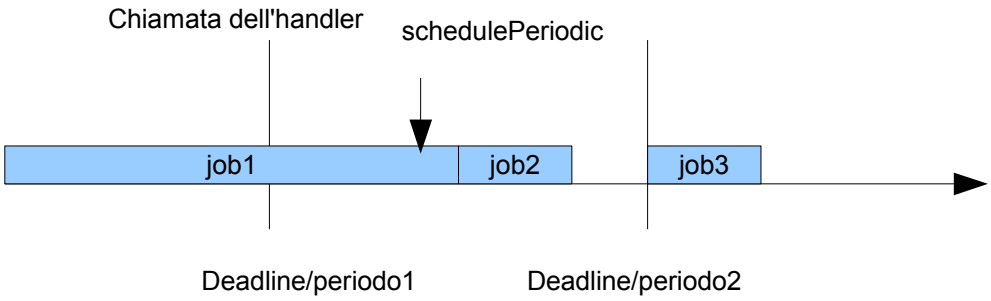
Cosa succede quando capita un deadline miss.

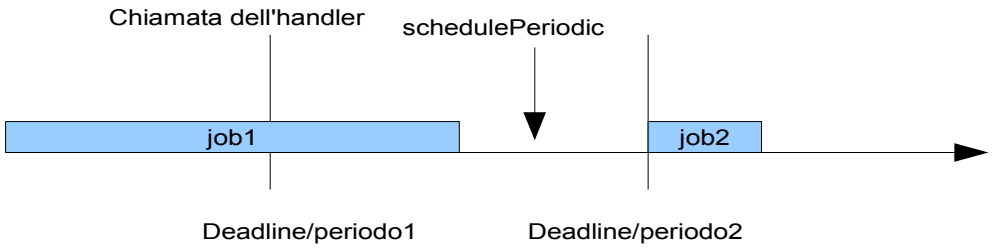
Quando si verifica un deadlineMissed, ossia quando un thread che è attivo (ossia in esecuzione, pronto per l'esecuzione o bloccato nell'accesso ad una risorsa) supera la deadline specificata nei suoi release Perameters, viene schedultato, se presente, l'handler associato per quel thread (se entra in esecuzione o meno dipende dalla prioriotà). Il job non viene mai interrotto, sia che l'handler chiami il metodo schedulePeriodic() sul thread o meno. Se non viene chiamato questo metodo il thread rimane bloccato sulla chiamata di waitForNextPeriod, schedulePeriodic serve, appunto, a sbloccare il thread.Questa soluzione può portare a problemi nel caso il thread sia entrato in deadlock ed eserciti preemption su altri processi.

Per poter gestire il momento in cui veniva eseguito il metodo schedulePeriodic si è introdotta una sleep nell'handler, in modo che il thread preposto a chiamare l'handler si addormentasse ed effettuasse l'operazione dopo che il job era giunto a terminazione. L'alternativa di eseguire una busy wait non era valida in quanto l'handler avrebbe occupato la cpu, non consentendo al job di finire il lavoro: difatto le condizioni non sarebbero mutate, ma si sarebbero presentate inalterate al termine della busy wait eseguita dall'handler.

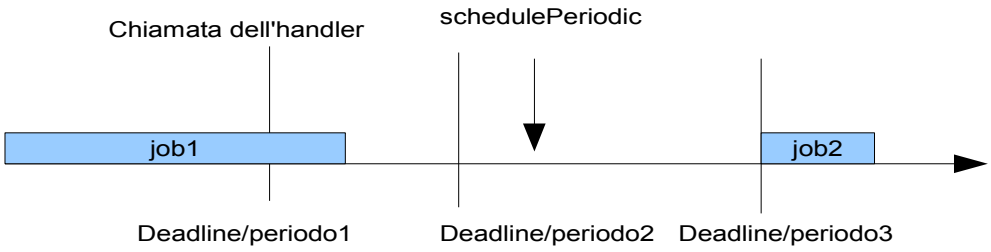
Attraverso una serie di figure mostro vari scenari di funzionamento.



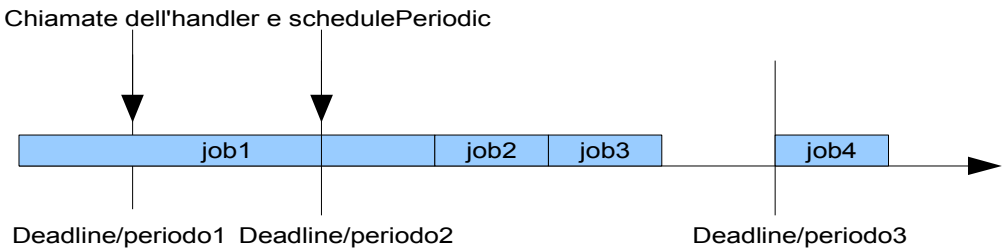
Se la chiamata dello schedule periodic avviene come in figura prima del termine del job che ha sforato la deadline il job successivo viene schedulato immediatamente al termine di quello precedente.



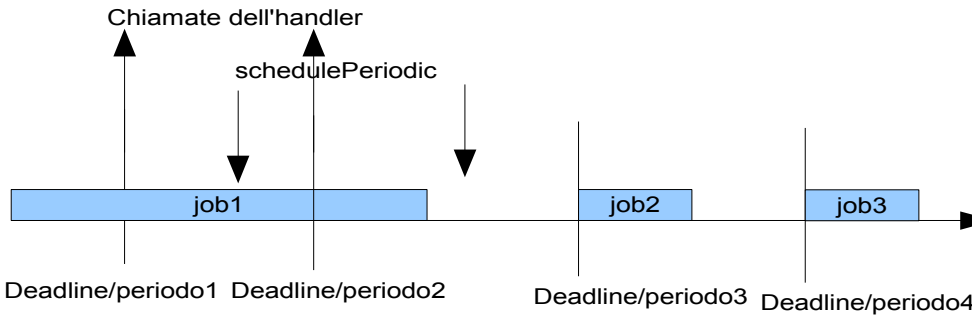
Nel caso in cui la chiamata da parte dell'handler del metodo schedule periodic avviene dopo il termine del job (e quindi dopo che questo ha chiamato waitForNextPeriod) il job successivo viene schedulato solamente nel prossimo periodo.



Di conseguenza se la chiamata di schedule periodic avviene nel secondo periodo, il prossimo job sarà schedulato solo nel terzo.



Come già visto qualora il job sfori due deadline si adotta una politca as soon as possible schedulando i job immediatamente dopo il termine del precedente finchè non si torna “in pari”.



Stranamente, se una schedulePeriodic avviene dopo il termine del job nessun job viene schedulato dopo il primo, ma vengono tutti avviati, uno per periodo, a partire dal periodo successivo. Si ha, a tutti gli effettti, una politica in stile SKIP. La differenza è che, se la skip pura impone di saltare i job, l'effetto di uno schedulePeriodic dopo che il thread ha terminato il job è quello di differirli. La differenza diventa inesistente nello schenario di thread non hanno un numero di iterazioni come quelli usati per i test, ma eseguono in loop infinito.

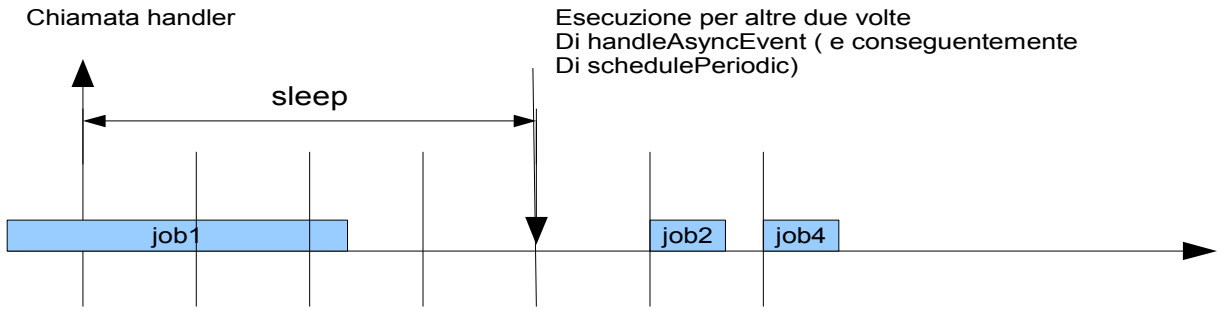
Handlers &Threads

Dalle mie prove emerge che ad ogni thread real time viene associato un altro thread che controlla se questo rispetta le deadline e, in caso di sforamento, provvede ad eseguire il metodo handleAsyncEvent ad esso associato. Sono arrivato a questa conclusione dopo alcuni esperimenti in cui stampavo il nome e l'identificativo del thread che eseguiva il metodo citato sopra.

In primo luogo ho fatto eseguire due distinti Thread ai quali erano associati due handler diversi, sia su processori differenti che sulla stessa cpu. Il risultato è che in entrambi i casi gli handler erano sempre chiamati da RealTimeThread diversi.

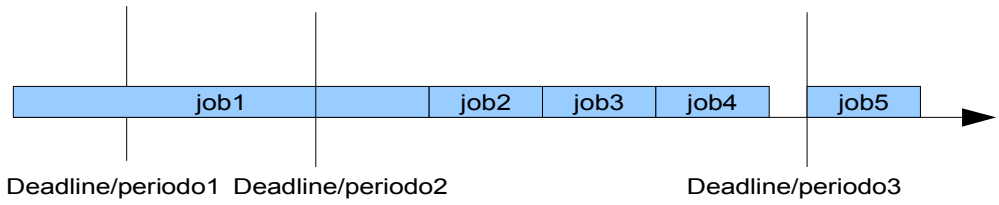
In un secondo momento ho provato a cambiare l'handler associato allo stesso thread tra due eventi, con il risultato che il thread servitore era lo stesso: segno che questo è collegato al thread periodico e non al suo handler.

Ho anche provato a fare eseguire una sleep al thread servitore che lo addormentasse e lo rendesse insensibile ad una serie di eventi verificatisi mentre era in sleep. Il risultato è che il metodo handle asyncEvent viene invocato solo al termine della sleep tante volte quanti gli eventi “pendenti”. Il sistema permette comunque di controllare il numero di eventi pendenti tramite tre metodi:`getAndClearPendingFireCount()`; `getAndDecrementPendingFireCount()`; `getAndIncrementPendingFireCount()`.Ad ogni modo, sia che la schedulePeriodic sia stata eseguita tre volte oppura solo una il comportamento del sistema resta lo stesso: visto che questa era stata eseguita dopo la fine del job i job successivi vengono schedultati a partire dal periodo successivo alla schedulePeriodic stessa (in questo caso a quello successivo il risveglio dell'handler)



Cosa succede in caso di mancanza dell'handler.

Curiosamente, nel caso in cui ad un thread non sia associato nessun handler, il sistema ha a defautl sì una politica alla as soon as possible, ma schedula, nel periodo in cui il thread termina il job che ha sforato la deadline, un job in più di quanto sia logico aspettarsi.



Nell'esempio riportato in figura, ad esempio, il primo job termina nel terzo periodo. Di conseguenza sarebbe logico aspettarsi che vengano schedulati immediatamente due job: quelli relativo al secondo ed al terzo periodo (come, ricordo, avviene nel caso in cui si associ al thred un handler che esegue tempestivamente la schedulePeriodic()). Invece, ne viene eseguito uno in più, prima di tornare all'esecuzione di un thread per periodo.

In caso di più schedulePeriodic

Ho anche provato ad eseguire più schedulePeriodic nello stessa chiamata dell'handler. Non cambia nulla, i thread vengono schedulati regolarmente, come se ci fosse stata una sola chiamata.