



Criteria di sincronizzazione

## Criteria avanzati di sincronizzazione



Anno accademico 2018/19  
Sistemi Concorrenti e Distribuiti

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

Laurea Magistrale in Informatica, Università di Padova1/20



Criteria di sincronizzazione

## Valutazione critica – 2

- È utile 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
- Quel lavoro identifica 6 tipi di vincoli influenti sulla sincronizzazione
  - Con potere espressivo superiore alla *exclusion synchronization*

Laurea Magistrale in Informatica, Università di Padova3/20



Criteria di sincronizzazione

## Valutazione critica – 1

- Un linguaggio e i suoi costrutti possono essere valutati sotto due criteri generali
- Potere espressivo
  - La capacità di soddisfare bisogni applicativi
- Usabilità
  - Il grado di interazione (efficacia) e di integrazione (coerenza) dei costrutti tra di loro e verso il resto del linguaggio

Laurea Magistrale in Informatica, Università di Padova2/20



Criteria di sincronizzazione

## Condizioni di sincronizzazione – 1

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

Laurea Magistrale in Informatica, Università di Padova4/20



**Criteri di sincronizzazione**

## Condizioni di sincronizzazione – 2

- In funzione del tipo di richiesta**
  - P.es., meglio letture che scritte
  - Espressa con canali tipati e guardie
- In funzione del tempo della richiesta**
  - Espressa con politiche di accodamento sulle code di guardia o di ordinamento dei clienti
- In funzione dei parametri della richiesta** 
  - P.es., algoritmo dell'ascensore

Laurea Magistrale in Informatica, Università di Padova5/20



**Criteri di sincronizzazione**

## L'allocazione delle risorse – 2

- Nel problema dato, il volume di richiesta di un cliente viene presentato 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 Padova7/20



**Criteri di sincronizzazione**

## L'allocazione delle risorse – 1

- Problema ricorrente in ogni modello di programmazione concorrente**
  - Coinvolge tutte le 6 dimensioni di interesse
  - Particolarmente difficile da trattare con sole guardie

**Esempio:** costruire un controllore per allocare un numero finito  $N$  di risorse a un insieme di clienti  $C_i$  concorrenti. Ciascun cliente  $C_i$  può richiedere  $1 \leq N_i \leq N$  risorse alla volta. Il protocollo di assegnazione sancisce che, se accettata, la richiesta 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 Padova6/20



**Criteri di sincronizzazione**

## L'allocazione delle risorse – 3

- L'uso delle guardie abilita l'uso di *avoidance synchronization***
  - Per evitare sincronizzazione ove questa non sia utile nello stato logico corrente della risorsa (e prima di valutare la richiesta)
- L'uso delle variabili di condizione del *monitor* di Hoare supporta attesa condizionale (*wait, signal*) all'interno della sincronizzazione**
  - Sembra che ciò che vogliamo: prima valutare il parametro e poi eventualmente imporre attesa
  - Ma il limite strutturale del *monitor* è proprio richiedere programmazione esplicita dell'attesa 

Laurea Magistrale in Informatica, Università di Padova8/20

**Criteri di sincronizzazione**

**L'allocazione delle risorse – 4**

- ❑ **Conviene estendere il protocollo base di *avoidance synchronization***
- ❑ **Vi sono almeno due soluzioni**
  - **L'espressione di guardia accede ai parametri in ingresso della richiesta**
    - Come nel linguaggio **SR** (*Synchronizing Resources*)  
<http://www.cs.arizona.edu/sr/>
  - **Il processo servente trasferisce la chiamata (accettata ma non soddisfacibile), su altra coda**
    - 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

**9/20**

**Criteri di sincronizzazione**

**L'allocazione delle risorse – 6**

- ❑ **Il fiasco dell'approccio SR è il suo eccessivo costo**
  - **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 `notifyAll()` in peggio
- ❑ **Assai meglio 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

**11/20**

**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;
                    
```

La guardia usa un parametro 'in' dell'entry

**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;
                    
```

Laurea Magistrale in Informatica, Università di Padova

**10/20**

**Criteri di sincronizzazione**

**Semantica del trasferimento di coda – 1**

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

**Chiamata di procedura**

**Trasferimento di coda**

Quando E1 esegue `requeue` su E2, E1 viene finalizzato e lasciato; si ritorna a C solo dopo l'esito di E2

Laurea Magistrale in Informatica, Università di Padova

**12/20**

UnipD - SCD 2018/19 - Sistemi Concorrenti e Distribuiti



Criteri di sincronizzazione

**Semantica del trasferimento di coda – 2**

❑ **Permettere il trasferimento di coda in modo programmatico pone 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

13/20



Criteri di sincronizzazione

**Semantica del trasferimento di coda – 4**

**2. Come trattare il *time-out* eventualmente 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

15/20



Criteri di sincronizzazione

**Semantica del trasferimento di coda – 3**

**1. Verso quale coda permettere trasferimento**

- **Qualsiasi *entry* (di server, di RP)**
  - Per maggior coesione funzionale conviene restare nell'entità di partenza
- **Il trasferimento verso coda di altra entità causa il rilascio dell'entità di partenza**
  - Necessario, ma potenzialmente indesiderabile
- **Il canale destinazione deve avere interfaccia compatibile con quello di partenza**
  - LSP: la stessa della chiamata iniziale oppure vuota
- **La direzione del trasferimento (interna o esterna) determina il destinatario finale del *lock* originale**

Laurea Magistrale in Informatica, Università di Padova

14/20



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

16/20



Criteri di sincronizzazione

**Esempi d'uso – 1**

- ❑ **Algoritmo di allocazione di risorse con trasferimento di coda**
- ❑ **Vedere soluzione base nell'esempio associato alla lezione di oggi**
- ❑ **Migliorarla evitando trasferimenti inutili**
  - **Come? Esercizio** 

Laurea Magistrale in Informatica, Università di Padova

17/20



Criteri di sincronizzazione

**Esempi d'uso – 3**

- ❑ **Il trasferimento di coda offre grande potere espressivo per trattare 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

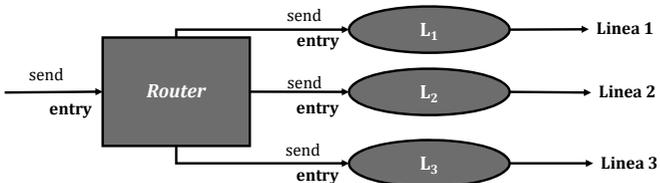
19/20



Criteri di sincronizzazione

**Esempi d'uso – 2**

Un *router* di rete può instradare pacchetti in ingresso verso  $N = 3$  linee di comunicazione  $L_{1,3}$ , distinte ma funzionalmente equivalenti. La linea  $L_1$  rappresenta la scelta preferenziale, ma le altre linee (prima  $L_2$  e poi  $L_3$ ) sono usate quando la precedente alternativa risulti sovraccarica. Per realizzare questa soluzione il trasferimento di coda avverrebbe tra entità *distinte* (da Router verso  $L_i$ )



Laurea Magistrale in Informatica, Università di Padova

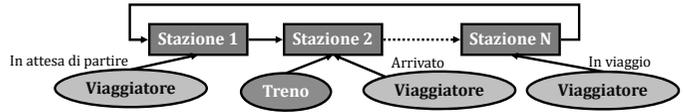
18/20



Criteri di sincronizzazione

**Idea di soluzione**

- ❑ **Il trasferimento di coda è un modo pratico di simulare il trasporto dei viaggiatori**
  - **L'arrivo del treno in una stazione abilita il trasferimento dei viaggiatori in attesa in essa (fino alla capacità massima del treno) nella coda della stazione di loro destinazione, dalla quale saranno rilasciati (= arrivati) all'arrivo del treno**



Laurea Magistrale in Informatica, Università di Padova

20/20