

Scheduling

Josef Mueller, Isabella Schoen
Gruppe1

Schedulingverfahren

Beschreibung

Ein Realzeitrechner bearbeite vier Tasks, deren Verarbeitungszeiten für diesen Rechner und deren Prozesszeiten gegeben sind:

Taskname	te in ms	tp in ms
A	50	200
B	75	300
C	100	500
D	125	800

Aufgaben

- ✓ 1. Ist der Rechner von den Zeitanforderungen (Auslastungsbedingung) prinzipiell in der Lage, die Aufgaben zu bearbeiten?

$$\rho_{ges, max} = \sum_{j=1}^n \frac{t_{E_{max,i}}}{t_{P_{min,i}}} \leq c$$

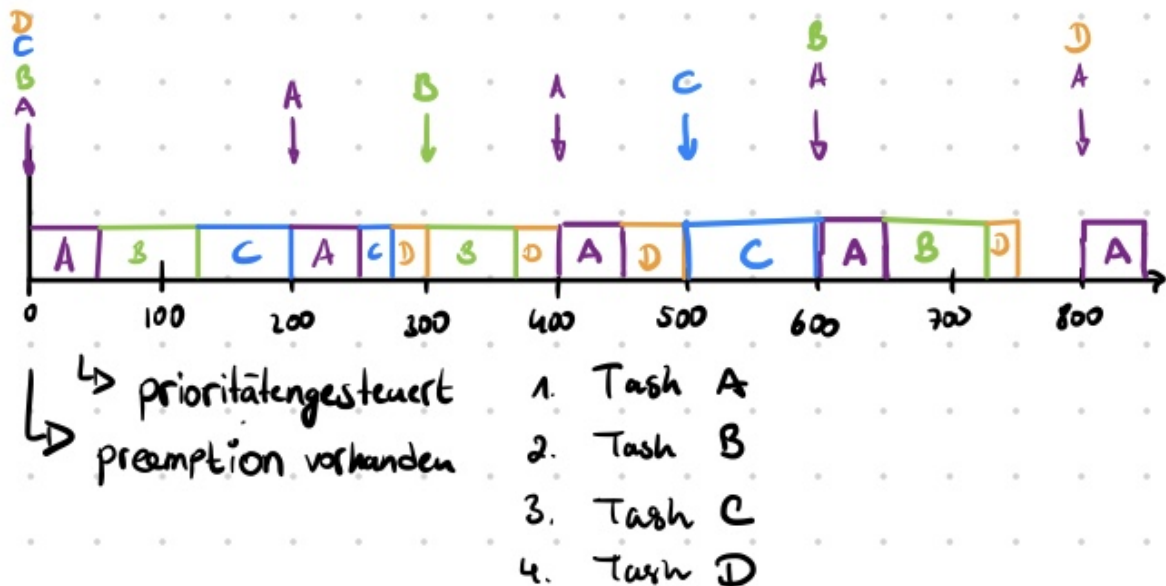
$$= \left(\frac{50}{200} \right) + \left(\frac{75}{300} \right) + \left(\frac{100}{500} \right) + \left(\frac{125}{800} \right)$$

Task A Task B Task C Task D

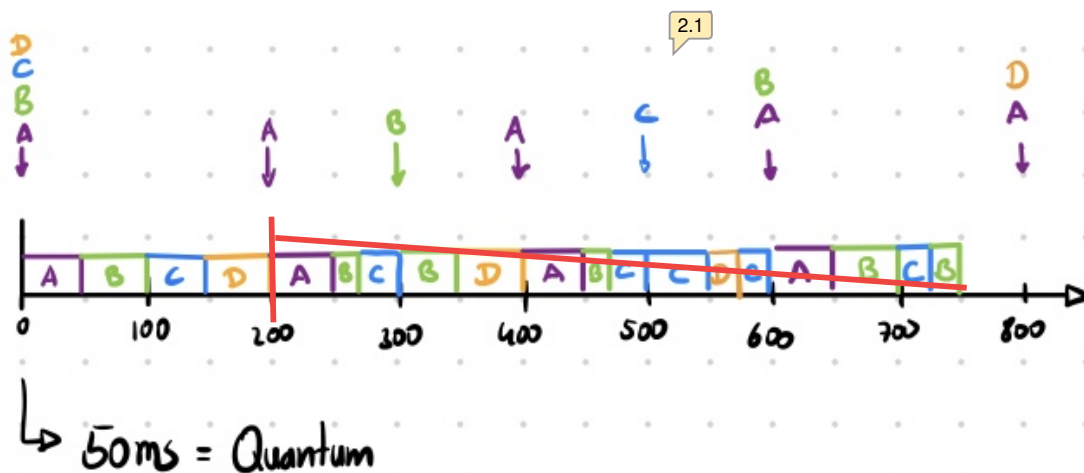
$$= 0,25 + 0,25 + 0,2 + 0,1562$$

$$= 0,8562 = 85,62\% \leq 100\% \quad (\checkmark)$$

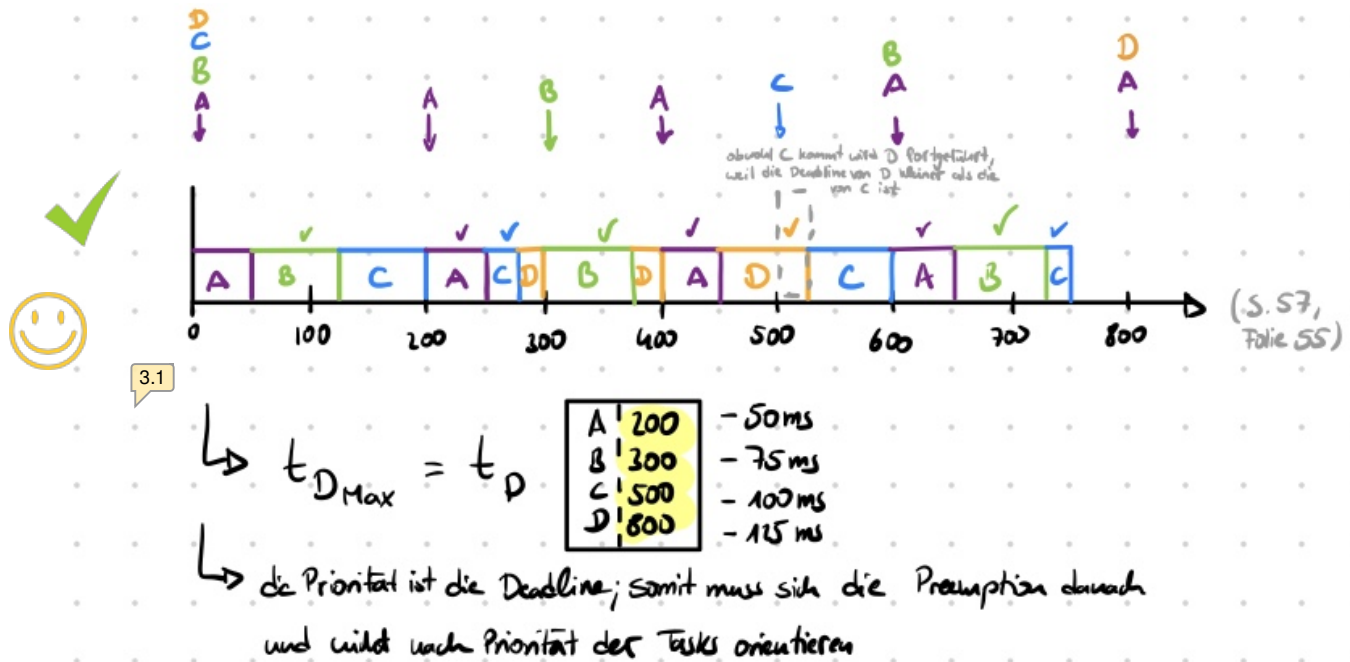
- ✓ 2. Zunächst werde ein prioritätengesteuertes Scheduling verwendet. Welche Task bekommt dabei welche Priorität? Geben Sie die Rechnerkernbelegung durch die vier Tasks im Intervall [0,800] an.



3. Als zweites soll ein Round-Robin Scheduling untersucht werden. Ein Quantum sei 50 ms. Geben Sie die Rechnerkernbelegung durch die vier Tasks beim Round-Robin Scheduling im Intervall $[0,800]$ an.
 Hinweis: Erstellen und verfolgen Sie ständig die Liste (Queue) der rechenbereiten Rechenprozesse!



4. Jetzt werde ein Deadline-Scheduling verwendet. Die maximal zulässigen Reaktionszeiten seien dabei durch die Prozeßzeiten gegeben ($t_{Dmax}=t_p$). Geben Sie auch für diesen Fall die Rechnerkernbelegung durch die vier Tasks im Intervall $[0,800]$ an. Hinweis: Nummerieren Sie in der Anforderungsfunktion die einzelnen Anforderungen durch und zeichnen Sie zusätzlich die Deadlines ein.



Zeitgesteuertes Scheduling mit konstanter Framesize

Beschreibung

Jedes der in den Aufgaben angegebenen Systeme mit periodischen Tasks wird entsprechend dem **zyklischen Scheduling** bearbeitet.

Aufgaben

1. Berechnen Sie für jedes System die möglichen Framegrößen. Unterbrechungen der Tasks sind nur erlaubt, wenn ansonsten kein Schedulingplan gefunden werden kann.

Es handelt sich um ein statisches Verfahren, einem zeitgesteuerten Scheduling-Plan mit konstanten Zeitslots. (ab S.47)

System 1

Task	tPmin	tEmax
1	6	1
2	10	2
3	18	2

Hyperperiode: $\text{kgV}(6, 10, 18) = 90 \text{ [ms]}$


```
f: 3 tPmin: 18 tEmax: 2 tDmax: 18 OK
3 is OK
f: 5 tPmin: 6 tEmax: 1 tDmax: 6
f: 5 tPmin: 10 tEmax: 2 tDmax: 10 OK
f: 6 tPmin: 6 tEmax: 1 tDmax: 6 OK
f: 6 tPmin: 10 tEmax: 2 tDmax: 10 OK
f: 6 tPmin: 18 tEmax: 2 tDmax: 18 OK
6 is OK
```

A: Hier eignen sich Framesizes von 2 ms, 3 ms und 6 ms.

System 2



Task	tPhase	tPmin	tEmax	tDmax
1	7	5	1	5
2	0	9	1	9
3	0	12	3	12
4	0.5	23	5	21
5	0.5	23	2	21

Hyperperiode: $\text{kgV}(5, 9, 12, 23) + 7 \text{ [ms]} = 4140 \text{ [ms]}$

Bei der Berechnung der Hyperperiode muss zusätzlich die maximale Phase aus allen Tasks addiert werden. (S. 232 oben)

$(7, 5; 1, 5)$
 $(9, 1)$
 $(12, 3)$
 $(0, 5; 23, 7; 21)$

f wäre $\geq \max(t_{E_{\max}, i})$
 aber sollte aufgeteilt werden: in bspw. $5ms + 2ms$
 da $7 > \min(t_{D_{\max}, i})$ und das nicht erlaubt ist

$(t_{p_i} + t_{p_{\min}} + t_{E_{\max}, i} + t_{D_{\max}, i})$
 $(t_{p_{\min}} + t_{E_{\max}})$

↓
5

1. Bedingung: $f \geq \max(\text{alle } E_{\max})$

$$f \geq \max(1, 2, 3, 5)$$

2. Bedingung: $f \nmid t_{p_{\min}} = 0$

$$f \in \{5, 9, 12, 23\}$$

3. Bedingung: $2 \cdot f - \gcd(t_{p_{\min}, i}, f) \leq t_{D_{\max}, i}$

Siehe Output vom Code

Output vom Code:


```

f: 5 tPmin: 5 tEmax: 1 tDmax: 5 OK
f: 5 tPmin: 9 tEmax: 1 tDmax: 9 OK
f: 5 tPmin: 12 tEmax: 3 tDmax: 12 OK
f: 5 tPmin: 23 tEmax: 4 tDmax: 21 OK
f: 5 tPmin: 23 tEmax: 3 tDmax: 21 OK
5 is OK
f: 9 tPmin: 5 tEmax: 1 tDmax: 5
f: 9 tPmin: 9 tEmax: 1 tDmax: 9 OK
f: 9 tPmin: 12 tEmax: 3 tDmax: 12
f: 9 tPmin: 23 tEmax: 4 tDmax: 21 OK
f: 9 tPmin: 23 tEmax: 3 tDmax: 21 OK
f: 12 tPmin: 5 tEmax: 1 tDmax: 5
f: 12 tPmin: 9 tEmax: 1 tDmax: 9
f: 12 tPmin: 12 tEmax: 3 tDmax: 12 OK
f: 12 tPmin: 23 tEmax: 4 tDmax: 21
f: 12 tPmin: 23 tEmax: 3 tDmax: 21
f: 23 tPmin: 5 tEmax: 1 tDmax: 5
  
```

```
f: 23 tPmin: 9 tEmax: 1 tDmax: 9
f: 23 tPmin: 12 tEmax: 3 tDmax: 12
f: 23 tPmin: 23 tEmax: 4 tDmax: 21
f: 23 tPmin: 23 tEmax: 3 tDmax: 21
```

A: Hier eignet sich eine Framesize von 5 ms.

Code zur Berechnung der Framesizes



```
import math as m

def checkCondition(f: int, t_Pmin: int, t_Dmax: int):
    return 2 * f - m.gcd(t_Pmin, f) <= t_Dmax

# tasks = [(6, 1), (10, 2), (18, 2)]
tasks = [(7, 5, 1, 5), (9, 1), (12, 3), (0.5, 23, 5, 21), (0.5, 23, 2, 21)]

# possibleFs = [2, 3, 5, 6]
possibleFs = [5, 9, 12, 23]

for f in possibleFs:
    workingF = True
    for task in tasks:
        tPhase = -1
        tPmin = -1
        tEmax = -1
        tDmax = -1
        if len(task) == 2:
            tPmin = task[0]
            tEmax = task[1]
        elif len(task) == 4:
            tPhase = task[0]
            tPmin = task[1]
            tEmax = task[2]
            tDmax = task[3]
        if tDmax == -1:
            tDmax = tPmin
        combination = "f: " + str(f) + " tPmin: " + str(tPmin) + " tEmax: " +
            str(tEmax) + " tDmax: " + str(tDmax)
        if checkCondition(f, tPmin, tDmax):
            print(combination + " OK")
        else:
            print(combination)
            workingF = False
    if workingF:
        print(str(f) + " is OK")
```

Index der Kommentare

- 2.1 Bei $T=200$ wird die wieder rechenbereite Task A hinten in die Queue eingefügt. Bei $T=200$ ist also die Task B mit ihren verbleibenden 25ms an der Reihe. Dadurch verschiebt sich das gesamte Scheduling.
- 3.1 Spitzen Darstellung!!