JAVA REAL-TIME: ANALISI DELL'ARCHITETTURA E VALUTAZIONE SPERIMENTALE DELLA PIATTAFORMA SOLARIS 10.9

RELATORE:
Chiar.mo Prof. Eugenio Faldella

Candidato:
Marco Nanni

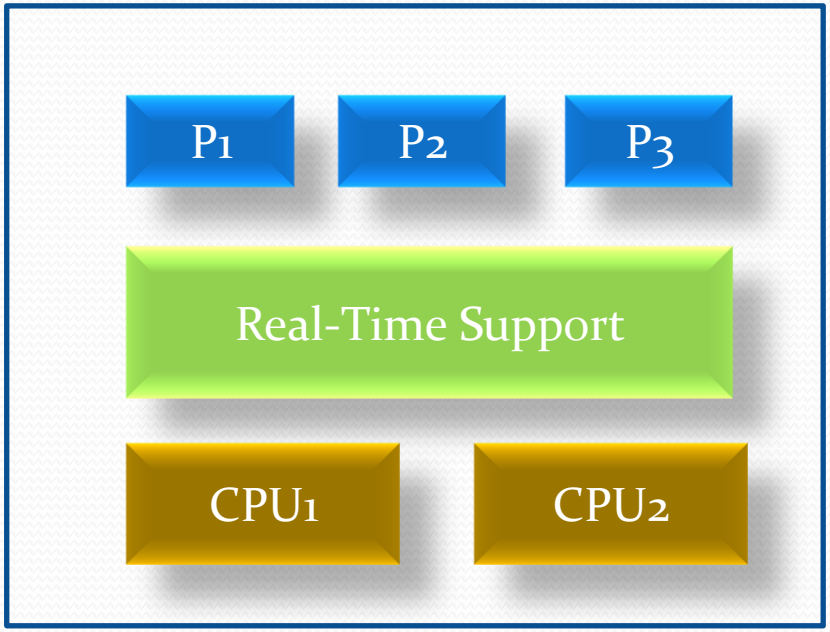
CORRELATORI
Chiar.mo Prof. Marco Prandini
Dott. Ing. Primiano Tucci

sommario

- I sistemi in tempo reale
- I limiti di Java come piattaforma in tempo reale
- La specifica Java Real – Time (RTSJ)
- Primi esperimenti con Java RealTime
- Realizzazione di scheduler EDF per Java Real-Time
- Analisi della politica di default di gestione dei deadline miss e delle sue criticità
- Realizzazione della politica SKIP in java Real-Time
- Realizzazione della politica SKIPSTOP in java Real-Time

I sistemi in tempo reale

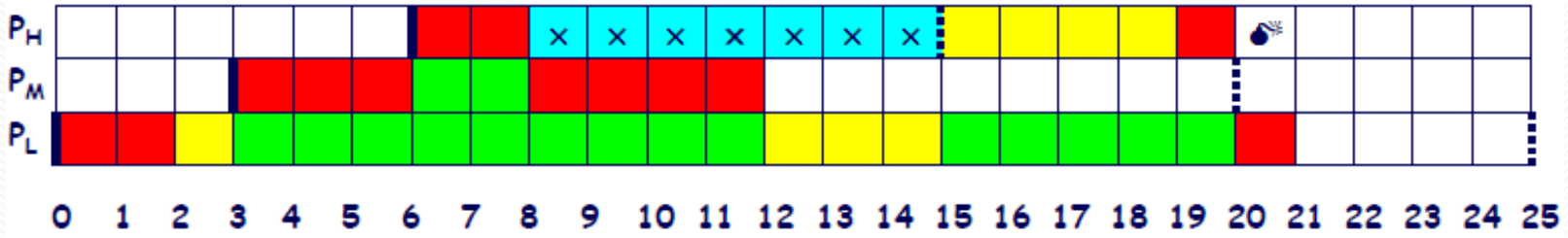
- Una tipica applicazione in tempo reale
  - N processi periodici
  - M cpu
  - Ogni processo caratterizzato da
    - Periodo
    - Deadline
    - Tempo di esecuzione
    - Possibile accesso a risorse condivise
  - Necessità di coordinare l'esecuzione dei vari thread



Java ed i sistemi in tempo reale

- Java piattaforma diffusa, ma con caratteristiche che ne limitano fortemente l'uso nei sistemi in tempo reale
  - Nessuna possibilità di caratterizzare temporalmente i thread
  - Scheduling non strettamente priority-driven.
  - Sistema soggetto ad inversioni incontrollate di priorità

	priorità	a [t.u.]	d [t.u.]	C [t.u.]
P_H	p_H (max)	6	15	7
P_M	p_M	3	20	7
P_L	p_L (min)	0	25	7



La specifica Java real-time

La specifica Java real-time (2002-2006) prevede:

- Possibilità di caratterizzare temporalmente un thread real-time;

```
th1 = new RealtimeThread(…,new
PeriodicParameters(period,deadline,deadlineMissHandler,…),
… );
```

La specifica Java real-time

La specifica Java real-time (2002-2006) prevede:

- Possibilità di caratterizzare temporalmente un thread real-time;
- Uno scheduling strettamente priority – driven;

```
th1 =
new RealtimeThread(…,new PriorityParameters(priority),… );
```

La specifica Java real-time

La specifica Java real-time (2002-2006) prevede:

- Possibilità di caratterizzare temporalmente un thread real-time;
- Uno scheduling strettamente priority – driven;
- I protocolli priorityInheritance e priorityCeiling per l'accesso a risorse condivise;

```
MonitorControl.setMonitorControl(object,
monitorControlPolicy,…);
```

La specifica Java real-time

La specifica Java real-time (2002-2006) prevede

- Possibilità di caratterizzare temporalmente un thread real-time;
- Uno scheduling strettamente priority – driven;
- I protocolli priorityInheritance e priorityCeiling per l'accesso a risorse condivise;
- Modifiche alla VM per ridurre la variabilità del tempo di esecuzione :
  - Preinizializzazione
  - Precompilazione
  - Real-Time Garbage Collection

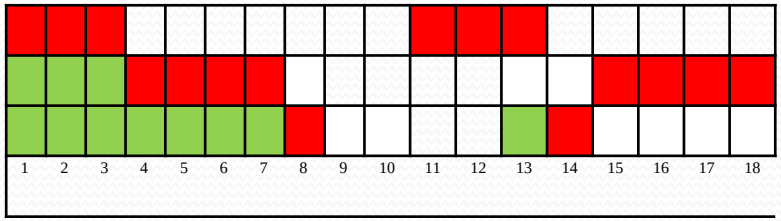
La piattaforma utilizzata

- Real-Time Java System (RTJS) versione 2.2 academic (2009) per SO Solaris 10.9, sviluppato da Sun-Oracle.
- Implementa l'ultima versione (1.02) della specifica RTSJ
- Versione gratuita per uso accademico.
- Esecuzione monocore.

Prime sperimentazioni

- Verifica dell'esecuzione di un'applicazione composta da thread periodici dal comportamento noto
- Politica di scheduling RMPO
- Necessità di modellare un'esecuzione di durata desiderata: BusyWait
- Necessità di monitorare l'attività del sistema: logging

Proc	t. Esec	periodo
T1	3	10
T2	4	14
T3	1	12



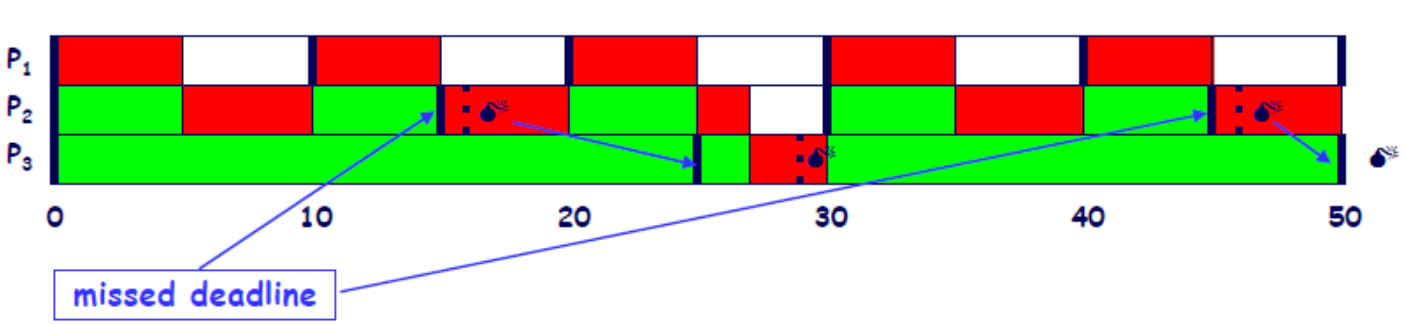

```
public void run(){
…
   for (numberOfIterations)
log.writeStartJob();
bw.busyWait(excecutionTime);
log.writeEndJob;
    waitForNextPeriod();
   }
…
}
```

Limiti di RMPO – Verso EDF

- RMPO non riesce a garantire la schedulabilità se il carico computazionale cresce

	C	T	C/T	p
P_1	5	10	0.50	max
P_2	6	15	0.40	
P_3	2	25	0.08	min

U = 0.98



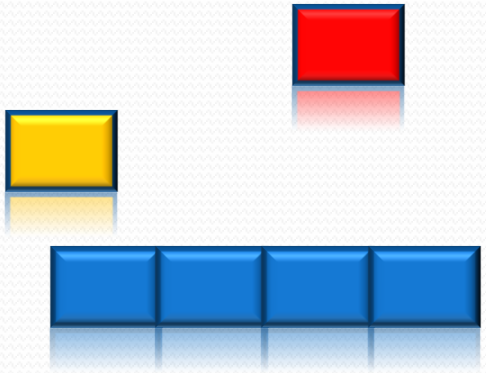
- L'applicazione sarebbe schedulabile secondo una strategia a priorità dinamica come EDF

Uno scheduler EDF per java Real-Time

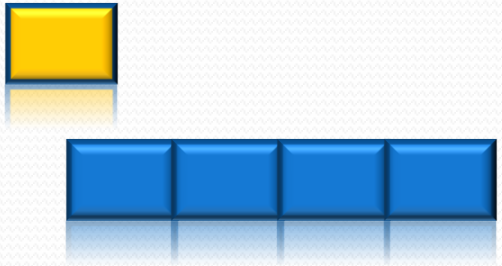
- EDF: in esecuzione job con deadline più imminente
- Tre livelli di priorità:
  - Alta
  - Media
  - Bassa
- Coda dei processi pronti ordinati secondo la prossima deadline a bassa priorità
- Il processo a deadline più imminente esegue a priorità media
- Thread eseguono prologo ed epilogo a priorità massima

Uno scheduler EDF per java Real-Time

- Il prologo:
  - Inserisce il thread nella coda o in esecuzione ponendolo alla priorità più adatta



- L'epilogo:
  - Estrae, se presente, il primo thread dalla coda e lo pone in esecuzione. Thread uscente lasciato a priorità max per non subire preemption in prossima release.

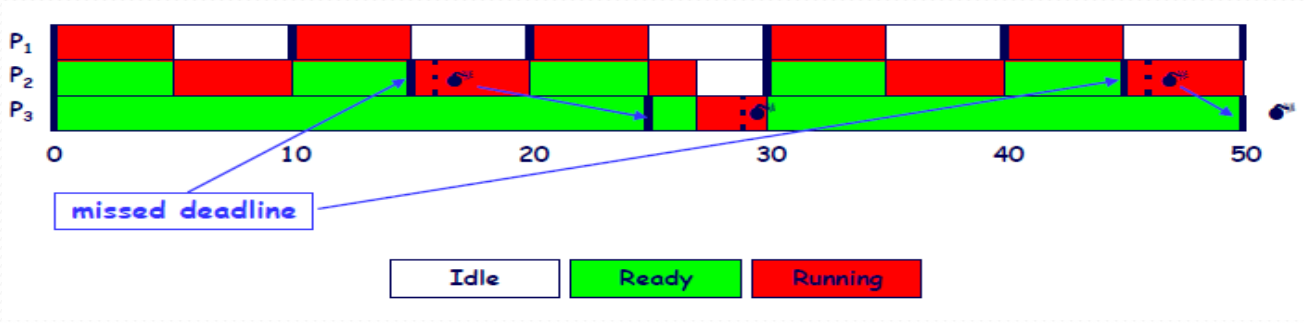


Confronto performance tra EDF e RMPO

	C	T	C/T	p
P_1	5	10	0.50	max
P_2	6	15	0.40	
P_3	2	25	0.08	min

U = 0.98

RMPO

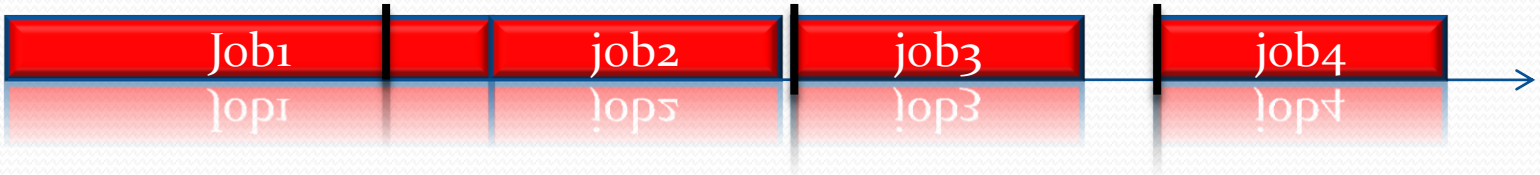


EDF

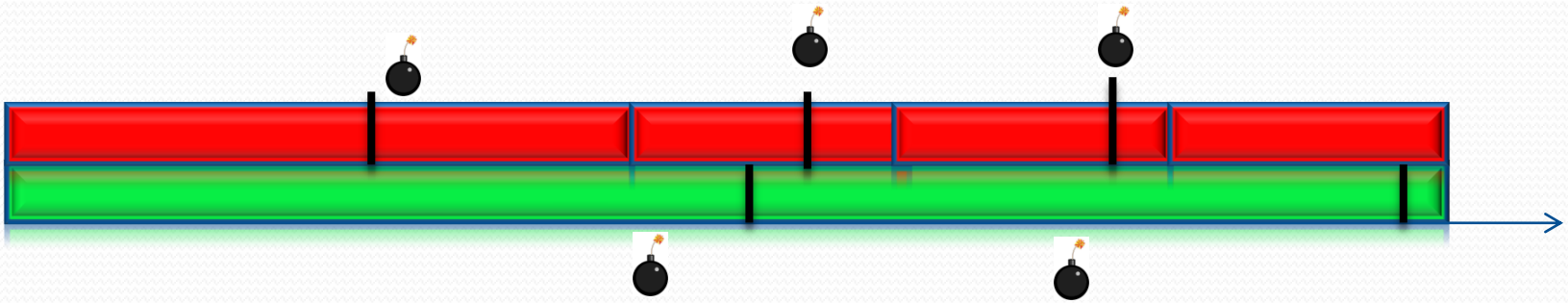
- Scheduler robusto anche in caso di accessi a risorse condivise.

La politica di default di Java real-time in caso di deadline miss

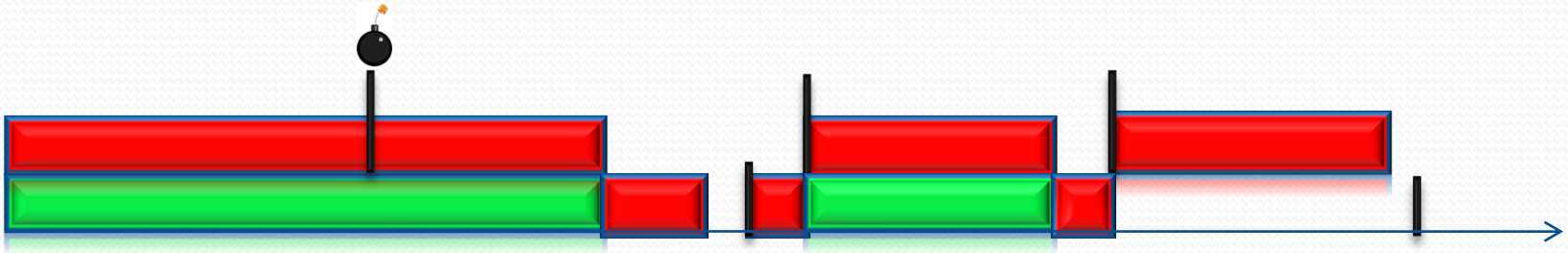
- Se non si specifica un handler, chiamata a waitForNextPeriod() non bloccante qualora sia già passato l'istante di release del prossimo Job.
- L'esecuzione del job non viene interrotta
- Il comportamento è conforme alla politica ASAP
- ASAP cerca di preservare la frequenza dell'esecuzione dei job.



Limiti della politica ASAP



- Non aiuta ad alleggerire situazioni di sovraccarico del sistema
- L'esecuzione del job può durare una quantità indefinita di tempo bloccando il sistema
- La poltica SKIP allevia il sovraccarico del sistema non schedulando altri job nel periodo successivo.



La politica di Java real-time in presenza di un handler

- In caso di deadline miss, se è presente un deadlineMissHandler nei Release Parameters chiamata di WaitForNextPeriod() bloccante finchè non viene chiamato il metodo schedulePeriodic() sul thread.
- Se handler effettua schedulePeriodic() comportamento ASAP
- Se il thread sfora più deadline consecutivamente l'handler viene chiamato ad ogni violazione

L'implementazione della politica Skip

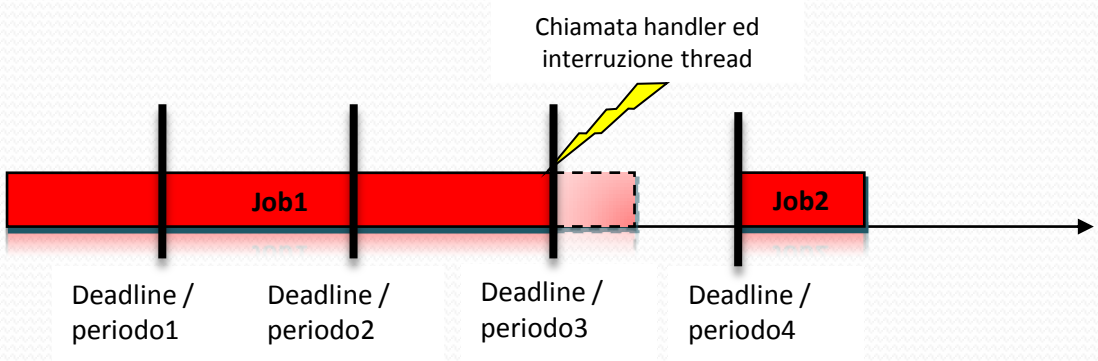
- Flag che permette al thread di differenziare release normali da quelle "di recupero".
- Codice da eseguire in release di recupero contenuto nell'handler
- SkipHandler contiene contatore delle deadline violate

```
public void run(){
…
  for (numberOfIterations){
   if (normalMode){
   log.writeStartJob();
 bw.busyWait(excexutionTime);
    logger.writeEndJob();}
else
handler.managePendingRelease();
waitForNextPeriod();
  }
…
}
```

```
onDeadlineMiss(){
  thread.setNormalMode(false);
   deadlineMissCount++;
managedThread.schedulePeriodic();
}
managePendigRelease(){
deadlineMissCount--;
If(deadlineMissCount==0)
managedThread.setNormalMode(true)
```

La politica SKIPSTOP

- Occorre evitare che un job possa eseguire per un tempo di lunghezza indefinita
- Tollerato un numero di violazioni consecutive pari ad un valore specificato
- Se si raggiunge la soglia l'esecuzione viene interrotta in modo sicuro
- I job persi non vengono recuperati



Il trasferimento asincrono di controllo

- La specifica RTSJ prevede di poter dichiarare un metodo interrompibile inserendo una AsyncronouslyInterruptedException nella throw-List del metodo

- Se si chiama il metodo interrupt() di un thread real-time mentre sta eseguendo un metodo interrompibile la AsyncronouslyInterruptedException viene lanciata e si può gestire l'interruzione nel blocco catch

Implementazione della politica SKIPSTOP

```
public void run(){
…
   for (numberOfIterations){
    if (normalMode){
    log.writeStartJob();
Try{
bw.interrumpiblebusyWait(exce
xutionTime);}
catch(AsynchronouslyInterrupt
edException e){
log.writeInterruptedThread();
}
     logger.writeEndJob();}
else
handler.managePendingRelease(
);
waitForNextPeriod();
   }
…
}
```

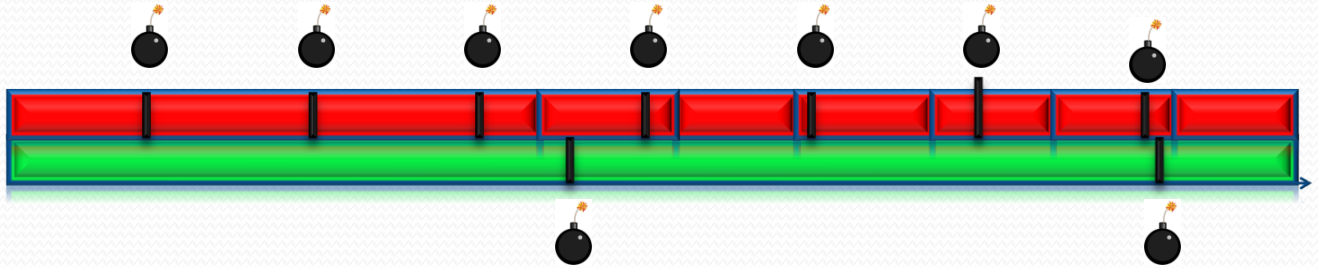
```
onDeadlineMiss(){
…
   thread.setNormalMode(false);
    deadlineMissCount++;
    if
   (deadlineMissCount>=threshlod)
       managedThread.interrupt();
managedThread.schedulePeriodic();
…
}

managePendigRelease(){
…
deadlineMissCount--;
if(deadlineMissCount==0)
 managedThread.setNormalMode(true);
…
}
```

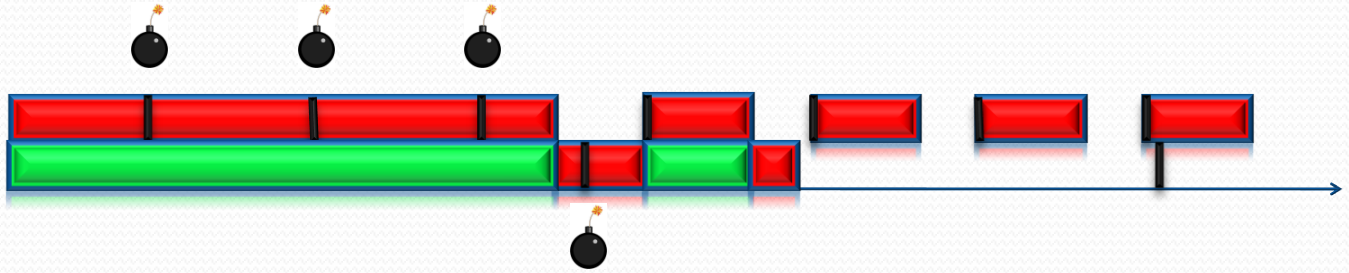
Risultati – confronto delle tre politiche

Proc	Periodo (ms)	t. Esec (ms)	Priorità
Th 1	20	15 (67)	max
Th 2	70	16	min

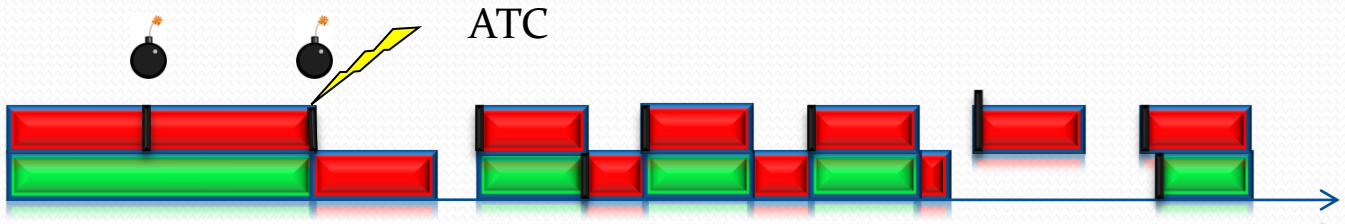
ASAP



SKIP



SKIP STOP (soglia=2)



Sviluppi futuri

- Estensione grafica del modulo di logging.
- Estensione a sistemi multiprocessore
- Estensione a insiemi di processi sporadici

- Analisi di schedulabilità
- Politiche di gestione della gestione della parte non real-time dell'applicazione.