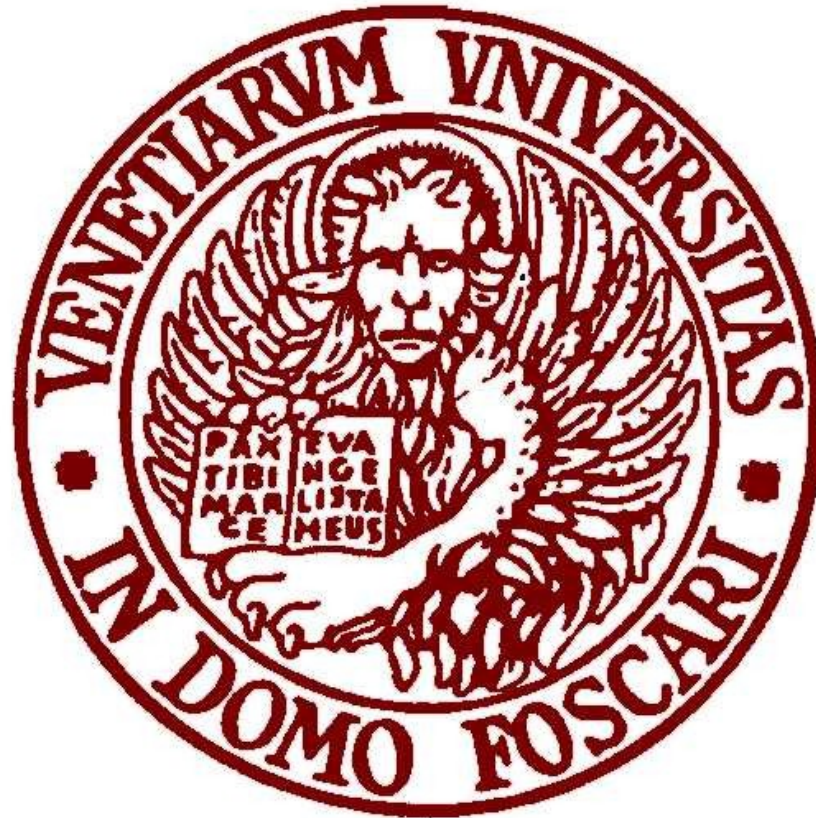


Sistemi Distribuiti Real-Time



Approfondimento del corso di Sistemi Distribuiti

Sebastiano Vascon

A.A. 2009/2010

Outline

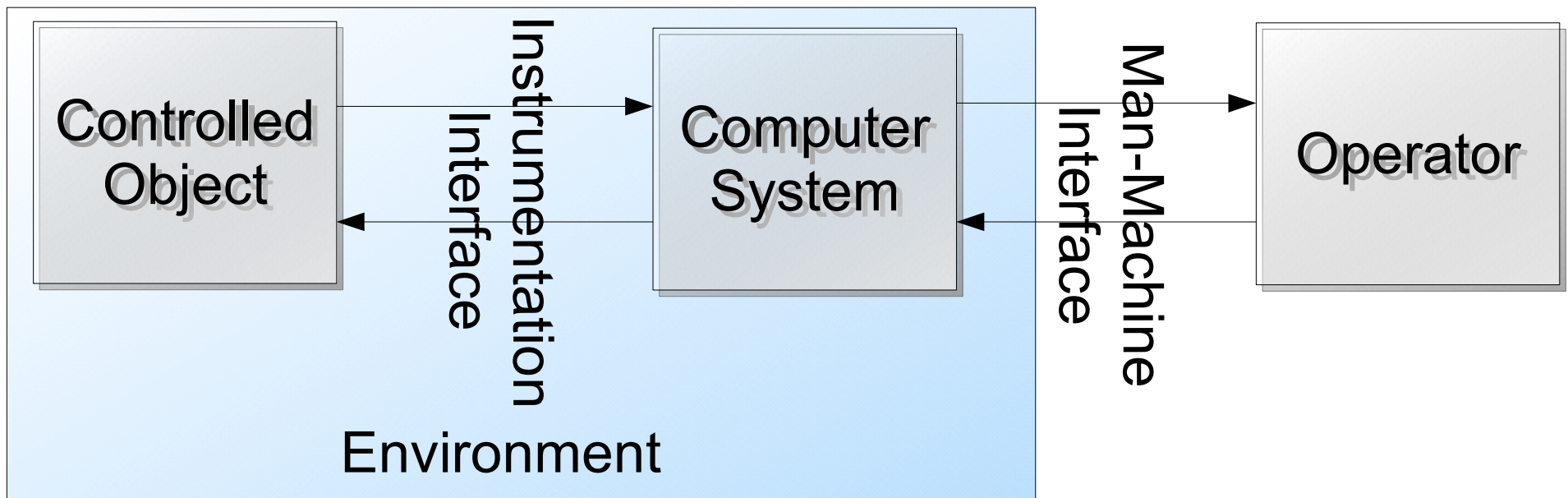
- Introduzione
- Classificazione
- Modelli
- Comunicazione
- Scheduling

Introduzione 1/3

Definizione di sistema real-time:

È un sistema che cambia il suo stato in funzione del tempo reale

Introduzione 2/3



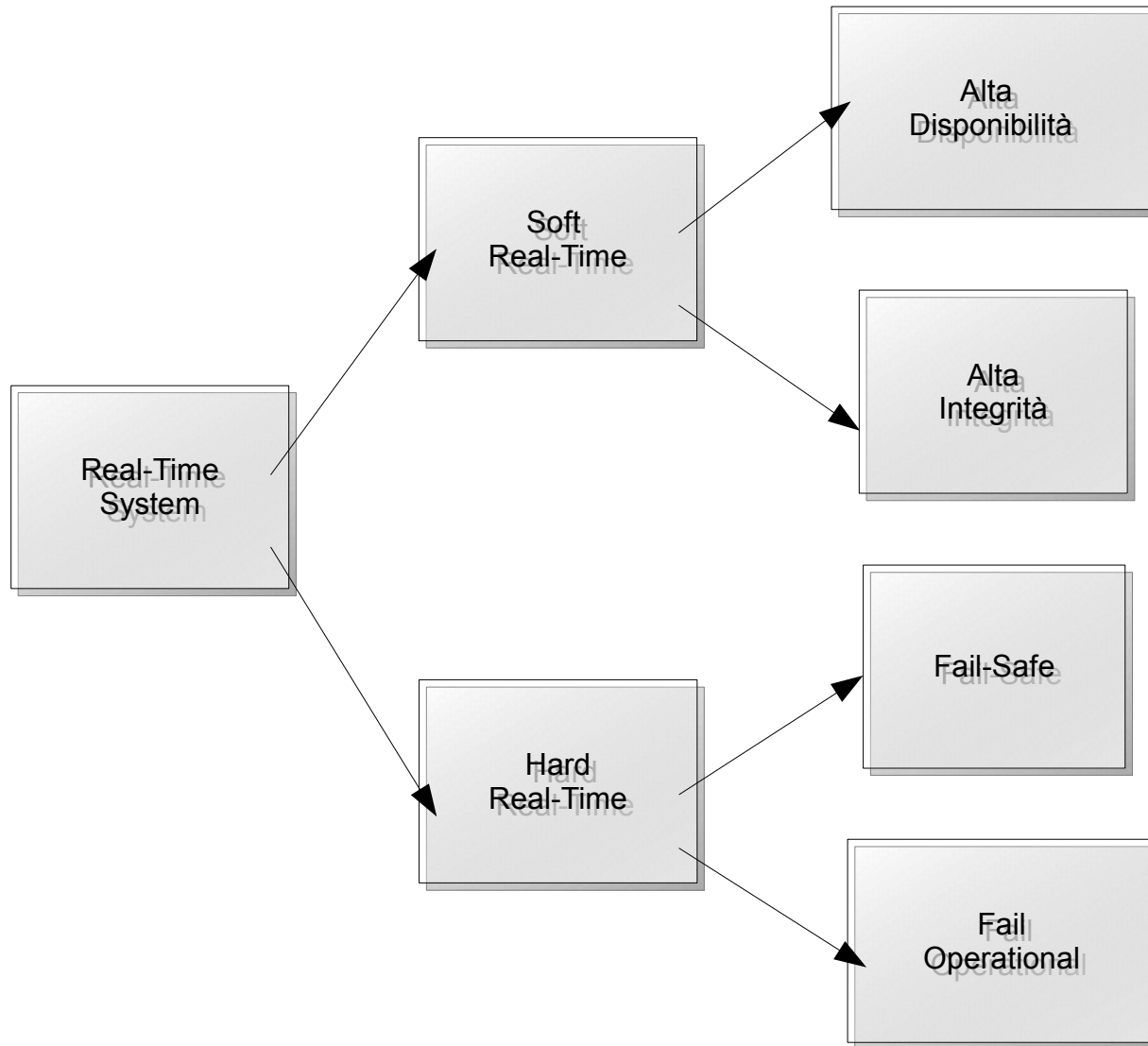
Il sistema deve reagire agli stimoli del controlled object o dell'operatore in un intervallo di tempo fissato a priori

Introduzione 3/3

Per garantire i requisiti temporali dovremo fare delle assunzioni sul sistema che vogliamo progettare:

- **Load hypothesis** (ipotesi di carico):
Definisce il traffico di picco che si assume sia generato dall'ambiente
- **Fault hypothesis** (ipotesi di fallimento)
Definisce il tipo e la frequenza dei fallimenti che il sistema deve essere in grado di gestire (fault-tolerance).

Classificazione 1/5



Classificazione 2/5

- Soft real-Time

I vincoli temporali non sono stretti e le conseguenze di un eventuale fallimento non sono disastrose

- *Alta integrità (integrity)*

I dati sono integri (non alterati) in tutto il processo comunicativo. Es sistema bancario on-line

- *Alta disponibilità (availability)*

Il sistema deve essere (teoricamente) pronto e reattivo in ogni istante. Es. Rete telefonica

Classificazione 3/5

- Hard real-time

I vincoli temporali SONO stretti e le conseguenze di un eventuale fallimento SONO disastrose

- *Fail-safe*

Il nostro sistema prevede degli stati sicuri nei quali è possibile salvarsi in caso di situazione critica. (es. Blocco di una linea ferroviaria)

- *Fail-operational*

Non esiste uno stato sicuro ma viene garantita l'operatività di base del sistema anche in caso di fallimento (es. Controllo aereo)

Classificazione 4/5

- Reactive system

Se il nostro sistema è in costante attività con il mondo esterno (consolle di un aereo)

- Embedded system

Se il nostro sistema real-time è parte di un processo più grande (sensore in un processo produttivo)

Classificazione 5/5

- Guaranteed response

Se il nostro sistema soddisfa le ipotesi e viene progettato per fornire *sempre* una risposta.

- Risorse adeguate
- Preferibile per hard real-time

- Best effort

Vengono fatte ipotesi probabilistiche sul sistema, le risorse usate sono meno e condivise.

- Preferibile per soft real-time

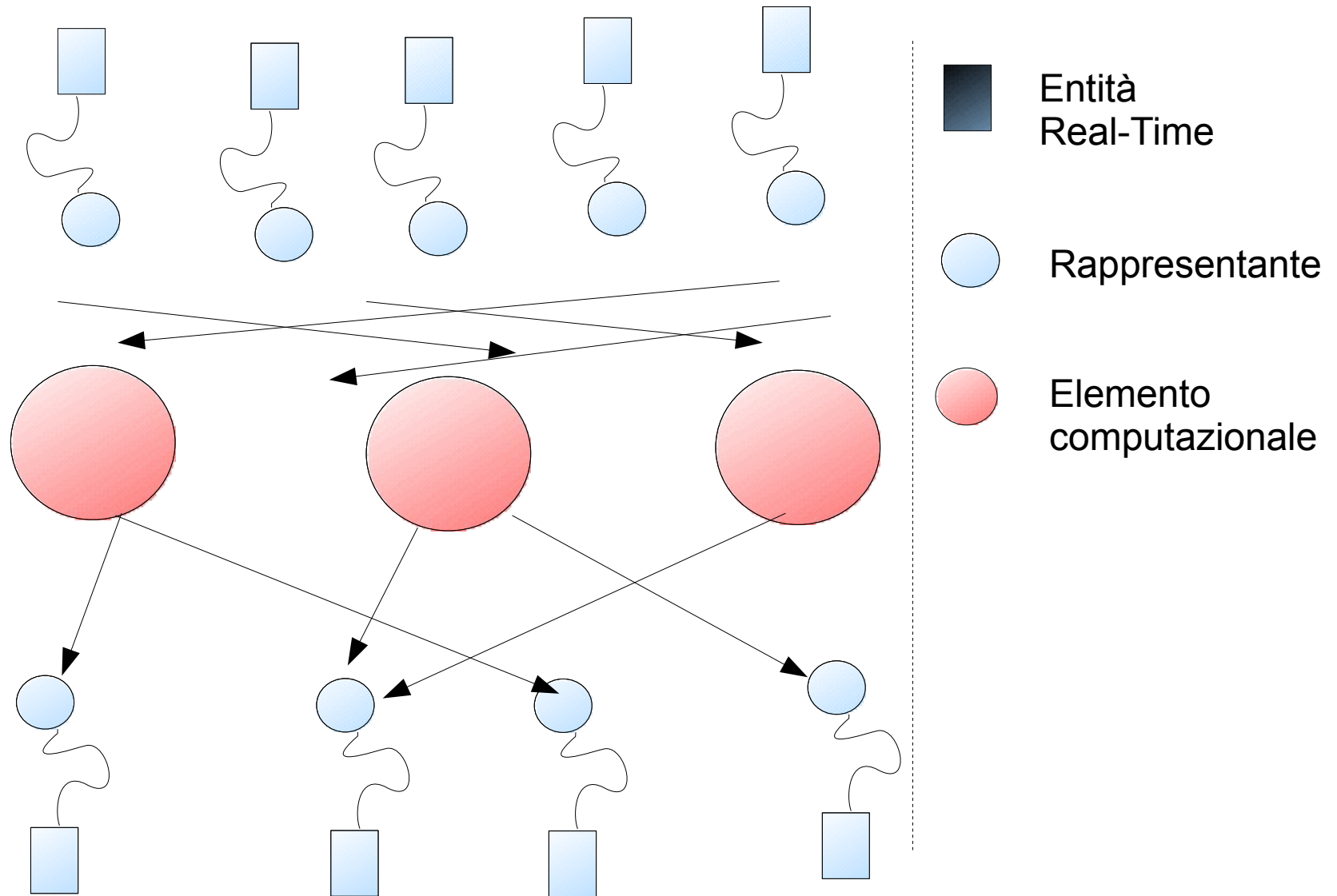
Modelli 1/6

I sistemi distribuiti real time sono essenzialmente reattivi e interagiscono con il mondo esterno.

Reagiscono agli stimoli e producono risultati sotto l'azione di specifici vincoli temporali.

Interagendo con il mondo esterno diventa necessario trovare un modello rappresentativo della realtà.

Modelli 2/6



Modelli 3/6

La progettazione di un sistema real-time si pone, dunque, due distinti obiettivi:

- la corretta percezione temporale delle entità RT
- la corretta temporizzazione dell'acquisizione delle informazioni e la relativa produzione di risposte.

Modelli 4/6

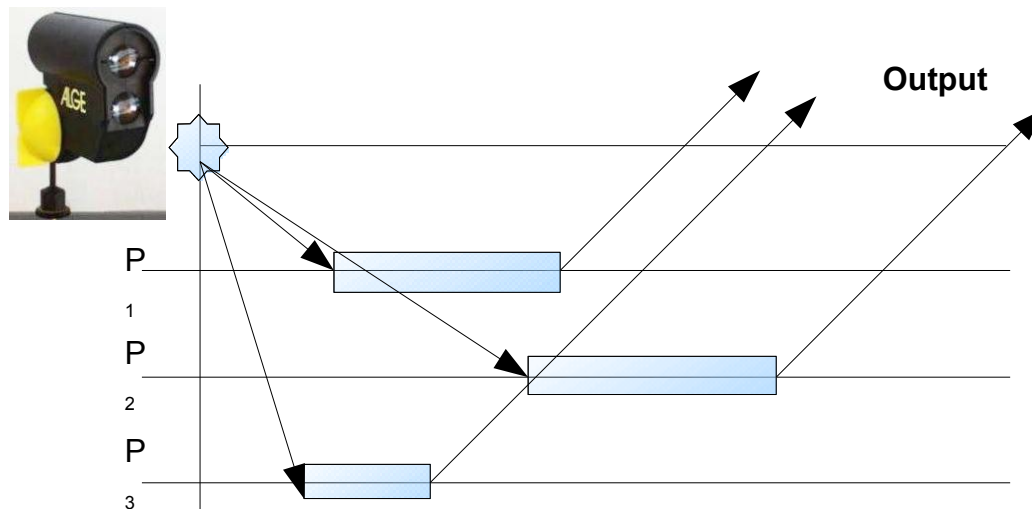
Il primo obiettivo concerne la progettazione dei collegamenti tra le entità RT e i rappresentanti. Una volta risolte le relative problematiche avremo mappato la realtà in una rappresentazione computabile.

Da questo punto il progettista proseguirà il suo lavoro ragionando esclusivamente nei termini di un sistema distribuito.

Il secondo obiettivo concerne la computazione dei dati in tempo reale.

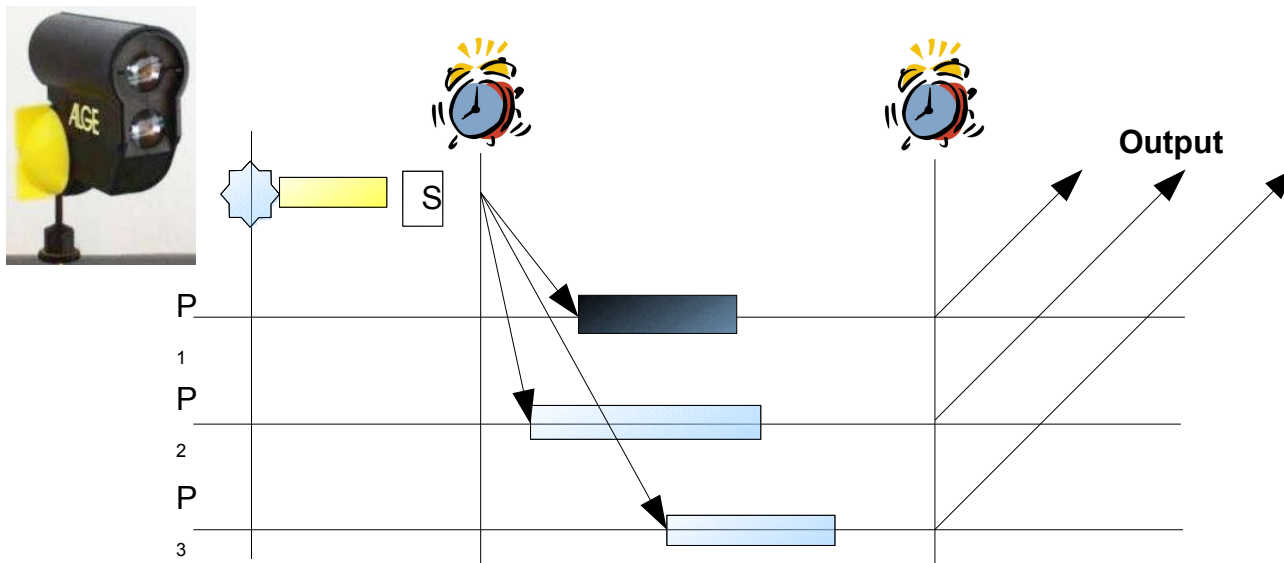
Modelli 5/6

- **Event Triggered System:**
Sono sistemi che reagiscono direttamente e immediatamente ad eventi esterni connessi al sistema (*problema event-shower*).



Modelli 6/6

- Time Triggered System:
l'ambiente e la sua evoluzione sono sottoposti a delle ben definite assunzioni di temporizzazione (non esistono eventi asincroni).



Comunicazione 1/2

Un sistema distribuito real-time deve prevedere anche un sistema di comunicazione real-time per permettere lo scambio di informazioni tra le sue componenti, sono necessari:

- Protocolli real-time per lo scambio dei messaggi (es. RTP/RTCP)
- Infrastruttura di rete real-time che evidenzia certe proprietà temporali (ritardo di propagazione) e di affidabilità

Comunicazione 2/2

In particolare dell'infrastruttura di rete devono essere noti:

- Ritardo nella consegna di un messaggio
- Ampiezza di banda necessaria
- Caratteristiche della microrete
- Riconoscimento di messaggi urgenti e quindi l'instaurazione di vie preferenziali
- Connettività, in quanto il mondo reale non attende che si risolva, ad esempio, un blackout

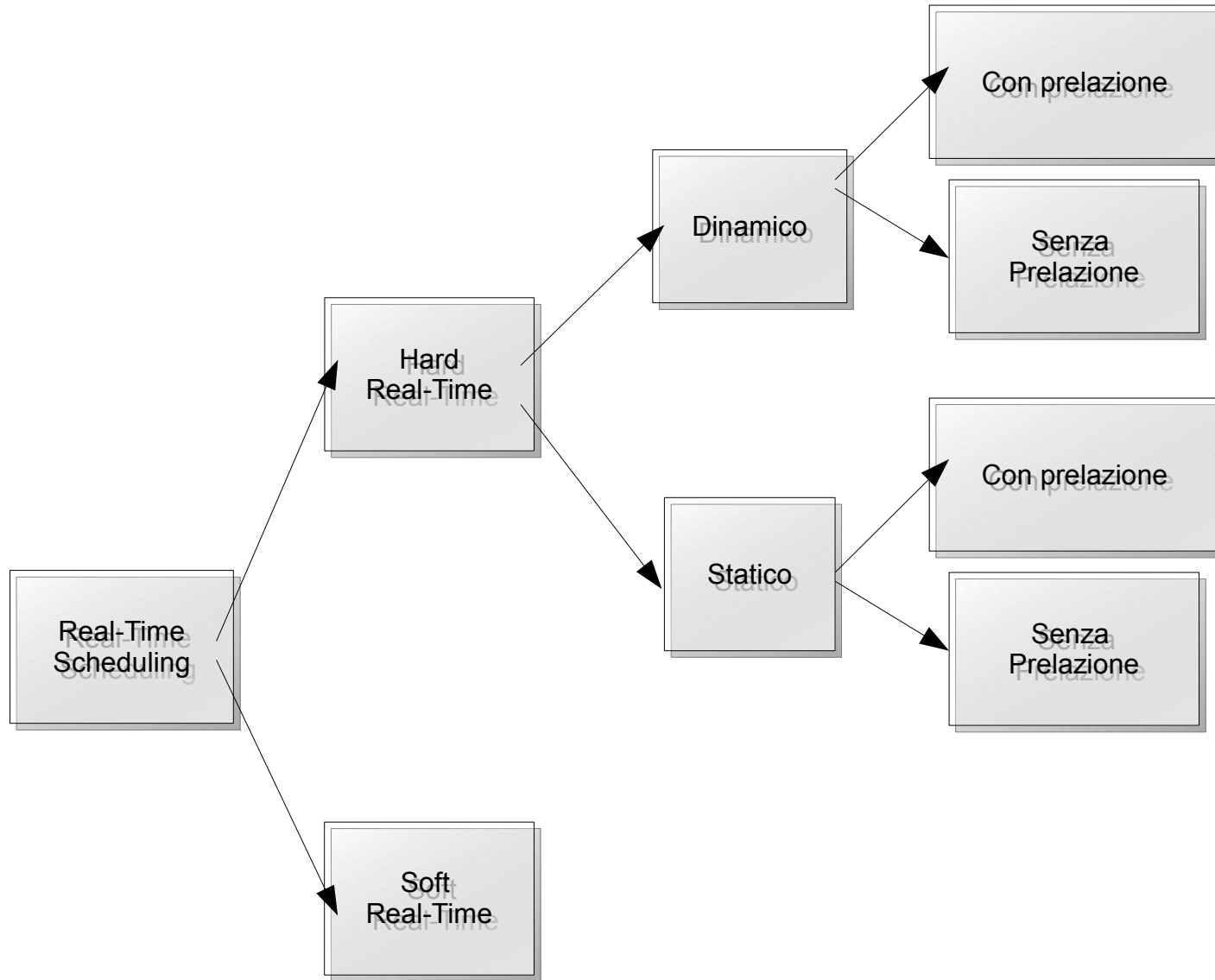
Scheduling 1/5

Parte fondamentale del processo real-time è lo scheduling delle operazioni da eseguire.

Oltre a garantire la correttezza sul risultato, bisogna anche garantire il tempo di esecuzione.

Quindi nei criteri decisionali degli scheduler per sistemi real-time si dovrà tenere conto dei vincoli temporali e della tipologia di sistema

Scheduling 2/5



Scheduling 3/5

- *Scheduler Statico*

Le decisioni sono prese a off-line e non vengono modificate con l'evoluzione del sistema. Questo prevede una profonda conoscenza dell'ambiente.

- *Scheduler Dinamico*

Le decisioni sono prese a on-line e vengono modificate con l'evoluzione del sistema.

Scheduling *Dinamico* 4/5

Assumendo che i task siano **indipendenti**

- *Rate monotonic* si assume che:

I task hard real-time siano periodici, la scadenza di ogni task (T_i) corrisponde alla durata del suo periodo (p_i), conosciamo a priori il tempo di computazione massimo per ogni task.

L'algorithmo assegna le priorità in base alla durata del periodo del task da eseguire. I task con periodo più corto avranno priorità più alta.

- *Earliest-deadline-first*

Viene selezionato il task con deadline più prossima

- *Least-laxity*

Viene selezionato il task con laxity minore. (Laxity è la differenza tra la deadline e il tempo di computazione)

Scheduling *Dinamico* 5/5

Assumendo che i task siano **dipendenti**

- *Kernelized monitor*

*Si alloca il processore per un quanto di tempo q non interrompibile inoltre si assume che una sezione critica sia eseguibile (quindi inizi e si concluda) all'interno del quanto di tempo. Viene usata per il task scheduling la politica **earliest deadline first**. Può portare ad un basso utilizzo del processore, in quanto si tara il tempo q sulla sezione critica più lunga.*

- *Priority ceiling protocol*

Viene utilizzato per schedulare un insieme periodico di task che hanno accesso esclusivo ad una o più risorse comuni protette da semafori.

Ideato per risolvere il problema dell'inversione di priorità

Esempi

- Automazione in impianti
- Telefonia
- Reti di sensori
- Centrali nucleari
- Reti ferroviarie
- Controllo aereo

Bibliografia

- [1] Sape Mullender, Distributed Systems, Addison-Wesley, 1994
- [2] Mok, A.K., Fundamental Design Problems of Distributed System for Hard Real-Time Environment, 1993
- [3] Coulouris – Dollimore – Kindberg , Distributed System Concept and Design 4ed. , Addison Wesley , 2009
- [4] L. Galli, Distributed Operating Systems – Concept & Practice , Prentice Hall , 1999

Licenza

Info

Il contenuto di questo documento è da ritenersi “as is” e non mi assumo la responsabilità per eventuali errori sia di forma che di contenuto. Se doveste trovare imprecisioni o errori siete pregati di informarmi e provvederò ad effettuare le dovute correzioni.

Creative Common License:

Sei libero di:

riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire, recitare e modificare quest'opera.

Alle seguenti condizioni:

Devi attribuire questo lavoro a Sebastiano Vascon (indicando questo link <http://sebastiano.vascon.it>). Non puoi usare quest'opera per fini commerciali Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.

Ulteriori dettagli, che vi consiglio di leggere, circa la licenza Creative Common li trovate a questo link: <http://creativecommons.org/licenses/by-nc-sa/2.5/it/>