Sistemi Embedded e Real-time (M. Cesati)

Compito scritto del 10 febbraio 2011

Esercizio 1. Si consideri il seguente sistema di task periodici schedulato su un processore con un algoritmo "cyclic schedule": $T_1 = (0, 2, 1, 12)$, $T_2 = (0, 3, 1, 9)$, $T_3 = (0, 5, 3, 12)$, $T_4 = (84, 10, 2, 10)$. Si noti che mentre i primi tre task sono in fase, il primo job del quarto task viene rilasciato all'istante 84. I job non sono interrompibili. Determinare la dimensione del frame che minimizza l'overhead dello scheduler, oppure concludere che tale dimensione non esiste.

Esercizio 2. Un server procastinabile con periodo $p_s = 4$, budget $e_s = 2$ e fase indeterminata è schedulato su un singolo processore insieme a due task periodici, indipendenti e interrompibili: $T_1 = (10,\ 1)$ e $T_2 = (11,\ 1,\ 5)$.

(a) Determinare analiticamente se il sistema è schedulabile con EDF.

(b) Determinare analiticamente se il sistema è schedulabile con RM.

Esercizio 3. Un sistema è costituito da tre task periodici con fasi indeterminate $T_1 = (10, 1)$, $T_2 = (19, 2)$, $T_3 = (22, 3)$ e $T_4 = (23, 5)$. Ciascun job dei task T_1, T_2 e T_4 è sempre interrompibile e si auto-sospende al massimo due volte; ciascun job del task T_3 è sempre interrompibile e si auto-sospende al massimo una volta. In totale, nessun job rimane sospeso per più di una unità di tempo. I task utilizzano due risorse condivise R_1 e R_2 come segue: $T_1 : [R_1, 1]$, $T_2 : [R_1, 2\,[R_2, 1]\,]$, $T_3 :$ (nulla) $T_4 : [R_2, 2\,[R_1, 1]\,]$.

Determinare se l'insieme di task è schedulabile su un singolo processore con RM assumendo che lo scheduler abbia overhead trascurabile e che l'accesso alle risorse condivise sia controllato dal protocollo *priority ceiling*.

Sistemi Embedded e Real-time (M. Cesati)

Soluzioni del compito scritto del 10 febbraio 2011

Esercizio 1. *Si consideri il seguente sistema di task periodici schedulato su un processore con un algoritmo "cyclic schedule":* $T_1 = (0, 2, 1, 12)$, $T_2 = (0, 3, 1, 9)$, $T_3 = (0, 5, 3, 12)$, $T_4 = (84, 10, 2, 10)$. *Si noti che mentre i primi tre task sono in fase, il primo job del quarto task viene rilasciato all'istante* 84. *I job non sono interrompibili. Determinare la dimensione del frame che minimizza l'overhead dello scheduler, oppure concludere che tale dimensione non esiste.*

Determiniamo la dimensione del frame f più appropriata:

- Vincolo sulle fasi dei task: i primi tre task hanno fase zero, quindi non impongono alcun vincolo per f. Invece T_4 ha fase 84, quindi f deve dividere esattamente 84 ($= 3 \cdot 4 \cdot 7$):
$$f \in \{1, 2, 3, 4, 6, 7, 12, 14, 21, 28, 42, 84\}$$
- Vincolo sui tempi d'esecuzione dei job:
$$f \geq \max\{1, 1, 3, 2\} = 3$$
- Vincolo sulla divisibilità della lunghezza dell'iperperiodo ($\text{mcm}\{2, 3, 5, 10\} = 30$):
$$f \in \{1, 2, 3, 5, 6, 10, 15, 30\}.$$
Quindi combinando con le due condizioni precedenti si ha:
$$f \in \{3, 6\}$$
- Vincolo sul task T_1 ($2f - \gcd\{2, f\} \leq 12$):
$$\begin{array}{lll} 2 \cdot 3 - \gcd\{2, 3\} = 5 & \leq 12 & \text{ok} \\ 2 \cdot 6 - \gcd\{2, 6\} = 10 & \leq 12 & \text{ok} \end{array}$$
- Vincolo sul task T_2 ($2f - \gcd\{3, f\} \leq 9$):
$$\begin{array}{lll} 2 \cdot 3 - \gcd\{3, 3\} = 3 & \leq 9 & \text{ok} \\ 2 \cdot 6 - \gcd\{3, 6\} = 9 & \leq 9 & \text{ok} \end{array}$$

- Vincolo sul task T_3 ($2f - \gcd\{5, f\} \leq 12$):

$$\begin{array}{ll} 2 \cdot 3 - \gcd\{5,3\} = 5 & \leq 12 \quad \text{ok} \\ 2 \cdot 6 - \gcd\{5,6\} = 11 & \leq 12 \quad \text{ok} \end{array}$$

- Vincolo sul task T_4 ($2f - \gcd\{10, f\} \leq 10$):

$$\begin{array}{ll} 2 \cdot 3 - \gcd\{10,3\} = 5 & \leq 10 \quad \text{ok} \\ 2 \cdot 6 - \gcd\{10,6\} = 10 & \leq 10 \quad \text{ok} \end{array}$$

In conclusione, sono ammissibili come dimensione del frame $f = 3$ e $f = 6$. Il valore che minimizza l'overhead dello scheduler è $f = 6$.

Esercizio 2. *Un server procastinabile con periodo $p_s = 4$, budget $e_s = 2$ e fase indeterminata è schedulato su un singolo processore insieme a due task periodici, indipendenti e interrompibili:* $T_1 = (10,\, 1)$ *e* $T_2 = (11,\, 1,\, 5)$.

(a) *Determinare analiticamente se il sistema è schedulabile con EDF.*

Utilizzando il fattore di utilizzazione, condizione sufficiente per la schedulabilità di T_i è:

$$\sum_{k=1}^{n} \frac{e_k}{\min(D_k, p_k)} + \frac{e_s}{p_s}\left(1 + \frac{p_s - e_s}{D_i}\right) \leq 1$$

- Server procastinabile: $u_s = 2/4 \leq 1 \quad \Rightarrow$ schedulabile
- Task T_1: $1/10 + 1/\min(11,5) + (1/2)(1 + 2/10) = 9/10 \leq 1 \quad \Rightarrow$ schedulabile
- Task T_2: $1/10 + 1/\min(11,5) + (1/2)(1 + 2/5) = 1 \quad \Rightarrow$ schedulabile

Il sistema di task è dunque schedulabile con EDF.

(b) *Determinare analiticamente se il sistema è schedulabile con RM.*

Utilizzando il fattore di utilizzazione, condizione sufficiente per la schedulabilità di T_i è:

$$\sum_{k=1}^{i} \frac{e_k}{p_k} + u_s + \frac{e_s}{p_i} \leq U_{\text{RM}}(i+1)$$

(a condizione che per ogni task $T_1, \ldots, T_i$ la scadenza relativa coincida con il periodo)

- Server procastinabile: $u_s = 2/4 \leq 1 \quad \Rightarrow$ schedulabile
- Task T_1: $1/10 + 2/4 + 2/10 = 4/5 = 0{,}8 < 0{,}828 < U_{\text{RM}}(2) \quad \Rightarrow$ schedulabile
- Task T_2: non applicabile in quanto $p_2 \neq D_2$

Analizziamo la schedulabilità di T_2 per mezzo della sua funzione di tempo necessario, che in presenza di server procastinabile è:

$$w_i(t) = e_i + e_s + \left\lceil \frac{t - e_s}{p_s} \right\rceil e_s + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

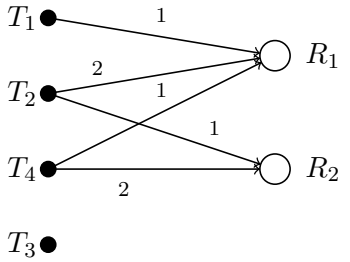
Perciò $w_2(t) = 3 + 2 \cdot \lceil (t-2)/4 \rceil + \lceil t/10 \rceil$. Determiniamo il punto fisso dell'equazione $w_2(t) = t$: $w_2(e_s + e_1 + e_2) = w_2(4) = 6$, $w_2(6) = 6$.

Poiché il tempo di risposta di T_2 (6) è maggiore della sua scadenza relativa $D_2 = 5$, il task T_2 non è schedulabile con RM.

Esercizio 3. *Un sistema è costituito da tre task periodici con fasi indeterminate* $T_1 = (10, 1)$, $T_2 = (19, 2)$, $T_3 = (22, 3)$ *e* $T_4 = (23, 5)$. *Ciascun job dei task* T_1, T_2 *e* T_4 *è sempre interrompibile e si auto-sospende al massimo due volte; ciascun job del task* T_3 *è sempre interrompibile e si auto-sospende al massimo una volta. In totale, nessun job rimane sospeso per più di una unità di tempo. I task utilizzano due risorse condivise* R_1 *e* R_2 *come segue:* $T_1 : [R_1, 1]$, $T_2 : [R_1, 2\,[R_2, 1]\,]$, T_3 : *(nulla)* $T_4 : [R_2, 2\,[R_1, 1]\,]$.

Determinare se l'insieme di task è schedulabile su un singolo processore con RM assumendo che lo scheduler abbia overhead trascurabile e che l'accesso alle risorse condivise sia controllato dal protocollo priority ceiling.

Determiniamo i massimi tempi di blocco per conflitto di risorse $b_i(\text{rc})$ dei tre task:



B_d	T_2	T_3	T_4
T_1	2		1
T_2	$*$		2
T_3		$*$	

B_i	T_2	T_3	T_4
T_1			
T_2	$*$		1
T_3		$*$	2

B_i	T_2	T_3	T_4
T_1			
T_2	$*$		1
T_3		$*$	

I tempi di blocco per conflitti di risorse sono: $b_1(\text{rc}) = 2$, $b_2(\text{rc}) = 2$, $b_3(\text{rc}) = 2$, $b_4(\text{rc}) = 0$.

Determiniamo i tempi di blocco dovuti all'auto-sospensione tramite la formula:

$$b_i(\text{ss}) = x_i + \sum_{k=1}^{i-1} \min(e_k, x_k)$$

$b_1(\text{ss}) = 1$; $b_2(\text{ss}) = 1 + \min(1,1) = 2$; $b_3(\text{ss}) = 1 + \min(1,1) + \min(2,1) = 3$; $b_4(\text{ss}) = 1 + \min(1,1) + \min(2,1) + \min(3,1) = 4$.

Poiché i job sono sempre interrompibili, i tempi totali di blocco sono dati dalla formula:

$$b_i = b_i(\text{ss}) + (K_i + 1) \cdot b_i(\text{rc})$$

$b_1 = 1 + 3 \cdot 2 = 7; \quad b_2 = 2 + 3 \cdot 2 = 8; \quad b_3 = 3 + 2 \cdot 2 = 7; \quad b_4 = 4 + 3 \cdot 0 = 4.$

Per determinare se il sistema è schedulabile, applichiamo la condizione di schedulabilità al singolo task T_i:

$$\sum_{k=1}^{i} \frac{e_k}{p_k} + \frac{b_i}{p_i} \leq U_{\mathrm{RM}}(i)$$

- Task T_1: $1/10 + 7/10 = 4/5 < 1 = U_{\mathrm{RM}}(1)$ $\Rightarrow$ schedulabile
- Task T_2: $1/10 + 2/19 + 8/19 = 119/190 < 0.63 < 0.828 < U_{\mathrm{RM}}(2)$ $\Rightarrow$ schedulabile
- Task T_3: $1/10 + 2/19 + 3/22 + 7/22 < 0.66 < 0.779 < U_{\mathrm{RM}}(3)$ $\Rightarrow$ schedulabile
- Task T_4: $1/10{+}2/19{+}3/22{+}5/23{+}4/23 < 0.74 < 0.756 < U_{\mathrm{RM}}(4)$ $\Rightarrow$ schedulabile

Il sistema di task è dunque schedulabile.