Progettazione di un Sistema Concorrente e Distribuito

Moreno Ambrosin

Università degli studi di Padova

Dipartimento di Matematica

Corso di laurea Magistrale in Informatica

Agosto 2013

Indice

Introduzione

- La progettazione di un Sistema Concorrente e Distribuito si compone di:
  - Analisi del problema.
    - Definizione della specifica.
    - Analisi degli aspetti legati alla Distribuzione.
    - Analisi degli aspetti legati alla Concorrenza.
  - Costruzione di una soluzione.
    - Definizione di una architettura di distribuzione.
    - Risoluzione delle problematiche di concorrenza.
  - Scelta del supporto tecnologico da utilizzare per l'implementazione.

Analisi del Problema

- Individuazione e prima definizione delle entità del Sistema.
  - ad es. nel progetto di un Sistema ferroviario:
    - Treno (entità attiva)
    - Viaggiatore (entità attiva)
    - Segmento (entità reattiva)
    - Stazione (entità passiva)
      - Piattaforma (entità reattiva)
      - Biglietteria (entità reattiva)
      - Pannello Informativo (entità reattiva)
- Identificazione e definizione dei requisiti di massima del Sistema.
  - Operazione sottovalutata ma importante.

Analisi del Problema - Distribuzione (1)

- Primo aspetto da valutare
  - Fornisce una visione di alto livello dell'architettura di Sistema.
  - Alcune scelte vincolano le modalità di interazione tra le entità.
- Il Sistema dovrà apparire agli utenti come unitario e coerente.
- Caratteristiche Desiderabili
  - **Trasparenza:** Il Sistema dovrà il più possibile rendere trasparenti all'utente le caratteristiche legate alla distribuzione (Accesso, Collocazione, Migrazione, Spostamento, Replicazione, Malfunzionamento, Persistenza)

Analisi del Problema - Distribuzione (2)

- **Openess:**
  - Il Sistema dovrà garantire portabilità e interoperabilità.
  - Il Sistema dovrà essere fruibile mediante regole standard (interfacce).
  - Organizzazione del Sistema in componenti di dimensione ridotta, e facilmente sostituibili.
  - Separazione tra *politiche* e *meccanismi*.
- **Scalabilità:**
  - Rispetto alla cardinalità del Sistema (ad es. nel progetto di un Sistema ferroviario, è desiderabile poter aumentare la popolazione di Stazioni e Segmenti di collegamento).
  - Rispetto alla distribuzione spaziale delle componenti.
  - Rispetto alle problematiche locali di gestione (che non devono affliggere l'intero Sistema).

Analisi del Problema - Distribuzione (3)

- **Fault Tollerance:**
  - Il Sistema deve essere progettato in modo tale da ridurre l'impatto causato da *partial failures*.
  - Il Sistema dovrà gestire errori di comunicazione tra i nodi.
- **Avvio ordinato:** Il Sistema dovrà essere avviato in modo tale da permettere a tutte le componenti di comunicare senza errori.
- **Terminazione in stato Consistente** Il Sistema deve poter essere terminato in uno stato consistente; nessun entità dovrà rimanere attiva dopo la procedura di terminazione.

Analisi del Problema - Distribuzione (4)

- Prima modellazione ad alto livello delle componenti distribuite del Sistema.
- Scelta di alcuni aspetti legati alla distribuzione
  - dove adottare distribuzione *verticale* o *orizzontale*;
  - modalità di comunicazione tra le componenti (*sincrona* o *asincrona*);
  - definizione di possibili interfacce.
- Valutazione delle implicazioni nell'adozione di gradi di distribuzione diversi sul Sistema.
- Individuazione delle problematiche specifiche del problema.
  - Ad es. effetti sul realismo della simulazione.

Analisi del Problema - Distribuzione - Esempio (1)

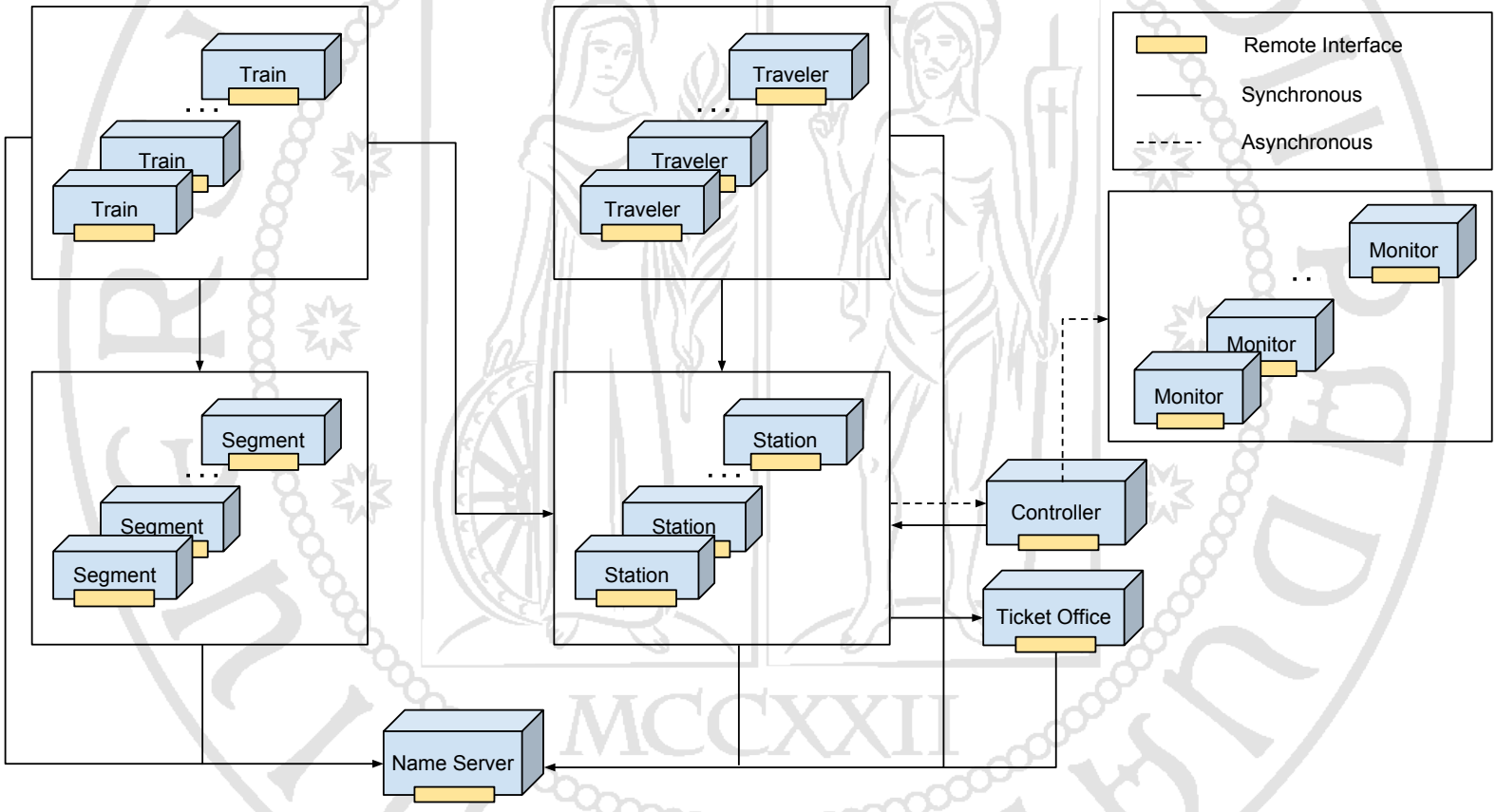

Figura: Architettura di alto livello in cui tutte le entità principali vengono distribuite.

Analisi del Problema - Distribuzione - Esempio (2)

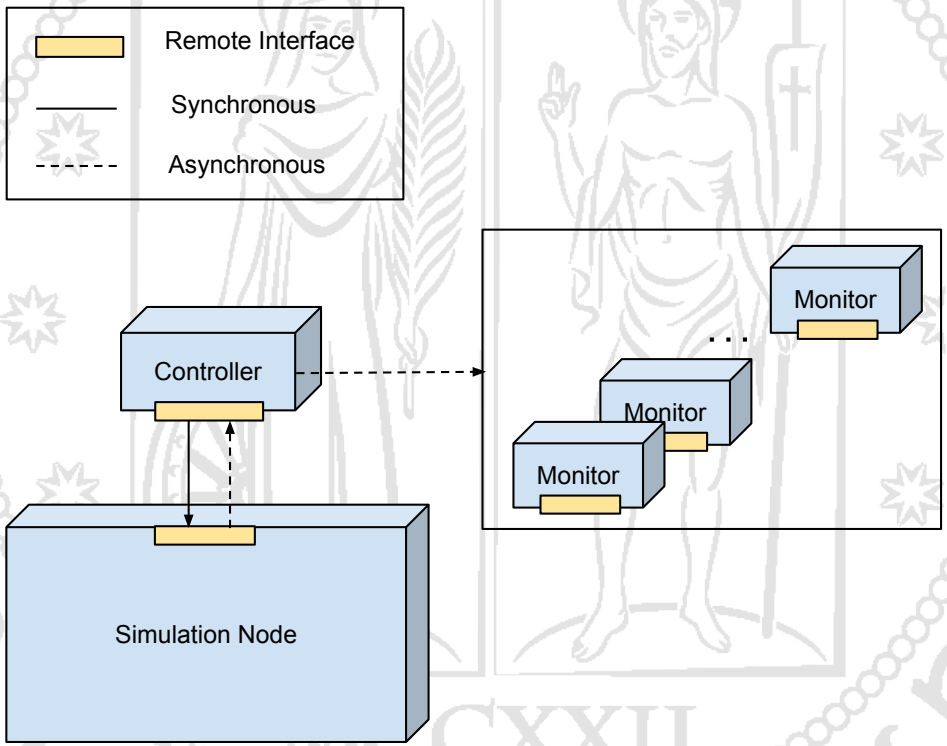

Figura: Architettura di alto livello in cui solo Controller Centrale e Visualizzazione sono distribuite.

Analisi del Problema - Distribuzione - Esempio (3)

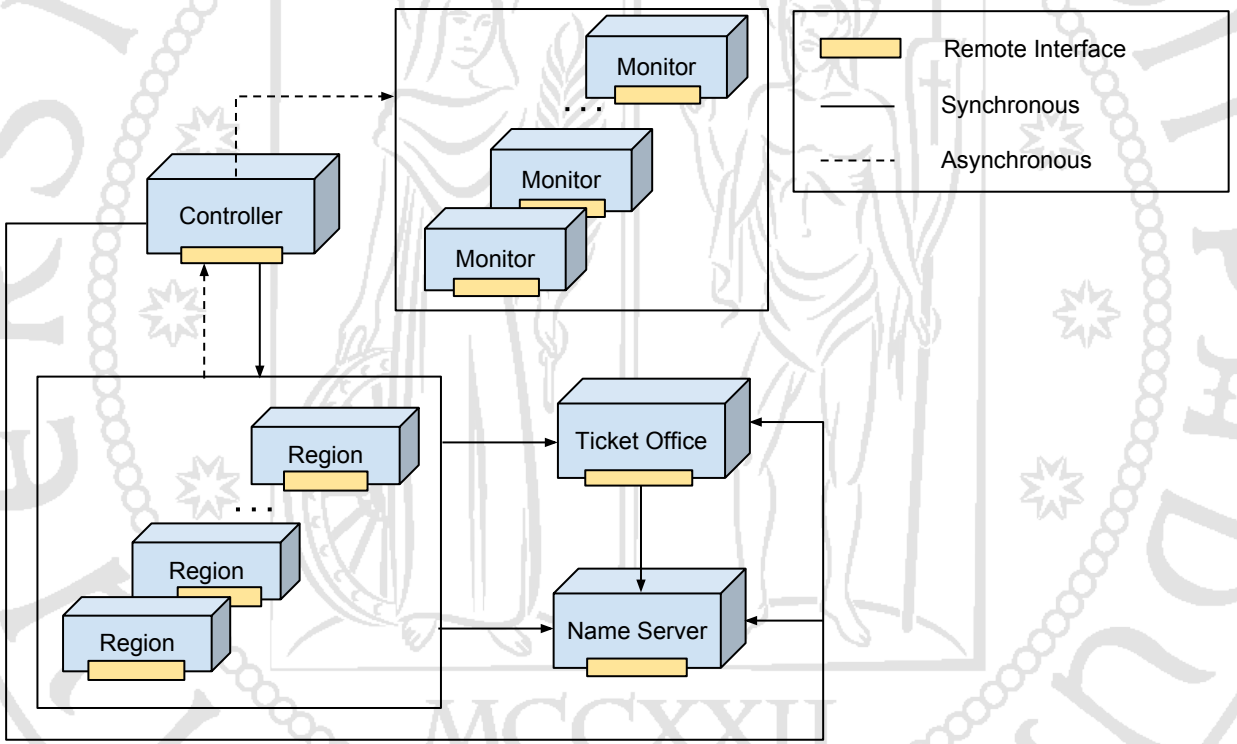

Figura: Architettura di alto livello con distribuzione a livello di *Regioni*.

Analisi del Problema - Concorrenza

- Prima definizione dei protocolli logici di interazione concorrente tra le entità.
  - Il più possibile indipendente dalla scelta di un modello di concorrenza specifico.
  - Identificazione dei punti critici in cui il problema è concorrente.
- Definizione delle caratteristiche specifiche per ciascuna entità.
  - ad es.: Segmento come entità reattiva con agente di controllo, a molteplicità $M \geq 1$.
  - ad es.: Stazione come una entità passiva, che mantiene al suo interno
    - un numero $M \geq 1$ di Piattaforme (entità reattive con agente di controllo, a molteplicità 1), ...

Analisi del Problema - Concorrenza - Esempio (1)

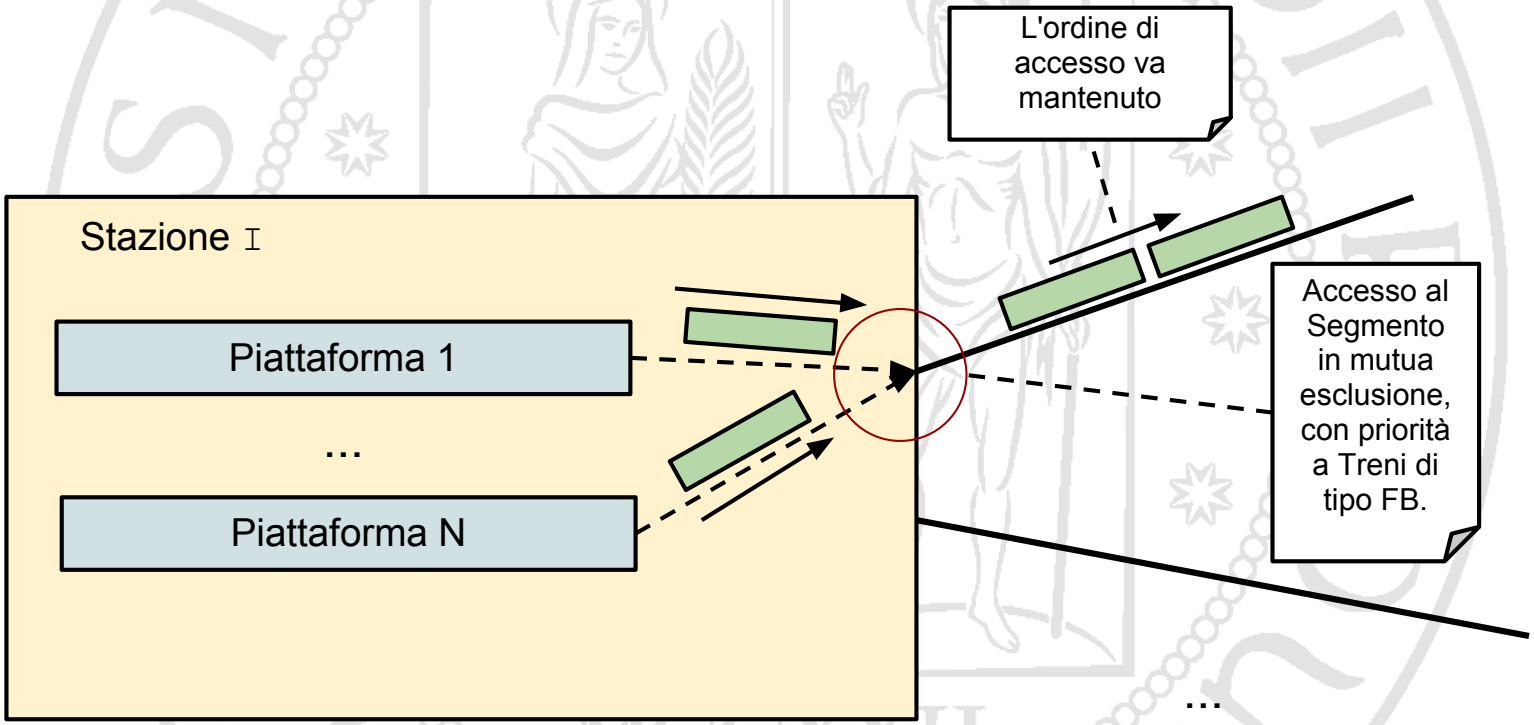

Figura: Accesso ad un segmento.

Analisi del Problema - Concorrenza - Esempio (2)

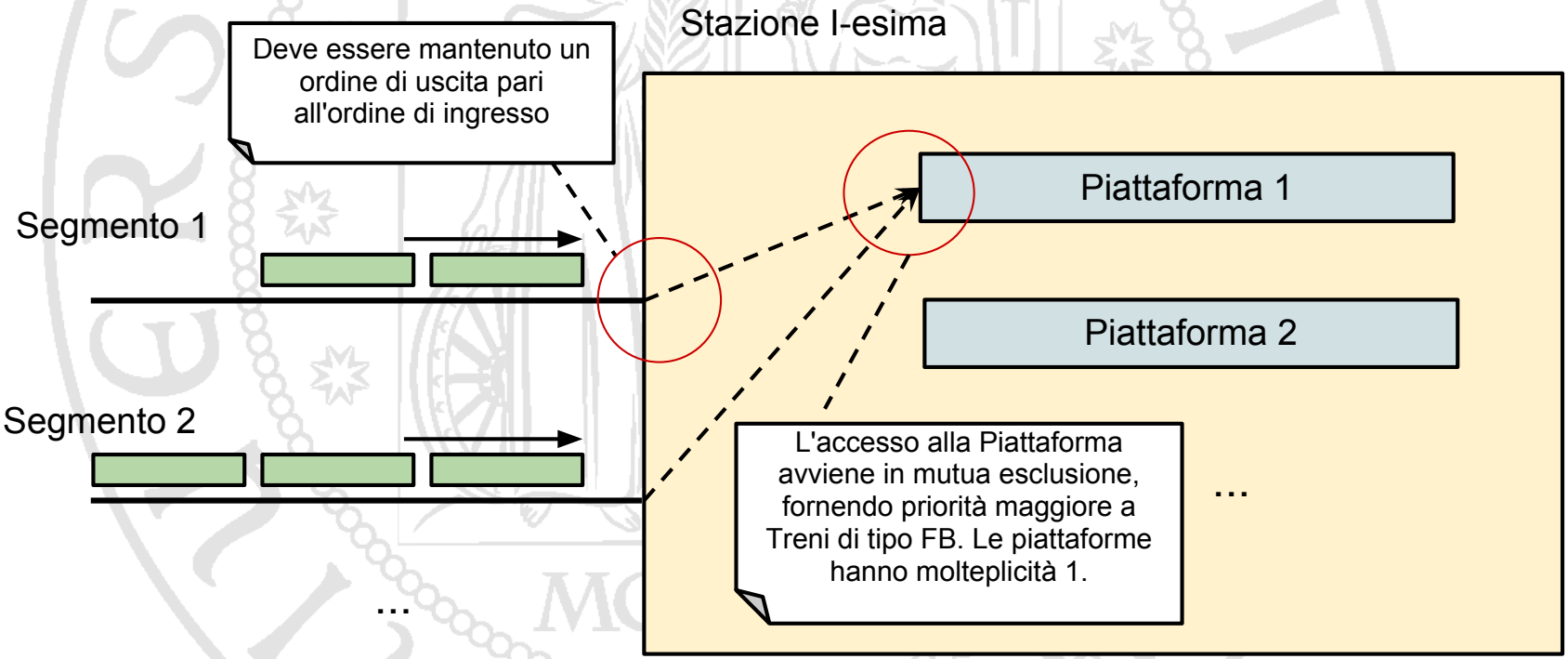

Figura: Uscita da un segmento e accesso alla Piattaforma successiva.

Costruzione di una Soluzione - Distribuzione

- Scelta di un'architettura di distribuzione.
- Possibile introduzione di nuove entità
  - ad es.: utilizzo di Regioni.
- Definizione architetturale a grana più fine
  - ad es.: introduzione di una gerarchia di Biglietterie per distribuire conoscenza e oneri di calcolo.
    - Centrale
    - Regionale
    - Interna alle Stazioni
- Scelta del modello di comunicazione tra le componenti.
- Identificazione delle problematiche conseguenti alle scelte architetturali.

Costruzione di una Soluzione - Distribuzione - Esempio

- Come realizzare il passaggio di entità Treno e Viaggiatore tra Regioni?
  - *possibile soluzione:* Utilizzare Stazioni speciali (di "gateway") per permettere l'uscita di un Treno da una Regione; trasferimento diretto di un Viaggiatore.
- Come si traduce il trasferimento remoto di una entità?
  - creazione/distruzione?
  - replicazione?
  - Vincolo sulla realizzazione dell'entità per facilitare il trasferimento remoto!
  - *possibile soluzione:* Disaccoppiamento tra entità e thread che ne esegue le operazioni: utilizzo di un thread pool e di descrittori di entità.

Costruzione di una Soluzione - Distribuzione - Avvio e Terminazione (1)

Avvio

- Deve essere coordinato tra le entità.
  - Devono essere evitati tentativi di comunicazione tra componenti non ancora pronte o allocate.
- È opportuno scegliere un ordine di avvio tra le componenti e separare la fase di inizializzazione della componente e l'avvio della simulazione.

Terminazione

- Ha come prerequisito la definizione dei limiti entro i quali uno Stato del Sistema è consistente. Ad es. nel progetto di un Sistema ferroviario:
  - è accettabile che il Sistema termini con un certo numero di Treni in attesa di accedere ad una Piattaforma;

Costruzione di una Soluzione - Distribuzione - Avvio e Terminazione (2)

- non è accettato lo stato di terminazione per il quale un Viaggiatore è in attesa di un Biglietto.

- È conveniente adattare algoritmi distribuiti noti (ad es. *distributed snapshot*).
- Nessun thread in esecuzione sui nodi di calcolo dopo la procedura.

Costruzione di una Soluzione - Concorrenza

- Scelta di un modello di concorrenza adatto alle caratteristiche del problema.
- Valutazione di modelli differenti
  - ad es. modello ad *Attori* o a *monitor*.
- Modellazione delle entità di simulazione e della loro interazione con strumenti di modello.
  - Scomposizione delle interazioni in sottoproblemi semplici.
- Evitare scelte di progettazione che utilizzano operazioni specifiche offerte dalle tecnologie
  - Nessuna assunzione a priori sul linguaggio che verrà utilizzato.
  - Nessuna assunzione sulle politiche di scheduling adottate dalla macchina sottostante.
- Attenzione a *deadlock* e *starvation*.

Costruzione di una Soluzione - Concorrenza - Esempio (1)

- Accesso ad un Segmento S da parte di un Treno T, dall'estremo D_T, $D_T =$ `First_End`.
  - Rischio starvation dei Treni in attesa.
- Segmento realizzato come risorsa protetta con agente di controllo *monitor*.
- T accede solo se:
  - S è libero
  - S non è libero ma i Treni in transito hanno avuto accesso da D_T e il numero di accessi massimo non è stato raggiunto.
  - S non è libero ma i Treni in transito hanno avuto accesso da D_T, il numero di accessi massimo è stato raggiunto, ma all'estremo opposto non vi sono Treni in attesa.
- In tutti gli altri casi T deve attendere presso l'estremo di accesso.
- All'uscita da S l'ultimo Treno risveglia i Treni in attesa presso l'estremo opposto.

Costruzione di una Soluzione - Concorrenza - Esempio (2)

```
procedure Access_Segment_Monitor(T:Train,Access_End:Integer) begin
    ...
    if Access_End = First_End then
        while
            ((not Free) and (Access_End /= Current_Direction))
            or
            ((not Free) and (Access_End = Current_Direction) and
                (Access_Number = MAX) and (Second_End_Count > 0))
        loop
            First_End_Count := First_End_Count + 1;
            wait(Can_Enter_First_End);
            First_End_Count := First_End_Count - 1;
        end loop;
    else
        ... // Simmetrico per accesso dalla direzione opposta
    end if;
    if (Free = True) then
        Free := False;
        if (Access_End /= Current_Direction) then
            Access_Number := 1;
            Current_Direction := Access_End;
        end if;
    else
        if (Access_Number < MAX) then
            Access_Number := Access_Number + 1;
        end if;
    end if;
    ...
end;
```

- Procedura che regola l'accesso ad un Segmento da parte di un Treno;
- Accesso multiplo al Segmento, con numero massimo *MAX* di ingressi consecutivi per estremo.
- `Current_Direction` mantiene la direzione dei Treni in transito.

Costruzione di una Soluzione - Concorrenza - Esempio (3)

Valutazione dei casi possibili per dimostrare la correttezza della soluzione presentata, una volta che T esegue all'interno della procedure di risorsa protetta `Access_Segment_Monitor`.

Caso 1: Accesso Consentito

Precondizione: `Free=True`

T imposta il valore di `Free` a `False`. Se l'estremo di accesso è diverso da quello corrente, allora il numero di accessi per estremo `Access_Number` viene incrementato di 1, e la direzione corrente `Current_Direction` è settata a 1. In questo modo una volta raggiunto il massimo numero di accessi MAX, esso viene re-impostato ad 1 solo se T proviene da una direzione diversa dall'ultima percorsa. Esegue infine le operazioni previste dopo aver ottenuto l'accesso.

Costruzione di una Soluzione - Concorrenza - Esempio (4)

Caso 2: Accesso Consentito

Precondizione: `Free=False` and `Current_Direction`$= D_T$ and $1 <$`Access_Number`$<$`MAX`

Perché sia verificata la Precondizione, almeno un altro Treno proveniente dallo stesso estremo deve aver avuto accesso al Segmento (Caso 1). In questo caso, T si limita a incrementare di 1 il contatore di accessi per estremo `Access_Number`, e ad eseguire le operazioni previste dopo l'accesso.

Caso 3: Accesso Consentito

Precondizione: `Free=False` and `Current_Direction`$= D_T$ and `Access_Number`$=$`MAX` and `Second_End_Count`$= 0$

Perché sia verificata la Precondizione, almeno MAX Treni provenienti dallo stesso estremo hanno avuto accesso al Segmento (Caso 1 + Caso 2). In questo caso, T non incrementare il contatore di accessi `Access_Number`, ed esegue le operazioni previste dopo l'accesso.

Costruzione di una Soluzione - Concorrenza - Esempio (5)

Caso 4: Accesso non consentito

Precondizione: `Free=False` and `Current_Direction`$\neq D_T$

Perché la pre-condizione sia verificata, almeno un Treno proveniente dall'estremo opposto rispetto a T ha eseguito nel Caso 1. Il thread corrente incrementa il contatore dei Treni in attesa per l'estremo corrente `First_End_Count`, e si pone in attesa su variabile di condizione `Can_Enter_First_End`.

Caso 5: Accesso non consentito

Precondizione: `Free=False` and `Current_Direction`$= D_T$ and `Access_Number=MAX` and `Second_End_Count`> 0

Perché la pre-condizione sia verificata, almeno un Treno proveniente dall'estremo opposto rispetto a T ha eseguito all'interno di `Access_Segment_Monitor` nel Caso 4 (relativamente alla propria direzione). Il thread corrente incrementa il contatore dei Treni in attesa per l'estremo corrente `First_End_Count`, e si pone in attesa su variabile di condizione `Can_Enter_First_End`.

Scelta degli strumenti tecnologici

- Utilizzo di linguaggi di programmazione e strumenti che meglio si adattano alle scelte di progetto (e non viceversa).
- Interessante l'utilizzo di tecnologie eterogenee
  - È difficile pensare ad un Sistema Distribuito realizzato con tecnologia uniforme.
  - Possibilità di utilizzare supporti tecnologici specifici per singola componente.
    Ad es. nella soluzione progettata:
    - ho utilizzato il linguaggio Ada per codificare le componenti che rappresentano le Regioni;
    - ho utilizzato il linguaggio Scala per la realizzazione di Name Server, Biglietteria e Controller Centrale;
    - ho utilizzato Javascript e HTML per la realizzazione dell'interfaccia grafica.

Conclusioni

- La progettazione di un Sistema Concorrente e Distribuito è un'operazione complessa.
- Molto importante l'analisi iniziale e il confronto fra diverse architetture si Sistema possibili.
- Errori comuni:
  - dare per scontato problematiche di distribuzione;
  - affidarsi a strumenti di linguaggio per risolvere problemi di concorrenza.