Sistemi Embedded e Real-time (M. Cesati)

Compito scritto del 8 settembre 2009

Esercizio 1. Il seguente sistema di task periodici è schedulato su un processore con un algoritmo "cyclic schedule": $T_1 = (5, 4, 0.5, 9)$, $T_2 = (5, 1, 7)$, $T_3 = (10, 2)$, $T_4 = (8, 2.5, 9)$. (Si noti che il task T_1 ha fase 5, mentre gli altri task hanno fase 0.)

(a) Determinare una dimensione intera appropriata per il frame. I job non sono interrompibili.

(b) Dimostrare formalmente che non esiste alcuna schedulazione valida per l'insieme di task utilizzando $f = 5$ come dimensione del frame.

Esercizio 2. Un server procastinabile con periodo $p_s = 1$, budget $e_s = 0.5$ e fase indeterminata è schedulato su un singolo processore insieme a due task periodici, indipendenti e interrompibili: $T_1 = (10, 1)$ e $T_2 = (11, 4.1)$.

(a) Dimostrare analiticamente che il sistema è schedulabile con EDF.

(b) Spiegare il motivo per cui il sistema di task periodici non è più necessariamente schedulabile se il server procastinabile mantiene la stessa dimensione ma viene invocato con un frequenza pari ad un sesto di quella originale ($p'_s = 6$, $e'_s = 3$).

(c) Con riferimento al punto (b), presentare un esempio di schedulazione EDF in cui uno dei due task periodici non rispetta la propria scadenza.

Esercizio 3. Un sistema è costituito da tre task periodici con fasi indeterminate $T_1 = (10, 2)$, $T_2 = (19, 3)$ e $T_3 = (20, 4)$. Ciascun job dei tre task è sempre interrompibile e si auto-sospende al massimo due volte; 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: $T_1 : [R_1, 1]$, $T_2 : [R_2, 1]\,[R_1, 2]$, $T_3 : [R_2, 2\,[R_1, 1]\,]$.

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 8 settembre 2009

Esercizio 1. *Il seguente sistema di task periodici è schedulato su un processore con un algoritmo "cyclic schedule":* $T_1 = (5, 4, 0.5, 9)$, $T_2 = (5, 1, 7)$, $T_3 = (10, 2)$, $T_4 = (8, 2.5, 9)$. *(Si noti che il task* T_1 *ha fase 5, mentre gli altri task hanno fase 0.)*

(a) *Determinare una dimensione intera appropriata per il frame. I job non sono interrompibili.*

- Vincolo sulle fasi dei task: per definizione, il primo job di ciascun task deve essere rilasciato esattamente all'inizio di un frame. In altri termini, la fase di ciascun task deve essere un multiplo intero non negativo della dimensione del frame.

  Poiché T_1 ha fase 5, cinque deve essere un multiplo intero di f. Gli altri task con fase 0 non pongono alcun vincolo.

- Vincolo sui tempi d'esecuzione dei job:

$$f \geq \max\{0.5, 1, 2, 2.5\} = 2.5$$

- Vincolo sulla divisibilità della lunghezza dell'iperperiodo ($H = \text{mcm}\{4, 5, 10, 8\} = 40$):

$$f \in \{1, 2, 4, 5, 8, 10, 20, 40\}.$$

  Considerando i vincoli precedenti: $f \in \{5\}$.

- Vincolo sul task T_1 ($2f - \gcd\{4, f\} \leq 9$): $\quad 2 \cdot 5 - \gcd\{4, 5\} = 9 \leq 9$
- Vincolo sul task T_2 ($2f - \gcd\{5, f\} \leq 7$): $\quad 2 \cdot 5 - \gcd\{5, 5\} = 5 \leq 7$
- Vincolo sul task T_3 ($2f - \gcd\{10, f\} \leq 10$): $\quad 2 \cdot 5 - \gcd\{10, 5\} = 5 \leq 10$
- Vincolo sul task T_4 ($2f - \gcd\{8, f\} \leq 9$): $\quad 2 \cdot 5 - \gcd\{8, 5\} = 9 \leq 9$

L'unica dimensione intera possibile per il frame è $f = 5$.

(b) *Dimostrare formalmente che non esiste alcuna schedulazione valida per l'insieme di task utilizzando* $f = 5$ *come dimensione del frame.*

Anche se il fattore di utilizzazione globale del sistema di task periodici

$$U_T = \frac{0.5}{4} + \frac{1}{5} + \frac{2}{10} + \frac{2.5}{8} = \frac{67}{80} = 0.8375$$

è inferiore ad uno, non esiste alcuna schedulazione ciclica valida per $f = 5$. Per dimostrarlo formalmente si deve tenere presente che:

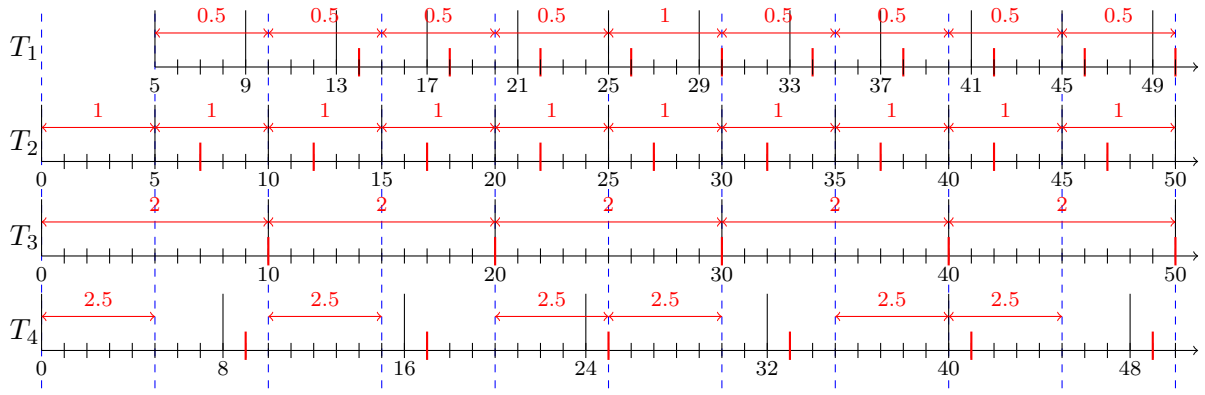
- Un job può essere schedulato solo se il suo istante di rilascio cade in un frame precedente oppure coincide con l'inizio stesso del frame.
- Lo scheduler deve controllare l'avvenuto rispetto delle scadenze dei vari job; per questo motivo, un job deve essere completato (e dunque deve essere completamente eseguito) in un frame precedente a quello contenente la sua scadenza oppure in un frame che termina esattamente nell'istante di scadenza del job.

Quindi per ciascun job è possibile determinare in quali frame esso deve essere eseguito.

Consideriamo ad esempio il quinto job di T_1 rilasciato all'istante 21. L'istante 21 cade all'interno del frame $(20, 25)$, quindi il job può essere eseguito a partire dal frame $(25, 30)$. D'altra parte la scadenza del job è all'istante $21 + 9 = 30$, ossia al termine del frame $(25, 30)$. Di conseguenza, il job può essere eseguito soltanto nel frame $(25, 30)$.

Stesso discorso vale anche per il sesto job di T_1: poiché è rilasciato all'istante 25, può essere eseguito dal frame $(25, 30)$ in poi. Dato che la sua scadenza $25 + 9 = 31$ cade nel frame $(30, 35)$, il frame $(25, 30)$ è l'unico in cui può essere eseguito il job.

Ripetendo il ragionamento per ciascun job nei primi 10 frame si ottiene la figura seguente; il lavoro da effettuare per ciascun task è rappresentato da frecce che si estendono tra uno o più frame e valori numerici che indicano la somma dei tempi d'esecuzione dei job in quei frame.



Osserviamo ora che nei due frame compresi tra l'istante 20 e l'istante 30 il lavoro totale da compiere è pari a $0.5 + 1 + 1 + 1 + 2 + 2.5 + 2.5 = 10.5$, e di conseguenza il fattore di utilizzazione è

$$\frac{10.5}{10} = 1.05 > 1.$$

Perciò in qualunque schedulazione almeno uno dei job che devono essere eseguiti in questi due frame non potrà essere completato. Ciò conclude la dimostrazione.

Esercizio 2. *Un server procastinabile con periodo $p_s = 1$, budget $e_s = 0.5$ e fase indeterminata è schedulato su un singolo processore insieme a due task periodici, indipendenti e interrompibili:* $T_1 = (10, 1)$ *e* $T_2 = (11, 4.1)$.

(a) *Dimostrare analiticamente che il sistema è schedulabile con EDF.*

La presenza del server procastinabile impone di dover controllare separatamente la schedulabilità di ciascun task. Condizione sufficiente per la schedulabilità di T_i è:

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

- Dimensione del server: $u_s = e_s/p_s = 1/2$
- Densità dei task periodici: $\sum_{k=1}^{n} \frac{e_k}{\min(p_k, D_k)} = \Delta = 1/10 + 4.1/11 = 26/55$
- Server procastinabile: $u_s + \Delta = 1/2 + 26/55 = 107/110 \leq 1 \quad \Rightarrow$ schedulabile
- Task T_1: $26/55 + 1/2 \cdot (1 + (1/2) \cdot 1/10) = 439/440 \leq 1 \quad \Rightarrow$ schedulabile
- Task T_2: $26/55 + 1/2 \cdot (1 + (1/2) \cdot 1/11) = 219/220 \leq 1 \quad \Rightarrow$ schedulabile

Il sistema è quindi schedulabile con EDF.

(b) *Spiegare il motivo per cui il sistema di task periodici non è più necessariamente schedulabile se il server procastinabile mantiene la stessa dimensione ma viene invocato con un frequenza pari ad un sesto di quella originale ($p'_s = 6$, $e'_s = 3$).*

Per prima cosa, verifichiamo che effettivamente con il nuovo server procastinabile il sistema non rispetta le condizioni di schedulabilità. La densità dei task periodici Δ e la dimensione del server u_s sono invariate; quindi:

- Task T_1: $26/55 + 1/2 \cdot (1 + 3 \cdot (1/10)) = 1 + 27/220 > 1 \quad \Rightarrow$ test fallito!
- Task T_2: $26/55 + 1/2 \cdot (1 + 3 \cdot (1/11)) = 1 + 6/55 > 1 \quad \Rightarrow$ test fallito!

Se il server deve mantenere una dimensione costante, ad una diminuzione della sua frequenza di invocazione, e quindi ad un aumento del suo periodo p_s, corrisponde proporzionalmente un aumento del budget e_s.

I server procastinabili in particolari condizioni possono consumare più tempo di quello richiesto da un task periodico equivalente. In particolare, se in un certo intervallo di tempo $(t_0, t\,]$:

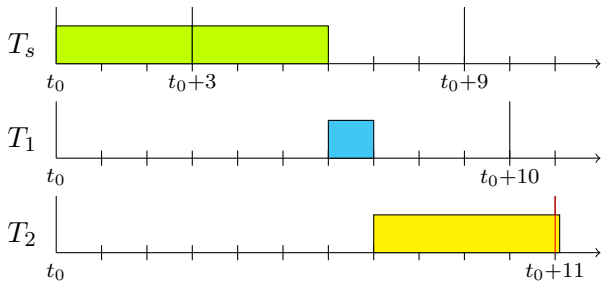
- il budget del server al tempo t_0 è pari a e_s,
- in $(t_0, t\,]$ il server è sempre impegnato,
- un nuovo periodo del server inizia a $t_0 + e_s$,
- la scadenza $t_0 + e_s$ precede la scadenza di tutti i job periodici pronti per l'esecuzione in $(t_0, t_0 + e_s\,]$,

allora il server occupa il processore per un tempo e_s in più rispetto ad un normale task periodico.

In conclusione, poiché il nuovo server procastinabile ha un budget maggiore rispetto a quello originale ($e'_s = 3$ contro $e_s = 0.5$), esso può potenzialmente ritardare in misura maggiore i task periodici, tanto da causare il mancato rispetto delle scadenze.

(c) *Con riferimento al punto (b), presentare un esempio di schedulazione EDF in cui uno dei due task periodici non rispetta la propria scadenza.*

Un esempio di schedulazione EDF in cui si verifica lo scenario descritto nel punto (b) è il seguente:

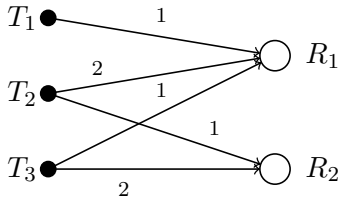


All'istante t_0 vengono rilasciati i job di T_1 e T_2; la scadenza del job di T_2 è a $t_0 + 11$, ma i due job di T_s ed il job di T_1 (che hanno scadenze più vicine) lasciano solo 4 unità di tempo a disposizione per T_2, insufficienti per completare il job di durata 4.1.

Esercizio 3. *Un sistema è costituito da tre task periodici con fasi indeterminate* $T_1 = (10, 2)$, $T_2 = (19, 3)$ *e* $T_3 = (20, 4)$. *Ciascun job dei tre task è sempre interrompibile e si auto-sospende al massimo due volte; 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:* $T_1 : [R_1, 1]$, $T_2 : [R_2, 1]\,[R_1, 2]$, $T_3 : [R_2, 2\,[R_1, 1]\,]$.

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_1	2	1
T_2	$*$	2

B_i	T_2	T_3
T_1		
T_2	$*$	1

B_c	T_2	T_3
T_1		
T_2	$*$	1

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

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

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

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

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

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

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

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

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

- Task T_1: $\quad 2/10 + 7/10 = 9/10 < 1 = U_{\text{RM}}(1)$ $\quad \Rightarrow$ schedulabile
- Task T_2: $\quad 2/10 + 3/19 + 8/19 = 74/95 < 0.779 < 0.828 < U_{\text{RM}}(2)$ $\quad \Rightarrow$ schedulabile
- Task T_3: $\quad 2/10 + 3/19 + 4/20 + 3/20 < 0.708 < 0.779 < U_{\text{RM}}(3)$ $\quad \Rightarrow$ schedulabile

Il sistema di task è dunque schedulabile.