



Un modello di *rendezvous*

SCD

Anno accademico 2009/10
Sistemi Concorrenti e Distribuiti

Tullio Vardanega, tullio.vardanega@math.unipd.it

Corso di Laurea Magistrale in Informatica, Università di Padova

1 / 18



Un modello di *rendezvous*

Modello base – 1

- Interazione di tipo cliente-servente
 - Il servente dichiara i servizi che è disposto a fornire ai clienti
 - La specifica del servente dichiara i canali d'accesso (*entry*) che corrispondono a ciascun servizio
 - Ciascun canale specifica il suo proprio protocollo di scambio parametri
 - Il cliente emette una richiesta (*entry call*) nominando servente e canale
 - Il servente fornisce uno dei servizi richiesti esprimendone esplicitamente la propria accettazione
 - La comunicazione tra servente e cliente è sincrona e non richiede necessariamente il passaggio di dati

Corso di Laurea Magistrale in Informatica, Università di Padova

2 / 18



Un modello di *rendezvous*

Modello base – 2

```
task type Operator is
  entry Query (A_Person : in Name;
               An_Address : in Address;
               A_Number : out Number);
end Operator;
Ann : Operator;

task type User;
task body User is
  My_Number : Number;
begin
  Ann.Query(
    "u", "u",
    My_Number);
end User;
```

```
task body Operator is
begin
  loop
    accept Query(A_Person : in Name;
                 An_Address : in Address;
                 A_Number : out Number) do
    ...
    end loop;
  end Operator;
```

Specifica di punto di accesso e protocollo

Invocazione

Realizzazione di accettazione

Corso di Laurea Magistrale in Informatica, Università di Padova

3 / 18



Un modello di *rendezvous*

Modello base – 3

- Storicamente chiamato *rendezvous*
 - Per rappresentare il fatto che cliente e servente si incontrano su uno specifico canale nello stesso istante temporale
- Al momento dell'incontro i parametri di modo *in* passano dal cliente al servente
- Il servente esegue il servizio richiesto come una normale procedura e poi restituisce i parametri di modo *out* al cliente
- A quel punto la sincronizzazione si interrompe e i processi riprendono la loro esecuzione concorrente

Corso di Laurea Magistrale in Informatica, Università di Padova

4 / 18



Un modello di *rendezvous*

Modello base – 4

- Nella forma base
 - Il servente si sospende in attesa di una richiesta
 - Come previsto per l'entità *Server* nel modello di concorrenza di lezione C01
 - Il cliente si sospende fino alla disponibilità del servizio
 - La chiamata del cliente viene posta in una coda associata al canale (*entry queue*)
 - L'ordine di accodamento è normalmente FIFO ma può essere configurato diversamente
 - Per esempio su base prioritaria
 - Ma con le conseguenze che questo può comportare in termini di *starvation*

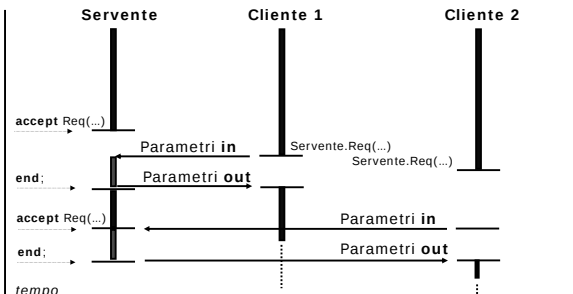
Corso di Laurea Magistrale in Informatica, Università di Padova

5 / 18



Un modello di *rendezvous*

Modello base – 5



Corso di Laurea Magistrale in Informatica, Università di Padova

6 / 18



Un modello di *rendezvous*

Esempio – 1

Il crivello di Eratostene rivisitato

- Una diversa realizzazione dell' algoritmo visto nella lezione precedente
- Ogni coppia di processi nella sequenza comunica tramite *rendezvous*
- L'effetto di sincronizzazione rende superflua la mutua esclusione
 - L'accodamento FIFO sulla coda d'accesso preserva l'ordine dei valori da esaminare

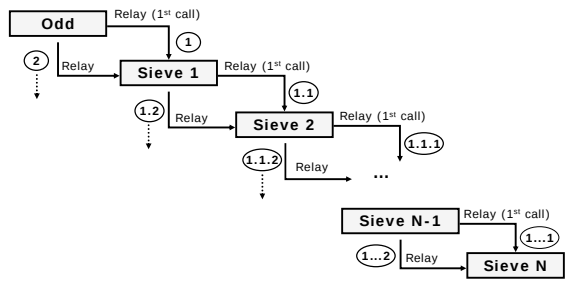
Corso di Laurea Magistrale in Informatica, Università di Padova

7 / 18



Un modello di *rendezvous*

Esempio – 2



Corso di Laurea Magistrale in Informatica, Università di Padova

8 / 18



Un modello di *rendezvous*

Sincronizzazione tripartita – 1

Il modello *rendezvous* è sincrono, asimmetrico e bidirezionale

Le azioni svolte dal servente durante la sincronizzazione possono coinvolgere processi terzi

- Ciò permette di realizzare forme complesse di sincronizzazione sempre preservando separazione funzionale tra i processi coinvolti
 - Proprietà desiderabile

Corso di Laurea Magistrale in Informatica, Università di Padova

9 / 18



Un modello di *rendezvous*

Sincronizzazione tripartita – 2

Il coinvolgimento di processi terzi può avvenire in 2 forme strutturalmente distinte ma duali tra loro

- Annidando accettazioni
 - Modellando una **macchina a stati** in cui alcuni stati sono raggiungibili solo a partire da un dato stato iniziale
- Invocando accettazioni nella realizzazione di una **accettazione**
 - Realizzando la fornitura di un **servizio composto** che racchiude il possibile contributo di più serventi che si siano ripartiti tra loro il lavoro

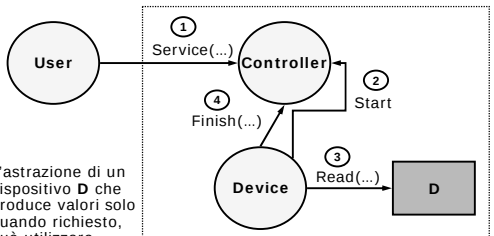
Corso di Laurea Magistrale in Informatica, Università di Padova

10 / 18



Un modello di *rendezvous*

Sincronizzazione tripartita – 3



L'astrazione di un dispositivo **D** che produce valori solo quando richiesto, può utilizzare sincronizzazione tripartita nella forma di macchina a stati imperniata sul **Controller**

Corso di Laurea Magistrale in Informatica, Università di Padova

11 / 18



Un modello di *rendezvous*

Sincronizzazione tripartita – 4

```
task User;
task Device;
task Controller is
entry Service (I : out Integer);
entry Start;
entry Finish (K : Integer);
end Controller;
```

```
task body User is
... Controller.Service(Val);
...
end User;
```

```
task body Controller is
begin
loop
accept Service (I : out Integer) do
accept Start;
accept Finish (K : Integer) do
I := K;
end Finish;
end Service;
end loop;
end Controller;
```

```
task body Device is
Val : Integer;
procedure Read
(I : out Integer) is ... ;
begin
loop
Controller.Start;
Read(Val);
Controller.Finish(Val);
end Device;
```

Corso di Laurea Magistrale in Informatica, Università di Padova

12 / 18

Un modello di *rendezvous*

Sincronizzazione tripartita – 5

Una fornitura di servizio che nasconde al cliente l'eventuale necessità di approvvigionamento presso componenti incapsulati in una articolazione complessa di "sistema servente"

Strutturazione gerarchica con *encapsulation*

Corso di Laurea Magistrale in Informatica, Università di Padova 13/18

Un modello di *rendezvous*

Sincronizzazione tripartita – 6

```

task Warehouse is
  entry Enquiry
    (Item : Part_Number;
     In_Stock : out Boolean);
end Warehouse;

task Customer_Service is
  entry Request_Part
    (Order : Part_Number;
     Part : Spare_Part;
     Order : Order_Number);
end Customer_Service;

task body Customer_Service is
  In_Stock : Boolean;
  begin
  loop
    accept Request_Part
      (Order : Part_Number;
       Part : Spare_Part;
       Order : Order_Number) do
      if In_Stock then
        Part := The_Part; Order := None;
      else
        Warehouse.Enquiry(Order, In_Stock);
        if In_Stock then
          -- go get part from Warehouse
          Part := The_Part; Order := Next_Order_Nr;
        end if;
      end if;
    end Request_Part;
  end loop;
end Customer_Service;
  
```

Corso di Laurea Magistrale in Informatica, Università di Padova 14/18

Un modello di *rendezvous*

Punti d'accesso privati – 1

- ❑ Un servente non deve necessariamente esporre al pubblico tutti i suoi canali
- ❑ Alcuni possono essere ristretti per motivi di incapsulazione e/o di astrazione
- ❑ La dichiarazione dei canali deve in tal caso distinguere tra pubblici e privati

Corso di Laurea Magistrale in Informatica, Università di Padova 15/18

Un modello di *rendezvous*

Punti d'accesso privati – 2

```

task User;
task Controller is
  entry Service (I : out Integer);
private
  entry Start;
  entry Finish (K : Integer);
end Controller;

task body Controller is
  task Device;
  task body Device is
    Val : Integer;
    procedure Read (I : out Integer) is ...;
  begin
  loop
    Controller.Start;
    Read(Val);
    Controller.Finish(Val);
  end Device;
  -- continues in sidebar
  begin -- Controller
  loop
    accept Service (I : out Integer) do
      accept Start;
      accept Finish (K : Integer) do
        I := K;
      end Completed;
    end Service;
  end loop;
end Controller;
  
```

La visibilità ai canali privati è ristretta al solo ambito (scope) del processo Controller

Corso di Laurea Magistrale in Informatica, Università di Padova 16/18

Un modello di *rendezvous*

Casi d'errore

- ❑ Una eccezione sollevata durante la sincronizzazione ne causa l'abbandono e si propaga a entrambi i partecipanti
- ❑ Emettere una richiesta d'accesso verso un processo terminato è un errore a tempo di esecuzione e solleva una eccezione nel chiamante

Corso di Laurea Magistrale in Informatica, Università di Padova 17/18

Un modello di *rendezvous*

Stati d'esecuzione di processo

Corso di Laurea Magistrale in Informatica, Università di Padova 18/18