



Criteri di sincronizzazione

## Criteri di sincronizzazione

SCD

Anno accademico 2010/11  
Sistemi Concorrenti e Distribuiti

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

Corso di Laurea Magistrale in Informatica, Università di Padova
1/19



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 direttamente 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
- **È interessante applicare questo schema di valutazione anche ai costrutti di sincronizzazione**
  - 1979, Toby Bloom, "Evaluating synchronisation mechanisms", Proc. 7<sup>th</sup> ACM Symposium on Operating System Principles, pp. 24-32

Corso di Laurea Magistrale in Informatica, Università di Padova
2/19



Criteri di sincronizzazione

## Criteri di valutazione – 2

- **Bloom propone criteri di valutazione molto superiori a *exclusion synchronization***
  - 1) **Preferenza in base al tipo di servizio richiesto**
    - Canale tipato
  - 2) **Preferenza in base al tempo relativo delle richieste (o la priorità del cliente)**
    - Politiche di accodamento
  - 3) **Preferenza in base ai parametri della richiesta**
  - 4) **Preferenza in base allo stato e storia d'uso della risorsa e il suo stato di sincronizzazione corrente**
    - Guardie [*avoidance synchronization*]

Corso di Laurea Magistrale in Informatica, Università di Padova
3/19



Criteri di sincronizzazione

## L'allocazione delle risorse – 1

- **Problema chiave e ricorrente in ogni modello di programmazione concorrente**
  - Coinvolge tutte le dimensioni identificate da Toby 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.

Corso di Laurea Magistrale in Informatica, Università di Padova
4/19



Criteri di sincronizzazione

## L'allocazione delle risorse – 2

- **L'ampiezza della richiesta di un cliente appare come parametro del messaggio in ingresso sul canale tipato**
- **Per leggere il parametro (e determinare la soddisfacibilità della richiesta) occorre però prima accettare la sincronizzazione**
  - Che fare se la richiesta non fosse soddisfacibile al momento?
  - Di sicuro non vogliamo ricorrere all'uso del "*busy wait*" !
  - Questione di usabilità prima che di potere espressivo

Corso di Laurea Magistrale in Informatica, Università di Padova
5/19



Criteri di sincronizzazione

## L'allocazione delle risorse – 3

- **L'uso delle guardie di Dijkstra permette *avoidance synchronization***
  - Evitare sincronizzazione ove questa non sia ammissibile nello stato corrente della risorsa
- **L'uso delle variabili di condizione del *monitor* di Hoare consente di ottenere sincronizzazione condizionale (*condition synchronization*)**
  - Ciò che ci servirebbe in questo caso: prima valutare il parametro e poi, se necessario, imporre attesa
  - Ma il limite strutturale del *monitor* è proprio richiedere programmazione esplicita dell'attesa!

Corso di Laurea Magistrale in Informatica, Università di Padova
6/19

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 un'altra coda d'attesa
    - Dove il cliente attenderà il verificarsi di condizioni più propizie
    - Consentendo così al canale di poter accogliere nuove richieste

Corso di Laurea Magistrale in Informatica, Università di Padova
7/19

Criteri di sincronizzazione

L'allocazione delle risorse – 5

Forma base (1 risorsa per richiesta)
Soluzione SR per il nostro problema

```

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;
                    
```

```

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' di ingresso

Corso di Laurea Magistrale in Informatica, Università di Padova
8/19

Criteri di sincronizzazione

L'allocazione delle risorse – 6

- Il difetto principale dell'approccio SR è il suo eccessivo costo realizzativo
  - Legare direttamente la sincronizzazione (e quindi l'accodamento di chiamata) al valore di parametri della richiesta rispetto allo stato della risorsa comporta che tutte le richieste accodate debbano essere rivalutate ogni volta che la valutazione possa essere cambiata!
- L'alternativa è trasferire la richiesta attualmente non soddisfacibile a un'altra coda
  - Senza doverne rivalutare l'eventuale guardia
  - Con una singola operazione atomica (→ senza *race condition*)

Corso di Laurea Magistrale in Informatica, Università di Padova
9/19

Criteri di sincronizzazione

L'allocazione delle risorse – 7

- Vediamo come il trasferimento di cosa ci possa dare il potere espressivo richiesto ...
  - Esercizio 9  
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)

Corso di Laurea Magistrale in Informatica, Università di Padova
10/19

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

Corso di Laurea Magistrale in Informatica, Università di Padova
11/19

Criteri di sincronizzazione

Semantica del trasferimento di coda – 2

- Permettere trasferimento di coda a livello di linguaggio pone due domande delicate
  - Verso quale coda di canale (*entry*) permetterlo
  - Come trattare il *time-out* eventualmente posto dal cliente sulla sua invocazione iniziale

Corso di Laurea Magistrale in Informatica, Università di Padova
12/19

Criteri di sincronizzazione

**Semantica del trasferimento di coda – 3**

- Verso quale coda di canale permetterlo
  - 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 il rilascio dell'entità di partenza
    - Il che potrebbe non essere desiderabile
  - Il canale destinazione deve essere compatibile con quello di partenza
    - Quindi o nessun parametro o esattamente gli stessi della chiamata iniziale
  - La direzione del trasferimento (interna o esterna) determina il destinatario finale del "lucchetto" originale

Corso di Laurea Magistrale in Informatica, Università di Padova

13/19

Criteri di sincronizzazione

**Semantica del trasferimento di coda – 4**

- 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

Corso di Laurea Magistrale in Informatica, Università di Padova

14/19

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 seconds
    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

Corso di Laurea Magistrale in Informatica, Università di Padova

15/19

Criteri di sincronizzazione

**Esercizio 10**

**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.

Corso di Laurea Magistrale in Informatica, Università di Padova

16/19

Criteri di sincronizzazione

**Un esempio complesso – 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

Corso di Laurea Magistrale in Informatica, Università di Padova

17/19

Criteri di sincronizzazione

**Un esempio complesso – 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

Corso di Laurea Magistrale in Informatica, Università di Padova

18/19



Criteria di sincronizzazione

## Un esempio complesso – 3

- ❑ Vediamo come programmare questa soluzione ...
- ❑ **Esercizio 11**  
Realizzare la stessa semantica della soluzione mostrata usando un altro linguaggio a piacere

**PENINSULAR RAIL ROAD LINE.**  
**CHANGE OF HOURS.**  
On and after Monday, January 16, 1951.

STATION	ARRIVAL	DEPARTURE
San Francisco	7:00 AM	7:30 AM
San Jose	8:00 AM	8:30 AM
San Diego	9:00 AM	9:30 AM
Los Angeles	10:00 AM	10:30 AM
San Francisco	11:00 AM	11:30 AM
San Jose	12:00 PM	12:30 PM
San Diego	1:00 PM	1:30 PM
Los Angeles	2:00 PM	2:30 PM
San Francisco	3:00 PM	3:30 PM
San Jose	4:00 PM	4:30 PM
San Diego	5:00 PM	5:30 PM
Los Angeles	6:00 PM	6:30 PM