Sistemi Embedded e Real-time (M. Cesati)

Compito scritto del 10 settembre 2010

Esercizio 1. Si consideri il seguente sistema di task periodici con job non interrompibili: $T_1 = (0, 2, 1, 5)$, $T_2 = (6, 3, 1, 3)$, $T_3 = (3, 8, 1, 8)$.

(a) Determinare una schedulazione ciclica strutturata per il sistema di task che minimizza l'overhead dello scheduler.

(b) Al tempo $t = 16$ viene generato un job sporadico di durata $e = 3$ e scadenza assoluta $t = 90$. È possibile accettare il job assumendo che non esistono altri job sporadici da eseguire? Giustificare la risposta.

Esercizio 2. Si consideri un sistema di tre task periodici con scadenze uguali al periodo così caratterizzati:

	T_1	T_2	T_3	
p_i	3	4	10	(periodo)
e_i	1	1	1	(tempo d'esecuzione)
K_i	1	1	2	(numero di auto-sospensioni)
x_i	0.1	0.5	0.5	(tempo max auto-sospensione)
θ_i	0.2	0.1	0.1	(tempo max non interrompibilità)

(a) Supponendo che i task siano indipendenti e senza risorse condivise, e che venga utilizzato uno scheduler RM con overhead e tempo di cambio di contesto trascurabile, determinare analiticamente se il sistema è schedulabile su un singolo processore.

(b) Ripetere l'analisi precedente supponendo che lo scheduler RM sia invocato periodicamente con periodo $p_0 = 0.5$; si consideri ancora trascurabile l'overhead dello scheduler (ossia $e_0 = CS_0 = CS = 0$).

Esercizio 3. Un sistema deve eseguire cinque job $J_1, \ldots, J_5$, con J_i di priorità maggiore di J_k se $i < k$. I job utilizzano tre risorse condivise R_1, R_2 e R_3, e le relative sezioni critiche sono: $J_1 : [R_1; 4]$, $J_2 :$ nessuna, $J_3 : [R_2; 3]$, $J_4 : [R_1; 3]\,[R_3; 1]$, $J_5 : [R_2; 2]\,[R_3; 1]$.

(a) Quali sono i tempi massimi di blocco dei job per conflitto di risorse utilizzando il procotollo NPCS?

(b) Quali sono i tempi massimi di blocco dei job per conflitto di risorse utilizzando il procotollo priority-inheritance?

Sistemi Embedded e Real-time (M. Cesati)

Soluzioni del compito scritto del 10 settembre 2010

Esercizio 1. *Si consideri il seguente sistema task periodici con job non interrompibili:* $T_1 = (0, 2, 1, 5)$, $T_2 = (6, 3, 1, 3)$, $T_3 = (3, 8, 1, 8)$.

(a) *Determinare una schedulazione ciclica strutturata per il sistema di task che minimizza l'overhead dello scheduler.*

Determiniamo la dimensione del frame f più appropriata:

- Vincolo sulle fasi dei task: f deve dividere esattamente la fase dei task T_2 e T_3, perciò $f \in \{1, 3\}$.
- Vincolo sui tempi d'esecuzione dei job:

$$f \geq \max\{1, 1, 1\} = 1$$

- Vincolo sulla divisibilità della lunghezza dell'iperperiodo ($\text{mcm}\{2, 3, 8\} = 24$):

$$f \in \{1, 2, 3, 4, 6, 8, 12, 24\} \cap \{1, 3\} = \{1, 3\}$$

- Vincolo sul task T_1 ($2f - \gcd\{2, f\} \leq 5$):

$$2 \cdot 1 - \gcd\{2, 1\} = 1 \leq 5$$
$$2 \cdot 3 - \gcd\{2, 3\} = 5 \leq 5$$

- Vincolo sul task T_2 ($2f - \gcd\{3, f\} \leq 3$):

$$2 \cdot 1 - \gcd\{3, 1\} = 1 \leq 3$$
$$2 \cdot 3 - \gcd\{3, 3\} = 3 \leq 3$$

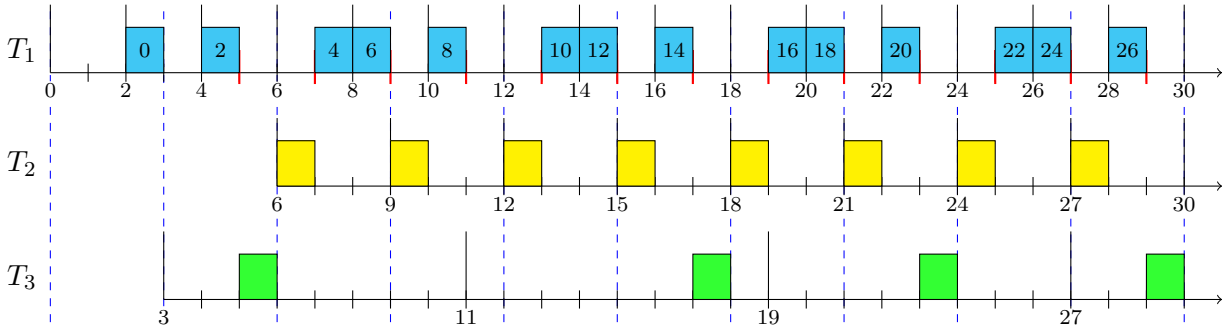
- Vincolo sul task T_3 ($2f - \gcd\{8, f\} \leq 8$):

$$2 \cdot 1 - \gcd\{8, 1\} = 1 \leq 8$$
$$2 \cdot 3 - \gcd\{8, 3\} = 5 \leq 8$$

Le dimensioni intere ammissibili per il frame sono dunque $f = 1$ e $f = 3$. La dimensione maggiore minimizza l'overhead globale dello scheduler, quindi scegliamo come dimensione del frame il valore tre.

Determiniamo ora una schedulazione ciclica strutturata per l'insieme di task. Notiamo che, poiché T_2 ha fase 6 e T_3 ha fase 3, non possiamo limitarci a considerare un intervallo lungo 24 unità di tempo iniziante dall'istante 0. Il primo istante in cui tutti i task sono attivi è 6, dunque determiniamo una schedulazione nell'intervallo tra 0 e $6 + 24 = 30$.

Una possibile soluzione è la seguente:



(Per maggior chiarezza è stato indicato in ciascun job di T_1 il corrispondente istante di rilascio.)
I due frame nell'intervallo $[0, 6]$ possono essere considerati identici ai due frame nell'intervallo $[24, 30]$, ovviamente omettendo i job non ancora rilasciati.
Si consideri ora che le condizioni dei tre task sono esattamente identiche all'istante 6 ed all'istante 30:

- Viene rilasciato un job di T_1 agli istanti 6 e 30; inoltre deve essere ancora eseguito un job di T_1 rilasciato in precedenza.
- Viene rilasciato un job di T_2 agli istanti 6 e 30; inoltre tutti i job di T_2 rilasciati in precedenza sono stati conclusi.
- Non esiste alcun job di T_3 da eseguire agli istanti 6 e 30; inoltre il prossimo job di T_3 sarà rilasciato agli istanti $6 + 5 = 11$ e $30 + 5 = 35$.

Perciò la schedulazione in $[6, 30]$ è fattibile anche nei successivi iperperiodi inizianti da 30.

(b) *Al tempo $t = 16$ viene generato un job sporadico di durata $e = 3$ e scadenza assoluta $t = 90$. È possibile accettare il job assumendo che non esistono altri job sporadici da eseguire? Giustificare la risposta.*

Il test di accettazione dei job sporadici utilizzato dall'algoritmo di schedulazione ciclica si basa sul calcolo dello *slack* lasciato libero dai task periodici. Osserviamo che nella

schedulazione determinata al punto precedente dell'esercizio esiste solo una unità di tempo lasciata libere dai task periodici in un iperperiodo, precisamente all'istante di tempo 11. Poiché la schedulazione si ripete identicamente, gli istanti di tempo a disposizione per i job sporadici rilasciati dopo l'istante 6 sono:

$$11 + 24 \cdot k \quad (k \geq 0) \quad = \quad 11, 35, 59, 83, 107, \ldots$$

Consideriamo ora il job sporadico rilasciato all'istante $t = 16$. Lo scheduler esegue il test di accettazione solo in corrispondenza dell'inizio del successivo frame, ossia all'istante $t = 18$. Di conseguenza i successivi tre intervalli liberi sono agli istanti 35, 59 e 83. Poiché la scadenza assoluta del job sporadico è all'istante 90, la scadenza può essere rispettata e lo scheduler deve dunque accettare il job sporadico.

Esercizio 2. *Si consideri un sistema di tre task periodici con scadenze uguali al periodo così caratterizzati:*

	T_1	T_2	T_3	
p_i	3	4	10	*(periodo)*
e_i	1	1	1	*(tempo d'esecuzione)*
K_i	1	1	2	*(numero di auto-sospensioni)*
x_i	0.1	0.5	0.5	*(tempo max auto-sospensione)*
θ_i	0.2	0.1	0.1	*(tempo max non interrompibilità)*

(a) *Supponendo che i task siano indipendenti e senza risorse condivise, e che venga utilizzato uno scheduler RM con overhead e tempo di cambio di contesto trascurabile, determinare analiticamente se il sistema è schedulabile su un singolo processore.*

Osserviamo preliminarmente che le scadenze relative dei task coincidono con i rispettivi periodi. Possiamo dunque cercare di applicare una condizione di schedulabilità basata sul fattore di utilizzazione del sistema di task.
Determiniamo i tempi massimi di blocco di ciascun task, come segue:

T_i	$b_i(ss)$	$b_i(np)$	b_i
T_1	$x_1 = 0.1$	$\max\{\theta_2, \theta_3\} = 0.1$	$b_1(ss)+(K_1+1)b_1(np) = 0.3$
T_2	$x_2+\min\{e_1, x_1\} = 0.6$	$\theta_3 = 0.1$	$b_2(ss)+(K_2+1)b_2(np) = 0.8$
T_3	$x_3+\min\{e_1, x_1\}+\min\{e_2, x_2\} = 1.1$	0	$b_3(ss)+(K_3+1)b_3(np) = 1.1$

Verifichiamo la condizione di schedulabilità per i tre job:

- Schedulabilità di T_1: $\quad \dfrac{e_1 + b_1}{p_1} = \dfrac{13}{30} < 1 = U_{RM}(1) \quad \Longrightarrow$ OK

- Schedulabilità di T_2: $\dfrac{e_1}{p_1} + \dfrac{e_2 + b_2}{p_2} = \dfrac{47}{60} < 0.8 < U_{RM}(2) \qquad \Longrightarrow$ OK
- Schedulabilità di T_3: $\dfrac{e_1}{p_1} + \dfrac{e_2}{p_2} + \dfrac{e_3 + b_3}{p_3} = \dfrac{119}{150} > 0.79 > U_{RM}(3) \qquad \Longrightarrow$ no

Applichiamo il test di schedulabilità al task T_3, l'unico che non rispetta la condizione di schedulabilità di Liu-Layland. La funzione di tempo necessario per T_3 è:

$$w_3(t) = e_3 + b_3 + \lceil t/p_1 \rceil e_1 + \lceil t/p_2 \rceil e_2 = 2.1 + \lceil t/3 \rceil + \lceil t/4 \rceil$$

Cerchiamo il punto fisso dell'equazione iterativa $w_3(t) = t$:

$$\begin{aligned} w_3(1) &= 2.1 + \lceil 1/3 \rceil + \lceil 1/4 \rceil &&= 4.1 \\ w_3(4.1) &= 2.1 + \lceil 4.1/3 \rceil + \lceil 4.1/4 \rceil &&= 6.1 \\ w_3(6.1) &= 2.1 + \lceil 6.1/3 \rceil + \lceil 6.1/4 \rceil &&= 7.1 \\ w_3(7.1) &= 2.1 + \lceil 7.1/3 \rceil + \lceil 7.1/4 \rceil &&= 7.1 \end{aligned}$$

Poiché l'istante 7.1 precede la scadenza assoluta del job (10), anche il task T_3 è schedulabile.

(b) *Ripetere l'analisi precedente supponendo che lo scheduler RM sia invocato periodicamente con periodo $p_0 = 0.5$; si consideri ancora trascurabile l'overhead dello scheduler (ossia $e_0 = CS_0 = CS = 0$).*

Poiché lo scheduler esegue in tempo trascurabile, i tempi di esecuzione dei task non devono essere aumentati, né è necessario considerare un job di massima priorità per modellare l'esecuzione dello scheduler. Inoltre non è necessario considerare il tempo d'esecuzione dei task che trasferiscono i job tra le varie code dello scheduler.
Però i tempi di blocco dei task devono essere aumentati per tenere in considerazione il fatto che lo scheduler è eseguito periodicamente. In particolare, per il tempo di blocco dovuto alla non interrompibilità si ha:

$$b_i(np)' = \left(\left\lceil \max_{i+1 \le k \le 3} \theta_k / p_0 \right\rceil + 1 \right) p_0$$

Perciò:

T_i	$b_i(ss)$	$b_i(np)'$	b_i'
T_1	0.1	$(\lceil \max\{\theta_2/p_0, \theta_3/p_0\} \rceil + 1)p_0 = 1$	$b_1(ss)+(K_1+1)b_1(np)' = 2.1$
T_2	0.6	$(\lceil \theta_3/p_0 \rceil + 1)p_0 = 1$	$b_2(ss)+(K_2+1)b_2(np)' = 2.6$
T_3	1.1	$p_0 = 0.5$	$b_3(ss)+(K_3+1)b_3(np)' = 2.6$

Consideriamo ora il task T_1: poiché $e_1 + b_1' = 3.1 > 3 = p_1$, il task non è più necessariamente schedulabile. La stessa conclusione è valida per tutti i task del sistema.

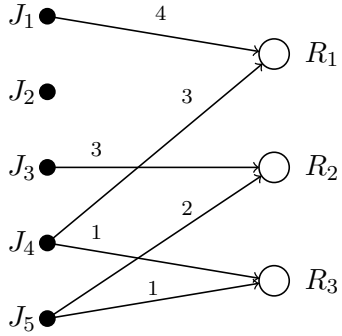
Esercizio 3. *Un sistema deve eseguire cinque job $J_1, \ldots, J_5$, con J_i di priorità maggiore di J_k se $i < k$. I job utilizzano tre risorse condivise R_1, R_2 e R_3, e le relative sezioni critiche sono: $J_1 : [R_1; 4]$, $J_2 :$ nessuna, $J_3 : [R_2; 3]$, $J_4 : [R_1; 3]\,[R_3; 1]$, $J_5 : [R_2; 2]\,[R_3; 1]$.*

(a) *Quali sono i tempi massimi di blocco dei job per conflitto di risorse utilizzando il procotollo NPCS?*

Il massimo tempo di blocco per conflitto di risorse $b_i(\text{rc})$ del job J_i è dato dalla lunghezza della più lunga sezione critica tra tutti i job di priorità inferiore. Perciò assumendo che i job non si auto-sospendano:

$$b_1(\text{rc}) = 3; \quad b_2(\text{rc}) = 3; \quad b_3(\text{rc}) = 3; \quad b_4(\text{rc}) = 2; \quad b_5(\text{rc}) = 0.$$

(b) *Quali sono i tempi massimi di blocco dei job per conflitto di risorse utilizzando il procotollo priority-inheritance?*



B_d	J_2	J_3	J_4	J_5
J_1			3	
J_2	$*$			
J_3		$*$		2
J_4			$*$	1

B_i	J_2	J_3	J_4	J_5
J_1				
J_2	$*$		3	
J_3		$*$	3	
J_4			$*$	2

Assumendo che i job non si auto-sospendano, i tempi di blocco per conflitti di risorse di ciascun job sono determinati dal valore massimo su ciascuna riga delle tabelle:

$$b_1(\text{rc}) = 3; \quad b_2(\text{rc}) = 3; \quad b_3(\text{rc}) = 3; \quad b_4(\text{rc}) = 2; \quad b_5(\text{rc}) = 0.$$