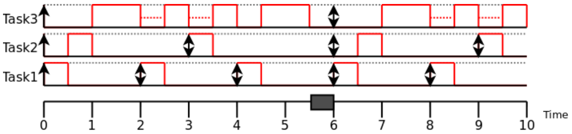


- $Rwc_3 = ?$
  - $Rwc_3(0) = C_1 + C_2 + C_3 = 4tu$

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

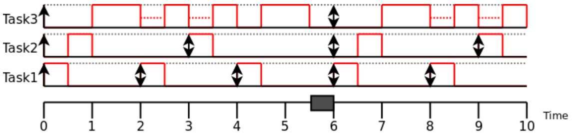


- $Rwc_3 = ?$
  - $Rwc_3(0) = C_1 + C_2 + C_3 = 4tu$
  - $Rwc_3(1) = \sum_{k \in \{1,2\}} \lceil \frac{Rwc_i(0)}{T_k} \rceil \cdot C_k + C_3 = \lceil \frac{4}{2} \rceil \cdot 0.5 + \lceil \frac{4}{3} \rceil \cdot 0.5 + 3 = 5tu$

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6

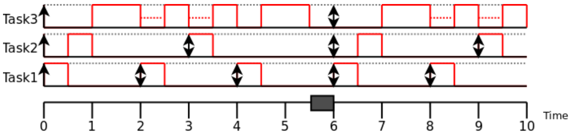


- Rwc_3=?
  - $Rwc_3(0)=C_1+C_2+C_3=4tu$
  - $Rwc_3(1)=\sum_{k\in\{1,2\}}\lceil\frac{Rwc_i(0)}{T_k}\rceil\cdot C_k+C_3=\lceil\frac{4}{2}\rceil\cdot 0.5+\lceil\frac{4}{3}\rceil\cdot 0.5+3=5tu$
  - $Rwc_3(2)=\sum_{k\in\{1,2\}}\lceil\frac{Rwc_i(1)}{T_k}\rceil\cdot C_k+C_3=\lceil\frac{5}{2}\rceil\cdot 0.5+\lceil\frac{5}{3}\rceil\cdot 0.5+3=5.5tu$

Response-time analysis

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6



- $Rwc_3 = ?$
  - $Rwc_3(0) = C_1 + C_2 + C_3 = 4tu$
  - $Rwc_3(1) = \sum_{k \in \{1,2\}} \lceil \frac{Rwc_i(0)}{T_k} \rceil \cdot C_k + C_3 = \lceil \frac{4}{2} \rceil \cdot 0.5 + \lceil \frac{4}{3} \rceil \cdot 0.5 + 3 = 5tu$
  - $Rwc_3(2) = \sum_{k \in \{1,2\}} \lceil \frac{Rwc_i(1)}{T_k} \rceil \cdot C_k + C_3 = \lceil \frac{5}{2} \rceil \cdot 0.5 + \lceil \frac{5}{3} \rceil \cdot 0.5 + 3 = 5.5tu$
  - $Rwc_3(3) = \sum_{k \in \{1,2\}} \lceil \frac{Rwc_i(2)}{T_k} \rceil \cdot C_k + C_3 = \lceil \frac{5.5}{2} \rceil \cdot 0.5 + \lceil \frac{5.5}{3} \rceil \cdot 0.5 + 3 = 5.5tu$
  - **Converged** , thus $Rwc_3 = 5.5tu$

As $Rwc(i) \leq D_i \forall i$, the task set is schedulable!

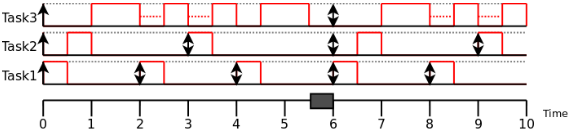
Restrictions to the schedulability tests previously presented

- The previous schedulability tests **must be modified** in the following cases:
  - Non-preemption
  - Tasks not independent
    - Share mutually exclusive resources
    - Have precedence constrains
  - It is also necessary to take into account the overhead of the kernel, because the scheduler, dispatcher and interrupts consume CPU time

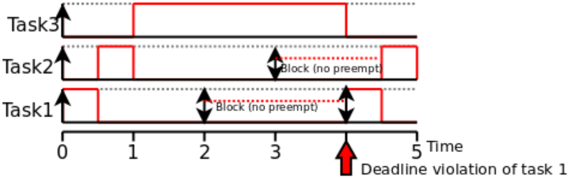
Impact of non-preemption

Example:

τ	C	T
1	0.5	2
2	0.5	3
3	3	6



With preemption



Without preemption

Summary

- On-line scheduling with fixed-priorities
- The Rate Monotonic scheduling policy – schedulability analysis based on utilization
- The Deadline Monotonic and arbitrary deadlines scheduling policies
- Response-time analysis