Real-Time Operating Systems

ROS01

Minor Embedded Systems

Week 6
Schedulability Analyses, Priority Assignment, RT Analyses (part 1)

versd@hr.nl
brojz@hr.nl

Planning ROS01

- Week 1: Introduction – Blinking leds
- Week 2: Super loop construct with an ISR
- Week 3: Cooperative Scheduling
- Week 4: Pre-emptive Scheduling
- Week 5: Using TI-RTOS
- **Week 6: Schedulability Analyses, Priority Assignment, Response Time Analyses (part 1)**
- Week 7: Response Time Analyses (part 2)
- Week 8: Finalizing Final Assignment

Free study material (…/minor Embedded Systems/ROS01/Books)

- Chapters 3 and 4 of: Ken Tindell and Hans Hansson, Real Time Systems by Fixed Priority Scheduling, Uppsala University, 1997.
- Chapter 12 of: Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition, MIT Press, ISBN 978-0-262-53381-2, 2017.

- Original papers:
  - C. L. Liu and J. W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, JACM, Volume 20, Number1, pages 46 to 61, 1973
  - M. Joseph and P. Pandya, Finding Response Times in a Real-Time System, The Computer Journal, Volume 29, Number 5, pages 390-395, 1986

Task scheduling

- In how many ways can you schedule 10 tasks (without preemption)?

- Choose one to start with (10 possibilities)
- Choose an other to go second (9 possibilities)
- …

- Total of 10x9x8x7x6x5x4x3x2x1 = 10! = 3628800 possible schedules

Scheduling tasks

- *N* tasks can be scheduled in *N*! different ways
  - For example 10 tasks: 3628800 possible schedules
  - With preemption there are many more possibilities
- The chosen schedule must meet all timing requirements
- A scheduling scheme (=plan) consists of:
  - An algorithm to find the "best" schedule
  - A method to predict the "worst-case" behavior of this schedule

Scheduling tasks… When do we do it?

- Static: the schedule is determined before the tasks are started
  - All task, their worst-case execution times and deadlines should be known beforehand
  - Using response time analysis it is possible to prove that all deadlines are met.
  - All response times are predictable!
  - Not able to react on “unforeseen” situations
- Dynamic: the schedule is determined when the tasks are running.
  - Behavior is less predictable
  - Can respond dynamically to unforeseen circumstances (e.g. a calculation that takes longer than expected)

Scheduling Real-Time systems

- Almost always a static scheduling method is used

- Most commonly used : Preemptive Priority Based scheduling
  - On each moment in time the ready task with the highest priority is running
  - Scheduling scheme:
    - An algorithm to assign a priority to each task
    - A method to predict the "worst-case" behavior of this schedule given the assigned priorities and to prove that all timing requirements are met

Scheduling - Simple model

- The number of task is known: N
- All tasks are periodical and all period times are known: T_i
- The tasks are independent from each other (no synchronization nor communication)
- System overhead is neglected
- The deadline of each task is equal to it's period time: $D_i = T_i$
- The worst-case execution time of each task is known: C_i

This model is **too** simple (but difficult enough).
Later we will look at realistic models.

Cyclic executive (Super loop)

- The schedule is determined upfront and is explicitly programmed.
- Example:

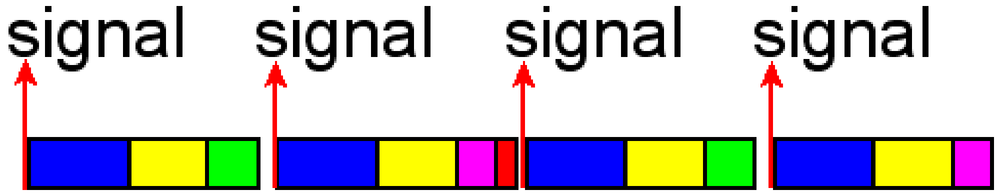
Taak	*T*	*C*
a	25	10
b	25	8
c	50	5
d	50	4
e	100	2

```
// Set timer to wake-up CPU every 25 ms
while (1) {
    sleep_until_wake_up(); a(); b(); c();
    sleep_until_wake_up(); a(); b(); d(); e();
    sleep_until_wake_up(); a(); b(); c();
    sleep_until_wake_up(); a(); b(); d();
}
```

- How to determine the schedulability?
- How to find a schedule?

Cyclic executive (Super loop)

signal = systick



- a T = 25 C = 10
- b T = 25 C = 8
- c T = 50 C = 5
- d T = 50 C = 4
- e T = 100 C = 2

minor cycle = 25 ns
major cycle = 100 ns

Utilization

U = 10/25+8/25+5/50+4/50+2/100
= **0,92**

What is the maximum utilization in the general case?

If C_e = 4 then U = **0,94** and the task set is **not** schedulable!

- Minor cycle = gcd(T_1, T_2, ... T_n). Major cycle = lcm(T_1, T_2, ... T_n).
- How to determine the schedulability?
- How to find a schedule?

Cyclic executive (Super loop)

- Characteristics:
  - There are no real tasks, only ordinary functions
  - Shared memory can be used for communication without protection (mutex is not needed)
  - All T's should be a multiple of the minor cycle time
  - System is deterministic (predictable)
- Issues:
  - Tasks with large differences in T's result in a large major cycle
  - Sporadic tasks (interrupts) can not be included!
  - Poorly maintainable, adaptable and expandable
  - Determining the schedule is NP-hard! (read: very, very hard)
- Alternatives:
  - **Fixed-Priority Scheduling** (FPS)
  - Earliest Deadline First (EDF)

FPS Fixed-priority Pre-emptive Scheduling

- Each task runs with a statically determined fixed priority
- This priority is determined by the timing requirements of all tasks
- The scheduling is preemptive:
  - When a task with a higher priority becomes ready, the running task will be preempted (interrupted)

RMPA = Rate Monotonic Priority Assignment

- The period time of a task determines the priority of that task
- The shorter the period time the higher the priority

$$T_i < T_j \Rightarrow P_i > P_j$$

- This (simple) method is optimal!
  - if some fixed-priority pre-emptive schedule exists, then, the rate monotonic fixed-priority pre-emptive schedule is also feasible

FPS-RMPA

- Utilization based schedulability test:

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N(2^{1/N} - 1)$$

- If this test is true then no deadlines are missed!
- If this test is false then maybe some deadlines are missed!

N	Test
1	$U \leq 1.000$
2	$U \leq 0.828$
3	$U \leq 0.780$
4	$U \leq 0.757$
5	$U \leq 0.743$
10	$U \leq 0.718$
infinite	$U \leq 0.693$

FPS-RMPA

- Utilization based schedulability test for $N \rightarrow \infty$:

$$U \equiv \sum_{i=1}^{\infty} \frac{C_i}{T_i} \leq \lim_{N \to \infty} N(2^{1/N} - 1) = \lim_{N \to \infty} \frac{2^{1/N} - 1}{1/N}$$

$$U \leq \lim_{M \to 0} \frac{2^M - 1}{M} = \frac{0}{0} \quad \text{use L'Hôpital's rule}$$

$$U \leq \lim_{M \to 0} \frac{\ln 2}{1} = 0.693$$

L'Hôpital's rule

FPS-RMPA Schedulability examples

- Possibilities:
  - Does not meet the test and some deadlines are not met
  - Does not meet the test but all deadlines are met
  - Does meet the test and all deadlines are met

- Meeting the test is sufficient evidence that all deadlines are met. But it is not necessary to satisfy the test in order to meet all deadlines.

FPS-RMPA Response time analysis

- In contrast to the utilization test, this analysis determines the exact response times. So we can say exactly whether all deadlines are met (and by what margin).

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

R_i appears on the **left** and the **right** side of the equation. This equation can not be simply solved. (because the ceiling function is not invertible)

R_i is the response time of task i
$hp(i)$ is the set of tasks with a higher priority than task i

FPS-RMPA Response time analysis

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- The response time of the highest priority task is: $R = C$
- All other tasks can be preempted. Their response time is: $R_i = C_i + I_i$
- Where I_i is the maximum "interference" time. This will occur when all tasks with a higher priority than $\boldsymbol{i}$ start at the same time as task $\boldsymbol{i}$
- The number of times task $\boldsymbol{j}$ with a higher priority than $\boldsymbol{i}$ can preempt task $\boldsymbol{i}$ is given by:

$$\text{Number of Releases} = \left\lceil \frac{R_i}{T_j} \right\rceil$$

- So $I_{i,j}$ equals: $\left\lceil \frac{R_i}{T_j} \right\rceil C_j$

FPS-RMPA Response time analysis

- The total maximum interference time is the sum of the maximum interference time of every task with a higher priority:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

- Which can be solved by using a recurrence relation:

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

Start with $w_i^0 = 0$ and continue until:

$w_i^n = w_i^{n+1}$ ✔ or $w_i^{n+1} > T_i$ ✘

FPS-RMPA Response time analysis

- Further extension of the analysis method is necessary to include:
  - Sporadic tasks
  - Tasks with $D < T$
  - Interaction between tasks
  - Release jitter
  - Tasks with $D > T$
  - Release offsets

FPS-RMPA $D < T$ and Sporadic tasks

- $D < T$:
  - Use DMPA instead of RMPA:

$$D_i < D_j \Rightarrow P_i > P_j$$

  - Use the following stop condition in the response time analysis:

$$W_i^{n+1} > D_i$$

- Sporadic tasks (interrupts):
  - Use the minimum time between two "starts" of this task as the period time
    T = minimum inter-arrival interval
  - For most sporadic tasks $D < T$

Homework assignment. ../ros01/Opdrachten/Huiswerk_Week_6.pdf

Een programma bestaat uit 4 taken T_1 t/m T_4. Deze taken gebruiken geen gedeelde resources. In de tabel 1 staat i voor het nummer van de taak, T_i voor de periodetijd van taak i en C_i voor de maximale executietijd van taak i. Gegeven is dat de deadline van elke taak gelijk is aan zijn periodetijd.

Tabel 1: De gegevens van de taken

i	T_i	C_i
1	100	50
2	280	45
3	200	20
4	300	40

Alle gegeven tijden zijn in ms.

A Bepaal de schedulability van deze taken met behulp van de "Utilization based schedulability test". Geef de benodigde berekening en trek daaruit je conclusie!

B Bepaal de prioriteiten P_i van de verschillende taken als gebruik gemaakt wordt van FPS-RMPA (Fixed-priority Pre-emptive Scheduling – Rate Monotonic Priority Assignment). Het systeem kent 4 verschillende prioriteiten (1 t/m 4) waarbij 4 de hoogste prioriteit is.

C Bereken voor alle taken of de deadline wordt gehaald en geef, indien de deadline wordt gehaald, de response tijd R_i.