Scheduling

"Dienstregeling" van taken

- N taken kun je in N! verschillende schedules uitvoeren.
  - Bijvoorbeeld 10 taken: 3628800 mogelijke schedules.
  - Met preemption nog veel meer mogelijkheden.
- De gekozen scheduling moet aan alle time requirements voldoen.
- Een scheduling scheme (=plan) bestaat uit:
  - Een algoritme om een schedule te bepalen.
  - Een methode om het "worst-case" gedrag van een met het algoritme bepaalde schedule te voorspellen.

THRijswijk

Scheduling

Bepaal volgorde van taken

- Statisch: schedule ligt vast voor uitvoering.
  - taken, worst-case executietijden en deadlines moeten van tevoren bekend zijn.
  - Je kunt door middel van een analyse aantonen dat alle deadlines worden gehaald.
  - Response tijden zijn voorspelbaar!
  - Kan niet reageren op "onvoorziene" situaties.
- Dynamisch: schedule bepaald tijdens uitvoeren.
  - Gedrag minder goed vooraf voorspelbaar.
  - Kan dynamisch inspelen op onvoorziene omstandigheden (b.v. berekening die langer duurt dan verwacht).

THRijswijk

Scheduling

Real-time systemen

- Bijna altijd wordt een statische methode gebruikt.
- Meest gebruikt: Preemptive Priority Based scheduling
  - op elk moment runt de ready taak met de hoogste prio
  - In dit geval bestaat een scheduling scheme uit:
    - Een algoritme om de prioriteit van elke taak te bepalen.
    - Een methode om de schedulability te voorspellen = Een methode om het "worst-case" gedrag bij de gekozen prioriteiten te voorspellen en te bekijken of aan alle time requirements wordt voldaan.

THRijswijk

Scheduling

Een simpel process model.

- Het aantal taken is bekend. N
- Alle taken zijn periodiek en de perioden zijn bekend. T_i
- De taken zijn onafhankelijk van elkaar (geen synchronisatie en communicatie).
- Systeem overhead wordt verwaarloosd.
- De deadline van elke taak is gelijk aan de periode. $D_i=T_i$
- De worst-case executietijd van elke taak is vast en bekend. C_i

Dit process model is te simpel (maar al moeilijk genoeg). Later zullen we realistische modellen bekijken.

THRijswijk

Scheduling

Cyclic executive approach

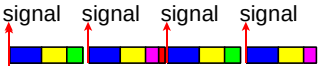
- Als schedule van tevoren bepaald is kun je het uitvoeren van de schedule expliciet programmeren.
- Voorbeeld:

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

```
// zet timer: elke 25 ms een signal
while (1) {
   sigwait(&set, &signum); a(); b(); c();
   sigwait(&set, &signum); a(); b(); d(); e();
   sigwait(&set, &signum); a(); b(); c();
   sigwait(&set, &signum); a(); b(); d();
}
```

THRijswijk

Cyclic executive



- 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

Hoe bepaal je de schedulability?
Hoe vind je een schedule?

Utilization (benutting) U = (4*10+4*8+2*5+2*4+1*2)/100 = 0,92
Als C_e = 4 dan geldt U = 0,94 maar is geen schedule mogelijk!

THRijswijk

Cyclic executive

- Eigenschappen:
  - Er zijn geen processen (of threads) de taken zijn gewone functies.
  - Er is shared memory voor communicatie. Protectie is niet nodig.
  - Alle T's moeten een veelvoud zijn van de minor cycle time.
  - Systeem is deterministisch.
- Problemen:
  - Taken met grote verschillen in T's geeft lange major cycle.
  - Sporadic taken zijn niet in te passen!
  - Slecht onderhoudbaar, aanpasbaar en uitbreidbaar.
  - Schedule bepalen is moeilijk!
- Alternatieven:
  - **Fixed-Priority Scheduling** (FPS)
  - Earliest Deadline First (EDF)

THRijswijk

FPS

Fixed-Priority Scheduling

- Elke taak draait als thread (of process) met een statisch bepaalde vaste prioriteit.
- De prioriteit wordt bepaald door de time requirements.
- Preemptive <=> NonPreemptive
  - **Preemptive**: Als thread met hogere prio ready wordt dan wordt running thread meteen onderbroken.
  - NonPreemptive: Als thread eenmaal gestart is wordt hij ook afgemaakt.
  - Deferred Preemption. Als thread met hogere prio ready wordt dan wordt running thread na een bepaalde tijd zeker onderbroken.

THRijswijk

FPS

Rate monotonic priority assignment

- De periode van een proces bepaalt de prioriteit van dat proces. Hoe korter de periode hoe hoger de prioriteit.
- $T_i < T_j \Rightarrow P_i > P_j$
- Deze methode is optimaal!
  - Dat wil zeggen dat als er een mogelijke preemptive FPS schedule is dan is rate monotonic FPS ook mogelijk.

THRijswijk

FPS

Rate monotonic priority assignment

- Utilization based schedulability test:

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

- Als deze test true geeft is worden geen deadlines gemist!

N	Test
1	<= 1.000
2	<= 0.828
3	<= 0.780
4	<= 0.757
5	<= 0.743
10	<= 0.718
oneindig	<= 0.693

THRijswijk

FPS-RMPA

Schedulability voorbeelden

- Boek:
  - Voorbeeld A: Tabel 13.5, figuur 13.2 (time line) en figuur 13.5 (Grantt chart). Voldoet niet aan de test en haalt deadlines niet.
  - Voorbeeld B: Tabel 13.6. Voldoet wel aan de test en haalt deadlines wel.
  - Voorbeeld C: Tabel 13.7 en figuur 13.4. Voldoet niet aan de test maar haalt deadlines wel.
- Als aan de test voldaan wordt is dat **voldoende** bewijs dat de deadlines gehaald worden. Maar het is niet **noodzakelijk** dat aan de test voldaan wordt om de deadlines te kunnen halen.

THRijswijk

FPS-RMPA

Response tijd analyse

- In tegenstelling tot de vorige test geeft deze analyse de **exacte** response tijden. We kunnen dus exact zeggen of alle deadlines gehaald worden (en met welke marge).

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

- R_i is de response tijd van taak i
- $hp(i)$ is de verzameling taken met een hogere prioriteit dan taak i
- R_i komt zowel links als rechts voor. De vergelijking is niet eenvoudig op te lossen (door de ceiling functie)

THRijswijk

FPS-RMPA

Response tijd analyse

- Voor de taak met de hoogste prioriteit geldt: $R = C$.
- Alle andere taken kunnen onderbroken worden en voor deze taken geldt: $R_i = C_i + I_i$
- I_i is de maximale "interference". Deze treed op als alle taken met een hogere prioriteit gelijk met taak i starten.
- Het aantal keer dat een taak met een hogere prioriteit j gestart wordt voordat i is afgelopen is:

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

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

THRijswijk

FPS-RMPA

Response tijd analyse

- De totale maximale interference is de som van de max interference van alle taken met een hogere prioriteit:

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

- Oplossing met behulp van recurrente betrekking:

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

Start met $w_i^0 = 0$ en ga door tot

$w_i^n = w_i^{n+1}$ ✔ of $_{n+1}$ ✘

THRijswijk

FPS-RMPA

Response tijd analyse

- Je kunt een programma schrijven om alle R_i's te berekenen: zie boek p. 476.
- Voorbeeld: zie boek p. 477.
- Verdere uitbreiding is nog nodig:
  - Taken met $D<T$.
  - Sporadic taken.
  - Interactie tussen taken.
  - Deferred preemption.
  - Release jitter.
  - Taken met $D>T$.
  - Fault tolerance.
  - Release offset.

THRijswijk