– Lösung zur Praktikumsaufgabe 8 –

Thema: *Ressourcen im ratenmonotonen Scheduling*

1.

2. a)

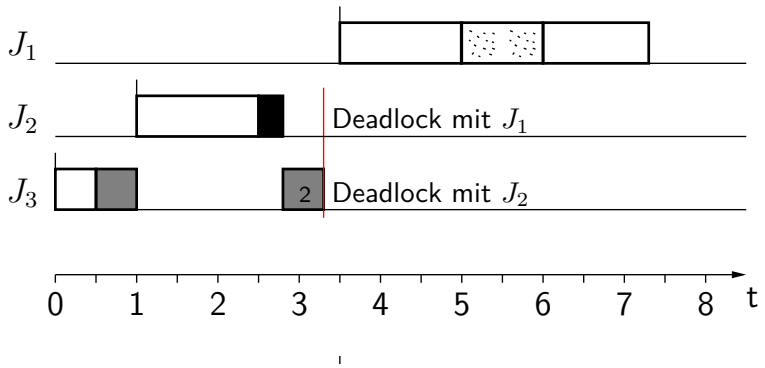

Abbildung 1: Reihenfolge der Jobaktivierungen für Prioritätsvererbung

b)

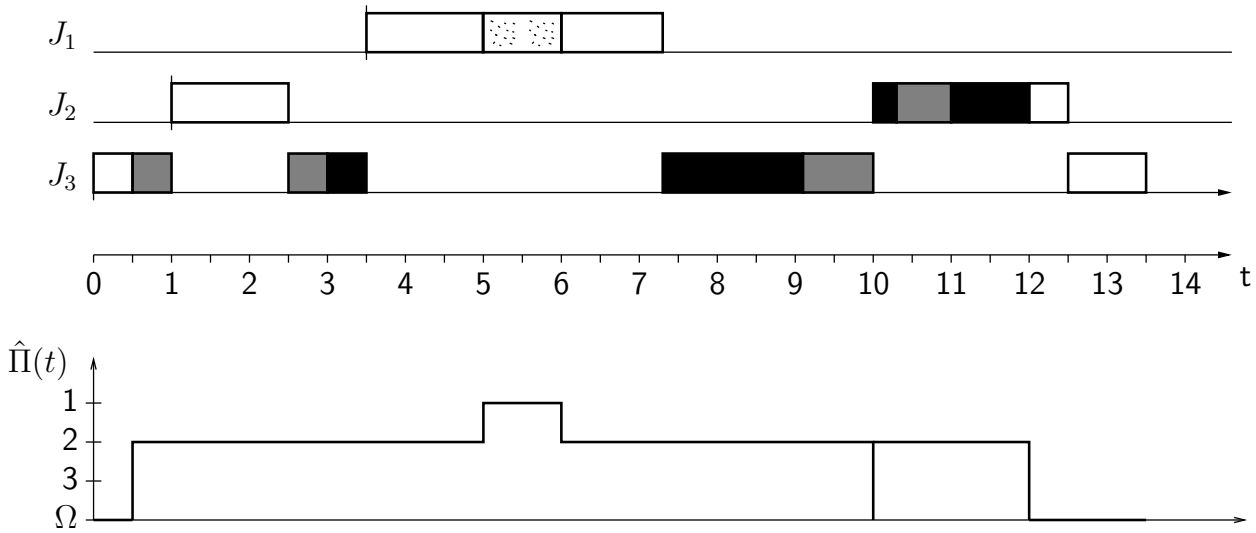

Abbildung 2: Reihenfolge der Jobaktivierungen für Basic Priority Ceiling

c)

Die Komplettierungszeiten der Jobs sind für Basic PCP und Stack-based Priority Ceiling identisch; letzteres verursacht einen Kontextwechsel weniger.

3. Es ist günstig, die drei verschiedenen Blockierungszeiten (vgl. Vorlesung) tabellenartig auszuwerten.

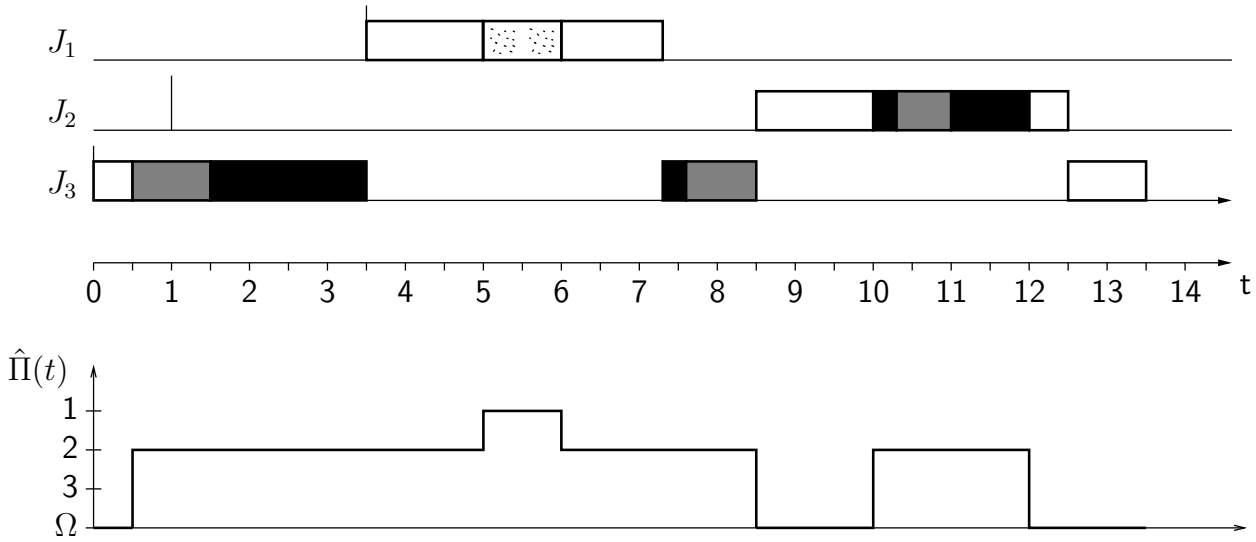

Abbildung 3: Reihenfolge der Jobaktivierungen für Stack-based Priority Ceiling

Die erste Spalte führt die blockierten Jobs auf, die erste Zeile die Verursacher der Blockierung (die *blockierenden* Jobs). Beispielsweise wird J_1 für maximal 6 Zeiteinheiten durch J_3 und maximal 2 Zeiteinheiten durch J_6 blockiert.

a) direkte Blockierung:

	verursacht Blockierung					
blockiert	J_1	J_2	J_3	J_4	J_5	J_6
J_1	*	0	6	0	0	2
J_2	–	*	0	5	0	0
J_3	–	–	*	0	0	4
J_4	–	–	–	*	0	0
J_5	–	–	–	–	*	0
J_6	–	–	–	–	–	*

Tabelle 1: Direkte Blockierungen in Aufgabe 3

Da per definitionem nur ein niedrigpriorisierten Job einen höherpriorisierten Job blockieren kann (und nicht umgekehrt), ist der Platz unterhalb der Hauptdiagonalen stets leer. Aus dem gleichen Grund kann der niedrigstpriorisierte Job (hier: J_6) nicht blockiert werden und der höchstpriorisierte Job (hier: J_1) einen anderen Job niemals blockieren (er *verdrängt* nur).

Ein Job, der keine Ressource anfordert, kann nicht direkt blockiert werden (hier: J_5).

Wenn zwei Jobs bezüglich mehrerer Ressourcen Konflikte haben (hier: J_3 und J_6, bezüglich X und Z), ist das Maximum der jeweiligen Blockierungszeit maßgebend (da nur für maximal einen kritischen Abschnitt blockiert werden kann).

b) indirekte Blockierung durch Prioritätsvererbung:

	verursacht Blockierung					
blockiert	J_1	J_2	J_3	J_4	J_5	J_6
J_1	*	–	–	–	–	–
J_2	–	*	6	0	0	2
J_3	–	–	*	5	0	2
J_4	–	–	–	*	0	4
J_5	–	–	–	–	*	4
J_6	–	–	–	–	–	*

Tabelle 2: Indirekte Blockierungen in Aufgabe 3

- höchstpriorisierter Job kann nicht indirekt blockieren
- Eintrag in Zeile i, Spalte $k :=$ max(Spalte k, Reihe $1 \ldots i-1$) der Tabelle der direkten Blockierung
- Beispiel: J_6 könnte J_2, J_3, J_4 und J_5 für 2 Einheiten indirekt blockieren, da er dann u.U. die Priorität 1 von J_1 geerbt hat. J_4 und J_5 könnte er sogar für 4 Zeiteinheiten blockieren, da er maximal 4 Zeiteinheiten Priorität 3 erbt (Konkurrenz um Z).

c) "avoidance blocking":
Satz: Wenn alle Prioritäten verschieden sind, dann sind die Blockierungszeiten für *avoidance blocking* im Worst Case identisch zu denen der indirekten Blockierung, mit der Ausnahme aller der Jobs, die keine Ressource fordern (deren *avoidance blocking*-Zeit wird 0). [Jane Liu: *Real-Time Systems*, S. 298]

Die Blockierungszeit eines Jobs ergibt sich aus dem Maximum aller Blockierungszeiten in allen drei Tabellen fuer den betreffenden Job. Also:

$$t_{b,1} = t_{b,2} = 6 \quad t_{b,3} = 5 \quad t_{b,4} = t_{b,5} = 4 \quad t_{b,6} = 0$$

Zur Ermittlung der Blockierungszeiten ist das Wissen, *wann* eine Ressource blockiert, irrelevant.

4. Eine grafische Veranschaulichung der Ressourcenforderungen zeigt Abbildung 4.

a) $t_{b,1} = t_{b,2} = t_{b,3} = 5; \quad t_{b,4} = 0$

Bereits T_1 verletzt im Worst Case seine Deadline: $t_{maxresp,1} = t_{e,1} + t_{b,1} = 3 + 5 = 8 > t_{d,1}$

b) Aus Abbildung 4 können die Blockierungszeiten für Priority Ceiling unmittelbar abgelesen werden:

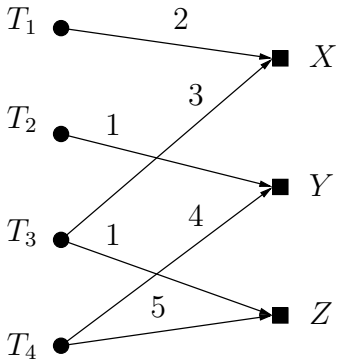

Abbildung 4: Ressourcenforderungen der Tasks in Aufgabe 4

$t_{b,1} = 3, \quad t_{b,2} = 4, \quad t_{b,3} = 5, \quad t_{b,4} = 0$

Für $T_1 \ldots T_4$ muß nun eine ratenmonotone Analyse ausgeführt werden:

T_1**:**

$$t_{maxresp,1} = t_{e,1} + t_{b,1} = 3 + 3 = 6 = t_{d,1} \quad \surd$$

T_2**:**

$$\begin{aligned} t_{rsp,2}^{(l+1)} &= t_{e,2} + t_{b,2} + \left\lceil \frac{t_{rsp,2}^{(l)}}{t_{p,1}} \right\rceil \cdot t_{e,1} \\ &= 9 + \left\lceil \frac{t_{rsp,2}^{(l)}}{6} \right\rceil 3 \end{aligned}$$

Iteration von $t_{rsp,2}^{(i)}$ ergibt $9 \to 15 \to 18 \to 18$.

$$t_{maxresp,2} = 18 < t_{d,2} = 20 \quad \surd$$

T_3**:**

$$\begin{aligned} t_{rsp,3}^{(l+1)} &= t_{e,3} + t_{b,3} + \left\lceil \frac{t_{rsp,3}^{(l)}}{t_{p,2}} \right\rceil \cdot t_{e,2} + \left\lceil \frac{t_{rsp,3}^{(l)}}{t_{p,1}} \right\rceil \cdot t_{e,1} \\ &= 10 + \left\lceil \frac{t_{rsp,3}^{(l)}}{20} \right\rceil 5 + \left\lceil \frac{t_{rsp,3}^{(l)}}{6} \right\rceil 3 \end{aligned}$$

Iteration von $t_{rsp,3}^{(i)}$ ergibt $10 \to 21 \to 32 \to 38 \to 41 \to 46 \to 49 \to 52 \to 52$.

$$t_{maxresp,3} = 52 < t_{d,3} = 200 \quad \surd$$

T_4**:**

$$\begin{aligned} t_{rsp,4}^{(l+1)} &= t_{e,4} + t_{b,4} + \left\lceil \frac{t_{rsp,4}^{(l)}}{t_{p,3}} \right\rceil \cdot t_{e,3} + \left\lceil \frac{t_{rsp,4}^{(l)}}{t_{p,2}} \right\rceil \cdot t_{e,2} + \left\lceil \frac{t_{rsp,4}^{(l)}}{t_{p,1}} \right\rceil \cdot t_{e,1} \\ &= 6 + \left\lceil \frac{t_{rsp,4}^{(l)}}{200} \right\rceil 5 + \left\lceil \frac{t_{rsp,4}^{(l)}}{20} \right\rceil 5 + \left\lceil \frac{t_{rsp,4}^{(l)}}{6} \right\rceil 3 \end{aligned}$$

Iteration von $t_{rsp,4}^{(i)}$ ergibt $6 \to 19 \to 28 \to 36 \to 39 \to 42 \to 47 \to 50 \to 53 \to 53$.

$$t_{maxresp,4} = 53 < t_{d,3} = 210 \quad \surd$$

$\rightsquigarrow$ Die Taskmenge ist unter Priority Ceiling (geradeso; im Worst Case erreicht T_1 seine Deadline) planbar, unter NPCS hingegen nicht.