



Criteri di sincronizzazione

## Criteri di sincronizzazione



Anno accademico 2015/16  
Sistemi Concorrenti e Distribuiti

Tullio Vardanega, [tullio.vardanega@math.unipd.it](mailto:tullio.vardanega@math.unipd.it)

Laurea Magistrale in Informatica, Università di Padova 1/21



Criteri di sincronizzazione

## Criteri di valutazione – 2

- È interessante applicare questo schema di valutazione ai costrutti di sincronizzazione
  - 1979, Toby Bloom, "Evaluating synchronisation mechanisms", Proc. 7<sup>th</sup> ACM Symposium on Operating System Principles, pp. 24-32
- Bloom identifica 6 tipi di vincoli influenti sulla sincronizzazione
  - Per requisiti di potere espressivo superiori alla *exclusion synchronization* come la conosciamo finora

Laurea Magistrale in Informatica, Università di Padova 3/21



Criteri di sincronizzazione

## Criteri di valutazione – 1

- Un linguaggio e i suoi costrutti possono essere valutati sotto due criteri generali
- Potere espressivo → criterio oggettivo
  - La capacità del linguaggio di soddisfare i bisogni del dominio applicativo
- Usabilità → criterio soggettivo
  - Il grado di interazione (efficacia) e di integrazione (coerenza) dei costrutti in esame, tra di loro e verso il resto del linguaggio [in dati contesti d'uso]

Laurea Magistrale in Informatica, Università di Padova 2/21



Criteri di sincronizzazione

## Condizioni di sincronizzazione – 1

- In funzione del tipo di richiesta
  - E.g., meglio letture che scritture
  - Espressa con canali tipati e guardie
- In funzione del tempo della richiesta
  - Espressa con politiche di accodamento sulle code di guardia o di ordinamento (*scheduling*) dei clienti
- In funzione dei parametri della richiesta 
  - E.g., algoritmo dell'ascensore

Laurea Magistrale in Informatica, Università di Padova 4/21



**Criteri di sincronizzazione**

## Condizioni di sincronizzazione – 2

- ❑ **In funzione dello stato di sincronizzazione della risorsa** 
  - E.g., #utenti correnti per molteplicità >1 oppure # richieste in attesa (`Count`)
- ❑ **In funzione dello stato logico interno della risorsa** 
  - E.g., *buffer is empty vs. buffer is full*
- ❑ **In funzione della storia d'esecuzione nella risorsa**

Laurea Magistrale in Informatica, Università di Padova5/21



**Criteri di sincronizzazione**

## L'allocazione delle risorse – 2

- ❑ **Nell'esempio, il volume di richiesta di un cliente appare come parametro in ingresso sul canale tipato**
- ❑ **Per leggere il parametro (e valutare se la richiesta è soddisfacibile) occorre però prima accettare la sincronizzazione**
  - Che fare se la richiesta non fosse soddisfacibile al momento?
  - Di sicuro non vogliamo ricorrere all'uso di "busy wait"
  - Questione di usabilità prima che di potere espressivo 

Laurea Magistrale in Informatica, Università di Padova7/21



**Criteri di sincronizzazione**

## L'allocazione delle risorse – 1

- ❑ **Problema chiave e ricorrente in ogni modello di programmazione concorrente**
  - Coinvolge tutte le dimensioni identificate da Bloom
  - Particolarmente difficile da risolvere solo con guardie

Esempio: Si costruisca un controllore che debba allocare risorse a un gruppo di clienti, ove le risorse siano in numero finito e i clienti siano in competizione tra loro per averle. Ciascun cliente  $C_i$  può richiedere  $N_i \geq 1$  risorse alla volta. Il protocollo di assegnazione sancisce che la richiesta, se accettata, debba essere soddisfatta integralmente, altrimenti sia tenuta in sospenso e il cliente bloccato sino a quando ne diventi possibile il soddisfacimento.

Laurea Magistrale in Informatica, Università di Padova6/21



**Criteri di sincronizzazione**

## L'allocazione delle risorse – 3

- ❑ **L'uso delle guardie di Dijkstra abilita l'uso di *avoidance synchronization***
  - Evitare sincronizzazione ove questa non sia utile nello stato logico corrente della risorsa, prima di valutare la richiesta
- ❑ **L'uso delle variabili di condizione del *monitor* di Hoare supporta attesa condizionale all'interno della sincronizzazione**
  - Sembra che ci serva: prima valutare il parametro e poi, se necessario, imporre attesa
  - Ma il limite strutturale del *monitor* è proprio richiedere programmazione esplicita dell'attesa (!)

Laurea Magistrale in Informatica, Università di Padova8/21



Criteri di sincronizzazione

**L'allocazione delle risorse – 4**

- ❑ **Serve estendere il protocollo base di *avoidance synchronization***
- ❑ **Ci sono almeno due soluzioni**
  - **Consentire all'espressione di guardia di accedere ai parametri in ingresso della richiesta**
    - Lo fa il linguaggio **SR** (*Synchronizing Resources*)  
<http://www.cs.arizona.edu/sr/>
  - **Consentire al processo servente di trasferire il cliente accettato ma non soddisfacibile, su altra coda d'attesa**
    - Dove il cliente attenderà il verificarsi di condizioni più propizie
    - Consentendo così al canale di poter accogliere nuove richieste



Laurea Magistrale in Informatica, Università di Padova

**9/21**



Criteri di sincronizzazione

**L'allocazione delle risorse – 6**

- ❑ **Il difetto principale dell'approccio SR è il suo eccessivo costo realizzativo**
  - **Legando sincronizzazione e accodamento di chiamata al valore di parametri della richiesta rispetto allo stato della risorsa, tutte le richieste accodate devono essere rivalutate ogni volta che lo stato della risorsa cambi (!)**
    - Una sorta di «Notify All» in peggio
- ❑ **Assai preferibile è trasferire ad altra coda la richiesta accettata ma non soddisfacibile**
  - **Senza dovere rivalutare l'eventuale guardia, la cui espressione potrà essere specializzata per la condizione di riaccodamento**
  - **Trasferendo la richiesta con singola operazione atomica**



Laurea Magistrale in Informatica, Università di Padova

**11/21**



Criteri di sincronizzazione

**L'allocazione delle risorse – 5**

**Forma base (1 risorsa per richiesta)**

```
protected Controller is
entry Allocate (R : out Resource);
procedure Release (R : Resource);
private
Free : Natural := Full_Capacity;
...
end Controller;
protected body Controller is
entry Allocate (R : out Resource)
when Free > 0 is
begin
Free := Free - 1;
...
end Allocate;
procedure Release (R : Resource) is
begin
Free := Free + 1;
end Release;
end Controller;
```

**Soluzione SR per il nostro problema**

```
type Request is range 1..Max_Requests;
protected Controller is
entry Allocate
(R : out Resource;
Amount : in Request);
procedure Release
(R : Resource;
Amount : Request);
private
Free : Request := Request_Last; ...
end Controller;
protected body Controller is
entry Allocate
(R : out Resource;
Amount : in Request)
when Amount <= Free is
begin
Free := Free - Amount;
end Allocate;
procedure Release (...) is ...
end Controller;
```

La guardia usa un parametro 'in' dell'entry →

Laurea Magistrale in Informatica, Università di Padova

**10/21**



Criteri di sincronizzazione

**L'allocazione delle risorse – 7**

- ❑ **Esempio 7 (lezione 10): proviamo il potere espressivo del «trasferimento di coda» ...**
  - **Esercizio**  
**Migliorare la soluzione proposta evitando di rivalutare inutilmente la coda sul canale interno Assign (quando le risorse disponibili non fossero ancora disponibili per soddisfare alcuna richiesta pendente)**

Laurea Magistrale in Informatica, Università di Padova

**12/21**

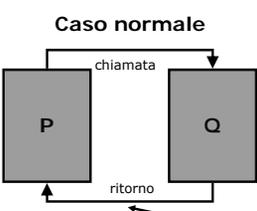


Criteri di sincronizzazione

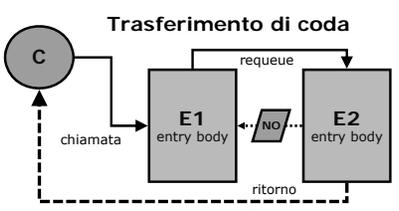
**Semantica del trasferimento di coda – 1**

**Il trasferimento di coda (*requeue*) non è una normale chiamata di procedura**

Caso normale



Trasferimento di coda



Quando E1 esegue *requeue* su E2, E1 viene finalizzato e lasciato

Laurea Magistrale in Informatica, Università di Padova

**13/21**



Criteri di sincronizzazione

**Semantica del trasferimento di coda – 3**

**1. Verso quale coda permettere trasferimento**

- La coda di destinazione può essere di qualsiasi *entry* (sia di processo che di risorsa protetta)
  - Per maggior coesione funzionale conviene restare nell'entità di partenza
- Il trasferimento verso coda di altra entità causa necessariamente il rilascio dell'entità di partenza
  - Il che potrebbe non essere desiderabile
- Il canale destinazione deve essere compatibile con quello di partenza
  - Quindi o esattamente gli stessi della chiamata iniziale oppure nessuno
- La direzione del trasferimento (interna o esterna) determina il destinatario finale del "lucchetto" originale

Laurea Magistrale in Informatica, Università di Padova

**15/21**



Criteri di sincronizzazione

**Semantica del trasferimento di coda – 2**

**Permettere il trasferimento di coda in modo programmatico solleva due domande delicate**

1. Verso quali code di canale (*entry*) permetterlo
2. Come trattare il *time-out* eventualmente posto dal cliente sulla sua invocazione iniziale

Laurea Magistrale in Informatica, Università di Padova

**14/21**



Criteri di sincronizzazione

**Semantica del trasferimento di coda – 4**

**2. Come trattare il *time-out* posto dal cliente sulla richiesta**

- Il trasferimento deve indicare esplicitamente se applicarlo al canale destinazione
- Oppure considerarlo soddisfatto con l'attuale accettazione

Laurea Magistrale in Informatica, Università di Padova

**16/21**

Criteri di sincronizzazione

### Semantica del trasferimento di coda – 3

Cosa succede in questo caso?

Vi sono 2 possibilità:

1. La chiamata B.E1 **non** viene accolta entro il tempo T1 → **chiamata annullata**
2. La chiamata viene accolta in E1, dove esegue per un tempo T2, ma verrà annullata se E2 non venisse accolta entro  $T2+T1$

```

-- A
select
B.E1;
or
delay T1;
end select;

-- B
select
accept E1 do
... -- T2 time units
requeue E2 with abort;
end E1;
or
...
end select;

```

La conseguenza è una distorsione temporale della tolleranza richiesta

Questa clausola *preserva* l'eventuale *time-out* posto dal chiamante sulla coda di destinazione

Laurea Magistrale in Informatica, Università di Padova

17/21

Criteri di sincronizzazione

### Esempio 8 /1

- ❑ La possibilità di trasferimento tra code consente grande potere espressivo che aiuta a rappresentare situazioni complesse
- ❑ Vogliamo simulare il comportamento di un sistema di trasporto viaggiatori su linea circolare
  - N stazioni su linea circolare → una risorsa protetta per stazione, presso la quale ciascun viaggiatore in partenza si blocca in attesa del treno
  - 1 treno a capienza finita → entità attiva
  - K viaggiatori che si recano alla loro stazione di partenza avendo una stazione di arrivo → un'entità attiva per ogni viaggiatore

Laurea Magistrale in Informatica, Università di Padova

19/21

Criteri di sincronizzazione

### Esercizio

Sull'utilità del trasferimento verso altra coda

Un *router* di rete può instradare pacchetti in ingresso verso 3 linee di comunicazione  $L_{a-c}$ , distinte ma funzionalmente equivalenti. La linea  $L_a$  rappresenta la scelta preferenziale, ma le altre linee (prima  $L_b$  e poi  $L_c$ ) sono usate quando la precedente alternativa risulti sovraccarica. Si realizzi un programma concorrente che realizzi questa politica, considerando il diagramma qui riportato come possibile fonte di ispirazione

Laurea Magistrale in Informatica, Università di Padova

18/21

Criteri di sincronizzazione

### Esempio 8 /2

- ❑ Il trasferimento di coda è un modo pratico di simulare il trasporto dei viaggiatori
  - Quando il treno arriva in una stazione il programma trasferisce i viaggiatori in attesa in quella stazione (ma solo fino alla capienza massima del treno) nella coda della loro stazione di destinazione, dalla quale saranno rilasciati (= arrivati) quando il treno vi farà sosta

Laurea Magistrale in Informatica, Università di Padova

20/21

