Anàlisi de planificabilitat amb co-rutines

1. Introducció

Sigui **θ** un conjunt de **N** tasques a gestionar cooperativament com co-rutines:

$$\theta = \{\tau_1, \tau_2, \ldots, \tau_i, \ldots \tau_N\}$$

Tota tasca **i** de **θ** tindrà la possibilitat de respondre a 1 o més esdeveniments. S'identificarà un esdeveniment **k** associat a la tasca **i** mitjançant e_i^k on $k = \{0,1,\ldots,\varepsilon_i\}$ i $\boldsymbol{\varepsilon_i}$ representa el número total d'esdeveniments a que respon la tasca **i**. Existeix també la possibilitat que una tasca no respongui a esdeveniments. Es considerarà esdeveniment la satisfacció de la condició associada a una transició. Les transicions immediates no es consideraran associades a cap esdeveniment.

La resposta a un esdeveniment requerirà que una tasca executi les accions associades a un o més estats entre els quals les transicions seran *immediates*. Al número total d'estats que segueixin a un esdeveniment mitjançant transicions immediates se'l denotarà per λ_i^k. No necessàriament tots els λ_i^k estats seran necessaris per donar resposta a l'esdeveniment e_i^k. El número d'estats necessaris per respondre a un esdeveniment **k** per la tasca **i** s'identificarà com l_i^k. Com les transicions són immediates amb l_i^k scans, la tasca **i** respondrà a l'esdeveniment **k**. Sempre s'acomplirà la següent relació: $1 \le l_i^k \le \lambda_i^k$.

Per cada esdeveniment es calcularà el pitjor temps de resposta, R_i^k i es contrastarà amb els terminis especificats, D_i^k. L'acompliment de la següent desigualtat comportarà la seguretat que les tasques considerades acompliran les especificacions temporals fins i tot en el cas més desfavorable o crític:

$$R_i^k \le D_i^k \quad \forall i,k$$

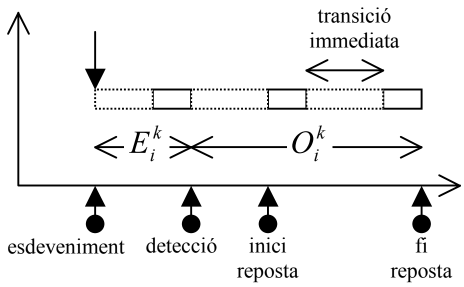
No obstant, la violació d'alguna desigualtat no esdevindrà indicador suficient de no planificabilitat. Això es deu a que en el càlcul dels temps de resposta es realitzaran algunes aproximacions de cara a simplificar les expressions resultants. Aquest fet comportarà l'obtenció d'expressions conservadores. Per això diem que *aquesta anàlisi és suficient però no necessària*.

Pel càlcul del temps de resposta farà falta conèixer els temps de còmput màxim per totes i cadascuna de les **N** tasques. Aquest temps de còmput representarà el temps màxim de *scan* d'una tasca sigui quin sigui el seu estat. Aquest temps de còmput màxim s'identificarà mitjançant C_i.

El temps de resposta màxim es descomposarà normalment en un temps de detecció de l'esdeveniment E_i^k i un temps de reacció O_i^k al mateix, en que se li donarà la resposta adequada. És a dir,

$$R_i^k = E_i^k + O_i^k$$

Tota tasca requerirà rebre 1 *scan* per detectar l'esdeveniment, ja que normalment aquest es detectarà mitjançant la condició associada a una transició d'entrada a un estat de la tasca. A aquest *scan* caldrà afegir l_i^k *scans* més per respondre l'esdeveniment convenientment. Per tant R_i^k representarà el pitjor temps necessari per que la tasca **i** rebi $\left(l_i^k + 1\right)$ *scans*.



A continuació s'analitzaran els tres tipus de gestors vistos a classe. Per a cadascun d'ells s'obtindrà una expressió diferent del temps de resposta.

2. Gestió seqüencial

En un gestor seqüencial totes les tasques reben un *scan* per cicle complert de gestió. Per cada *scan* de la tasca **i,** les demés tasques reberan també un *scan*. La situació més desfavorable es que l'esdeveniment e_i^k es produeixi just després que finalitzi la darrera invocació de la tasca **i**. Aquest fet comporta que la totalitat de les tasques hauran de rebre $\left(l_i^k + 1\right)$ *scans* fins que la tasca **i** hagi finalitzat la resposta a l'esdeveniment e_i^k . En definitiva:

$$R_i^k = E_i^k + O_i^k = \sum_{\forall \tau_j \in \theta} C_j + l_i^k \sum_{\forall \tau_j \in \theta} C_j = \left(l_i^k + 1\right) \sum_{\forall \tau_j \in \theta} C_j$$

Tots aquells esdeveniments que requereixin el mateix número de scans per ésser respostos tindran el mateix temps de resposta màxim.

3. Gestió amb prioritats

En un gestor amb prioritats algunes tasques reben més *scans* que d'altres per cicle complert. Això comporta que el seu temps de resposta millora respecte al que assolien amb un gestor seqüencial. Suposarem una sola tasca a prioritzar $\boldsymbol{\tau_p}$ que rep **L** scans per cicle complert de les demés. Dues invocacions successives de la tasca prioritària hauran d'estar separades com a mínim per la invocació d'una tasca no prioritària. Això implica que s'haurà d'acomplir la condició següent:

$$L \leq N - 1$$

Un cop l'ordre d'invocació de totes les tasques haurà quedat prefixat, es pot determinar una partició del conjunt de tasques no prioritàries $\theta_{np} = \theta - \{\tau_p\}$:

$$\pi = \{\rho_1, \rho_2, \ldots, \rho_L\} \quad \theta_{np} = \bigcup_{m=1}^{L} \rho_m \quad i \quad \rho_m \cap \rho_n = \phi \ \forall m, n, m \neq n$$

$\boldsymbol{\rho_m}$ és un subconjunt de $\boldsymbol{\theta_{np}}$ que conté les tasques que són invocades entre dues invocacions successives de la tasca prioritària $\boldsymbol{\tau_p}$.

El cas més desfavorable per la tasca prioritària correspon a que entre l'esdeveniment i la seva detecció s'interposin les tasques corresponents a un $\boldsymbol{\rho_m}$ tal que la suma dels seus temps de còmputs sigui màxim. És a dir,

$$R_p^k = E_p^k + O_p^k = \left(\max_{\forall m} \sum_{\forall \tau_j \in \rho_m} C_j + C_p \right) + l_p^k \left(\max_{\forall m} \sum_{\forall \tau_j \in \rho_m} C_j + C_p \right) = \left(l_p^k + 1\right) \left(\max_{\forall m} \sum_{\forall \tau_j \in \rho_m} C_j + C_p \right)$$

Per qualsevol tasca no prioritària el cas més desfavorable continua essent com en el gestor seqüencial, quan l'esdeveniment es produeix just després d'una invocació de la pròpia tasca. Per que aquesta rebi $\left(l_i^k+1\right)$ *scans* cal que totes les demés no prioritàries rebin igual número de *scans* i la prioritària en reberà aquest número multiplicat per L:

$$R_i^k = E_i^k + O_i^k = \left(\sum_{\forall \tau_j \in \theta_{np}} C_j + LC_p\right) + l_i^k \left(\sum_{\forall \tau_j \in \theta_{np}} C_j + LC_p\right) = \left(l_i^k+1\right)\left(\sum_{\forall \tau_j \in \theta} C_j + (L-1)C_p\right)$$

S'observa que el temps de resposta de la tasca prioritària és menor mentre que el de les demés augmenta lleugerament respecte al gestor seqüencial, tal com era d'esperar.

Un cop decidit el valor de L, interessarà que la partició es faci de tal manera que el màxim sigui el mínim de totes les possible particions. És a dir, s'intercalaran les invocacions a la tasca prioritària de forma que els temps de còmput de les demés tasques quedin repartits el més homogèniament possible entre els $\boldsymbol{\rho_m}$.

4. Gestió de mínima latència

En un gestor de mínima latència les tasques prioritzades reben un *scan* per cada invocació d'una tasca no prioritària. A més, en resposta a esdeveniments, rebran immediatament tants *scans* successius com facin falta fins a finalitzar-ne la resposta. Això comporta que el seu temps de detecció queda reduït al mínim així com el temps de reacció, millorant, per tant, el temps de resposta respecte als gestors precedents. Suposarem una sola tasca a prioritzar $\boldsymbol{\tau_p}$ que rep **N-1** *scans* per cicle complert de les demés.

En aquest cas la partició comporta que cada subconjunt $\boldsymbol{\rho_m}$ comprèn una sola tasca. És a dir,

$$\pi = \{\rho_1, \rho_2, \ldots, \rho_{N-1}\} = \{\{\tau_1\}, \{\tau_2\}, \ldots, \{\tau_{N-1}\}\} \quad \tau_p = \tau_N$$

El cas més desfavorable per la tasca prioritària correspon a que entre l'esdeveniment i la seva detecció s'interposi la tasca amb major temps de còmput:

$$R_p^k = E_p^k + O_p^k = \left(\max_{\forall \tau_j \in \theta_{np}} C_j + C_p\right) + l_p^k C_p = \max_{\forall \tau_j \in \theta_{np}} C_j + \left(l_p^k+1\right)C_p$$

Cal observar que en el temps de reacció no intervé la invocació que cap altra tasca que no sigui la prioritària.

Pel que fa a les tasques no prioritàries, el cas més desfavorable continua essent com en el gestor seqüencial, quan l'esdeveniment es produeix just després d'una invocació de la pròpia tasca. Per que aquesta rebi $\left(l_i^k+1\right)$ *scans* cal que totes les demés no prioritàries rebin igual número de *scans* i la prioritària en reberà aquest número multiplicat per **N-1**. A més es suposarà que en cada invocació de la prioritària aquesta ha de respondre a l'esdeveniment e_p^k amb més número d'estats successius encadenats per transicions immediates, λ_i^k :

$$R_i^k = E_i^k + O_i^k = \left(\sum_{\forall \tau_j \in \theta_{np}} C_j + (N-1)\left(\max_{\forall k} \lambda_p^k + 1\right)C_p\right) + l_i^k \left(\sum_{\forall \tau_j \in \theta_{np}} C_j + (N-1)\left(\max_{\forall k} \lambda_p^k + 1\right)C_p\right) =$$

$$= \left(l_i^k+1\right)\left(\sum_{\forall \tau_j \in \theta_{np}} C_j + (N-1)\left(\max_{\forall k} \lambda_p^k + 1\right)C_p\right)$$

S'observa que el temps de resposta de la tasca prioritària és menor mentre que el de les demés pot augmentar notablement respecte al gestor seqüencial, degut a que s'està considerant una situació molt desfavorable.