Railway Simulation

Moreno Ambrosin

Università degli studi di Padova

Facoltà di Scienze MM. FF. NN.

Corso di laurea in Informatica

Settembre 2013

Indice

Introduzione

- La progettazione di un sistema distribuito si compone di:
  - Analisi del problema.
    - Aspetti legati alla Distribuzione.
    - Aspetti legati alla Concorrenza.
  - Costruzione di una soluzione.
  - Scelta delle Tecnologie.

Analisi del Problema

- Identificare e definire i requisiti di massima del sistema.
  - Operazione sottovalutata ma importante.
- Individuazione e prima definizione delle entità del sistema.
  - ad es. nel progetto di un sistema ferroviario:
    - Treno (entità attiva)
    - Viaggiatore (antità attiva)
    - Segmento (entità reattiva)
    - Piattaforma (entità reattiva)
    - Biglietteria (entità reattiva)
    - Stazione (entità reattiva)

Analisi del Problema - Distribuzione (1)

- Primo aspetto da valutare
  - Fornisce un'architettura di alto livello del sistema.
  - Alcune scelte vincolano le modalità di interazione tra le entità.
- Caratteristiche Desiderabili
  - Il sistema dovrà apparire agli utenti in modo unitario e coerente.
    - ad es. nel progetto di un sistema ferroviario, la componente di visualizzazione dovrà nascondere l'architettura di distribuzione.
  - **Trasparenza:** Il sistema dovra 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 - Scelte di progetto

- Modellazione ad alto livello delle componenti del sistema necessarie e loro distribuzione.
- Scelta delle modalità di distribuzione
  - Dove adottare distribuzione *verticale* o *orizzontale*
- Scelta della modalità di comunicazione tra le componenti (*sincrona* o *asincrona*) e definizione di possibili interfacce.
- Valutazione delle implicazioni nell'adozione di gradi di distribuzione diversi sul sistema.
- Individuazione delle problematiche specifiche del problema, ad es.
  - Gestione del *Name Resolution* per le componenti.
  - Gestione della sincronizzazione tra clock fisici dei nodi che ospitano le varie componenti.

Analisi del Problema - Distribuzione - Esempio (1)

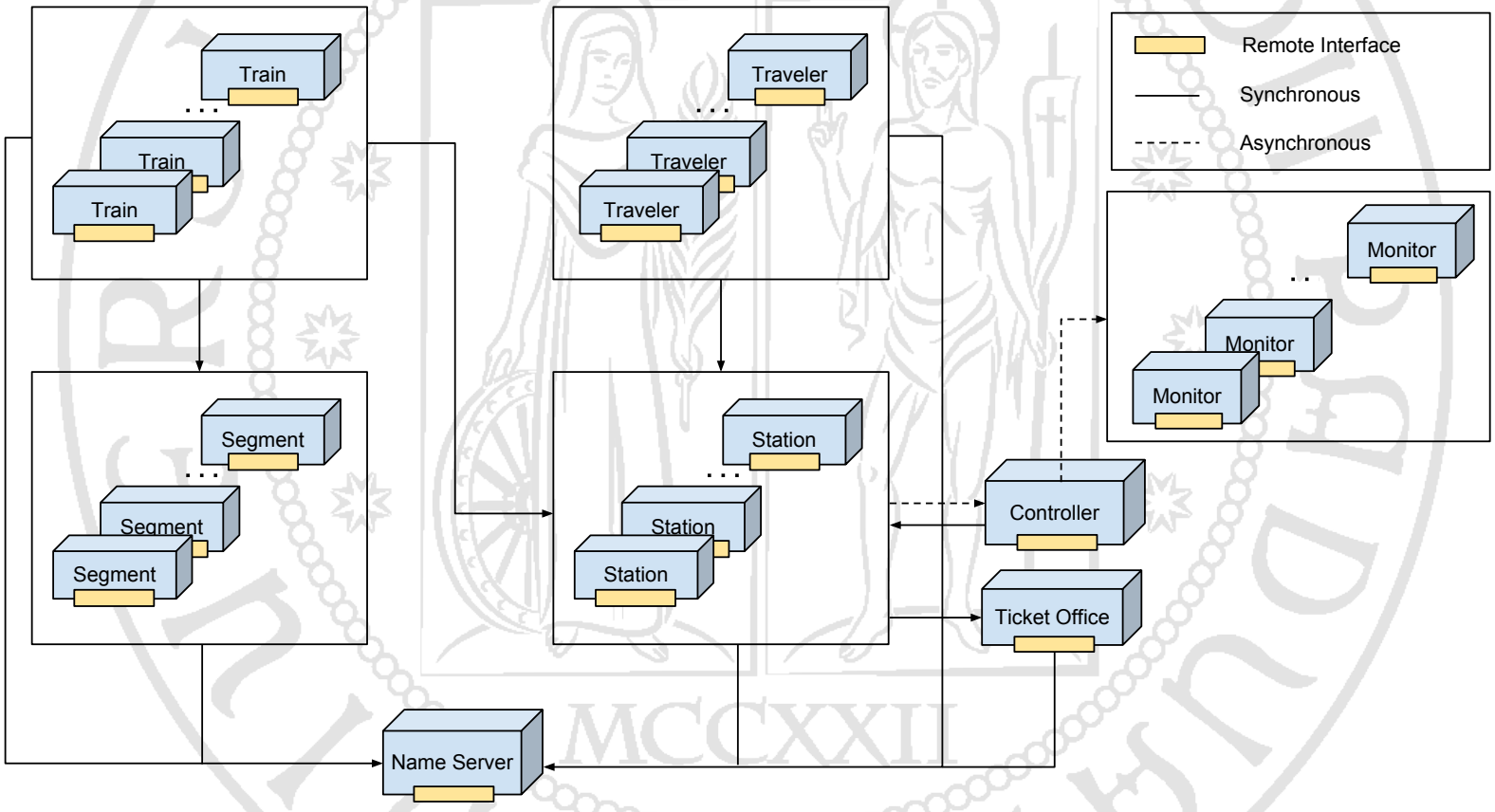

Figura: Architettura di distribuzione in cui tutte le entità principali vengono distribuite.

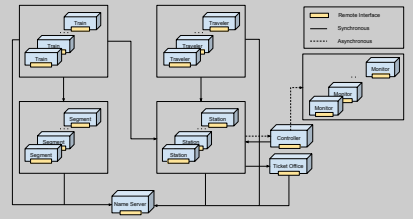

Figura: Architettura di distribuzione in cui tutte le entità principali vengono distribuite.

PRO:

- Scalabilità in dimensione
- Fault Tollerance a partial failures
- Semplice sostituibilità dei componenti

CONTRO

- Complessità terminazione
- Assenza di riferimento temporale unico
- Elevato traffico di rete (ok in LAN, bad on WAN)

Deve essere adottato un sistema di Naming che scali meglio di semplice tabella [Nome,Indirizzo]

Analisi del Problema - Distribuzione - Esempio (2)

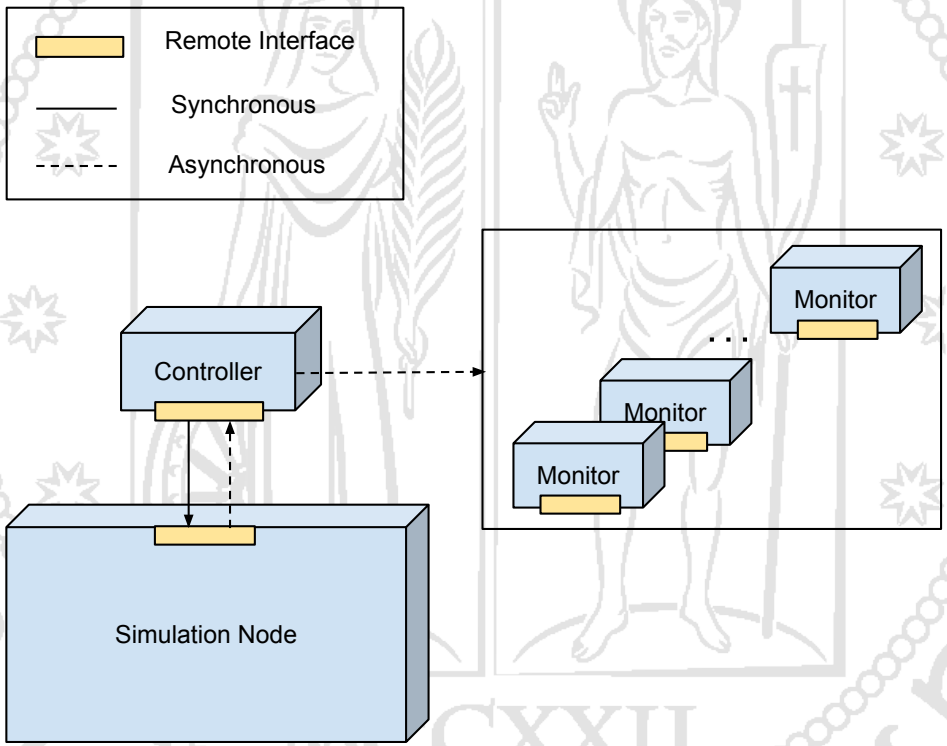

Figura: Architettura di distribuzione in cui solo Controller Centrale e Visualizzazione sono distribuite.

Analisi del Problema - Distribuzione - Esempio (3)

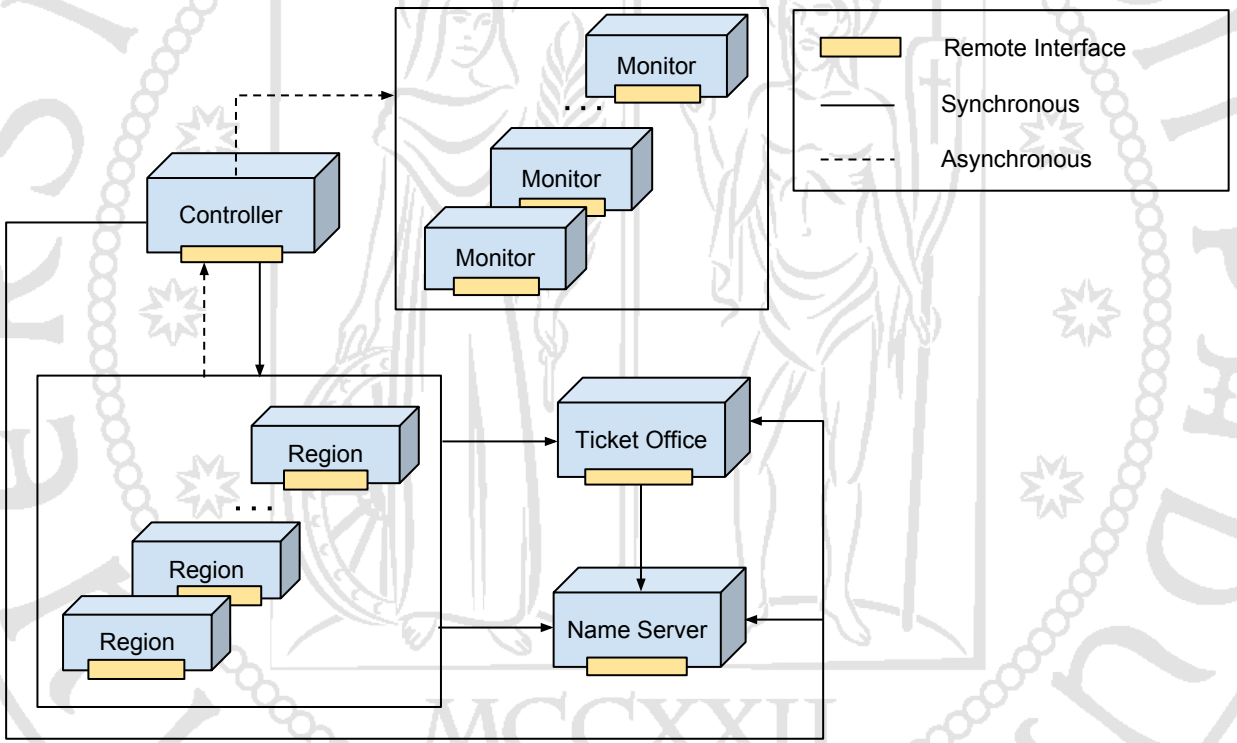

Figura: Architettura di distribuzione in cui sono aggiunte e distribuite *regioni*.

Analisi del Problema - Concorrenza

- Scelta delle caratteristiche specifiche per ciascuna entità.
  - ad es.: scelta di definire un Segmento come entità reattiva con agente di controllo, ad accesso mutuamente esclusivo con molteplicità $N \geq 1$.
  - ad es.: scelta di definire una Stazione come una struttura che mantiene al suo interno
    - un numero $M \geq 1$ di Piattaforme (entità reattive con agente di controllo ad accesso mutuamente esclusivo con molteplicità 1), ...
- Prima definizione dei protocolli logici di interazione tra entità che risiedono sullo stesso nodo di calcolo.
  - Il più possibile indipendente dalla scelta di un modello di concorrenza specifico.
  - Identificazione dei punti critici in cui il problema è concorrente.

Analisi del Problema - Concorrenza - Esempio (1)

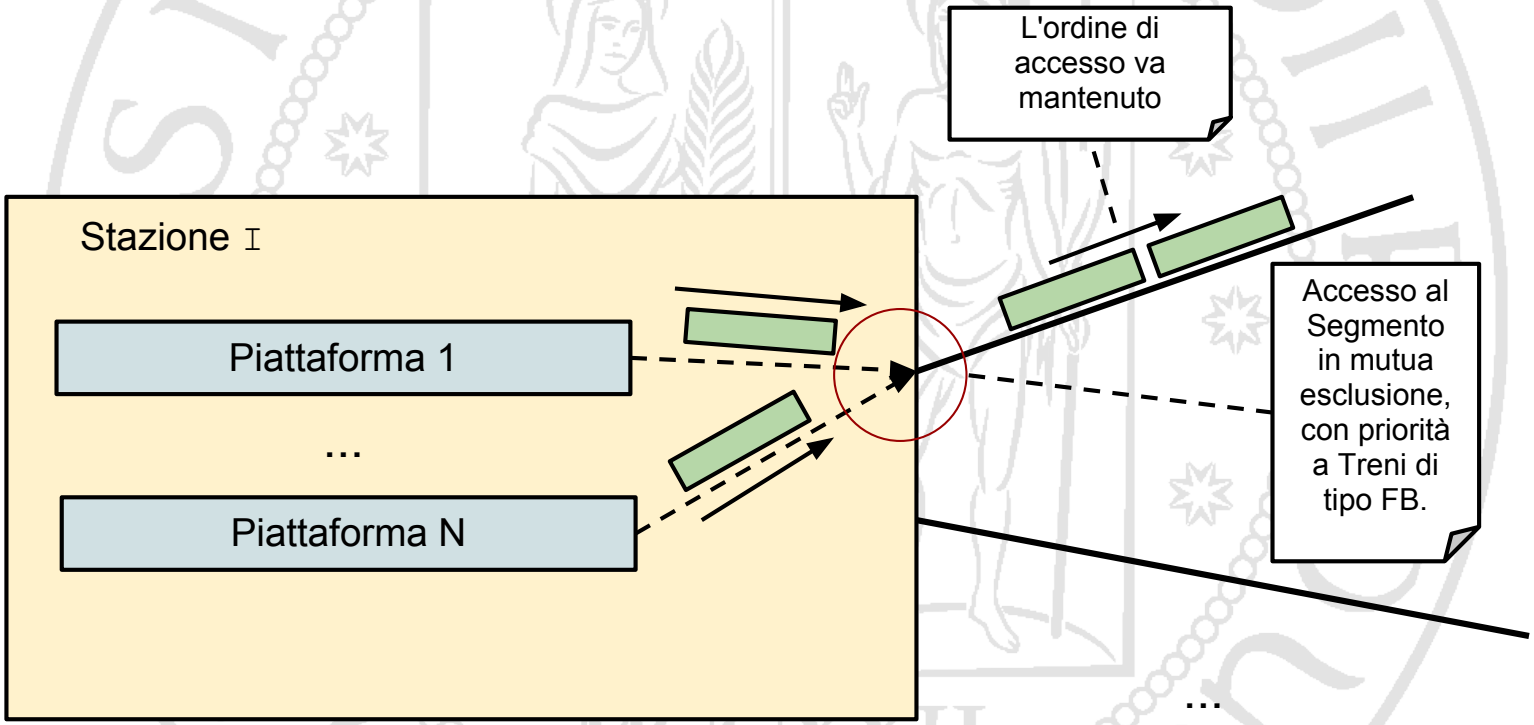

Figura: Ingresso in un segmento

Analisi del Problema - Concorrenza - Esempio (2)

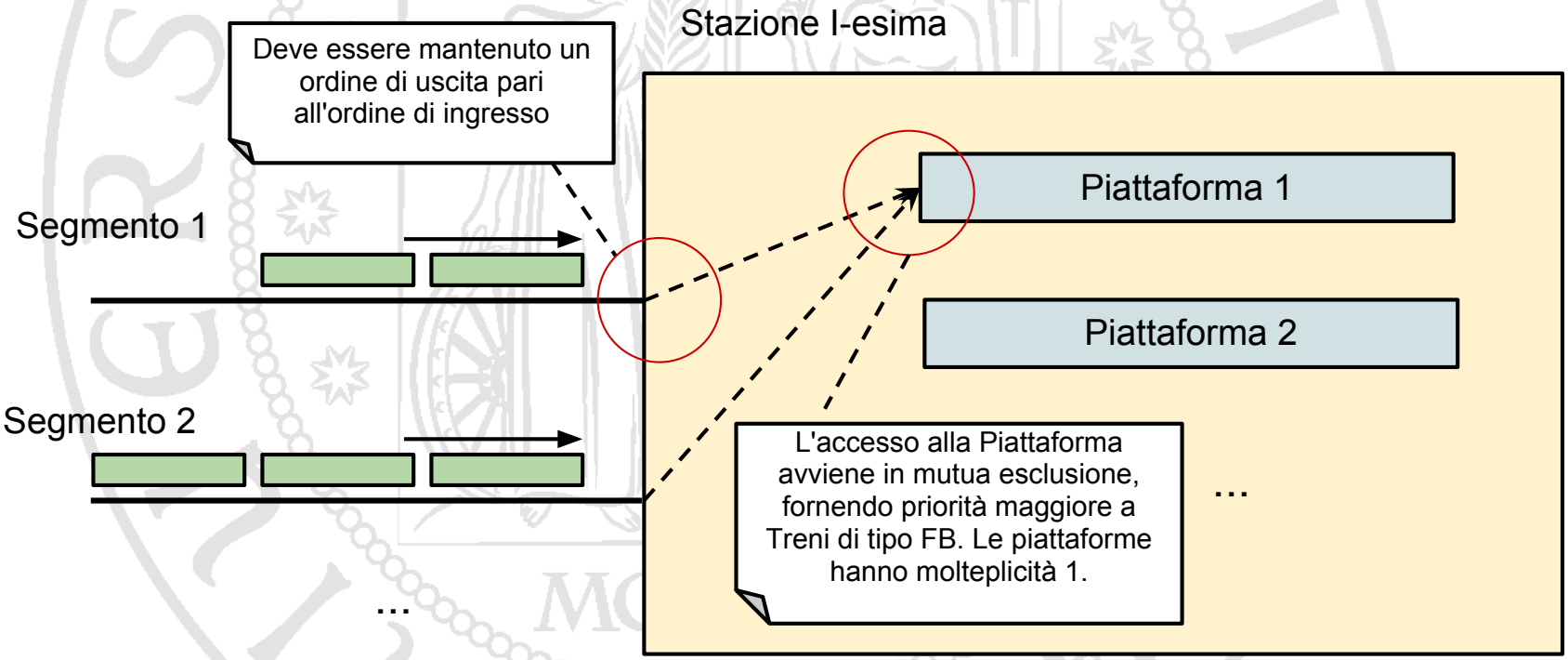

Figura: Ingresso in un segmento

Soluzione al Problema - Distribuzione

- Scelta di un'architettura di distribuzione.
  - Introduce nuove entità?
  - ad es.: utilizzare Regioni come collezione di Stazioni e Segmenti, che risiedono su un singolo nodo di calcolo, e sulle quali transitano Treni e Viaggiatori.
  - ad es.: introduzione di una gerarchia di Biglietterie per distribuire conoscenza e oneri di calcolo.
    - Centrale
    - Regionale
    - di Stazione
- Identificazione delle problematiche conseguenti a tale scelta.

Soluzione al Problema - 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!

Soluzione al Problema - Distribuzione - Avvio e Terminazione (1)

Avvio

- Deve essere coordinato tra le entità.
- 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 pre-requisito 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;
  - non è accettato lo stato di terminazione per il quale un Viaggiatore è in attesa di un Biglietto.

Soluzione al Problema - Distribuzione - Avvio e Terminazione (2)

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

Soluzione al Problema - 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.

Soluzione al Problema - Concorrenza - Esempio (1)

```
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 la direzione opposta
        ...
    end if;
    if (Free = True) then
        Free := False;
        Access_Number := 1;
        Current_Direction := Access_End;
    else
        if (Access_Number < MAX) then
            Access_Number := Access_Number + 1;
        end if;
    end if;
    ...
end;
```

Soluzione al Problema - Concorrenza - Esempio (2)

- Supponiamo che un thread esegua per un dato Treno T, e che T intenda accedere al Segmento S dall'estremo D_T, con $D_T =$`First_End`.
- 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:

Pre-condizione: `Free=True`

T imposta il valore di `Free` a `False`, il numero di accessi per direzione `Access_Number` a 1, e la direzione corrente `Current_Direction=1`. Esegue infine le operazioni previste dopo aver ottenuto l'accesso.

Soluzione al Problema - Concorrenza - Esempio (3)

Caso 2:

Pre-condizione: `Free=False` and `Current_Direction`$= D_T$ and $1 <$`Access_Number`$<$`MAX`

Poiché sia verificata la Pre-condizione, 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:

Pre-condizione: `Free=False` and `Current_Direction`$= D_T$ and `Access_Number`$=$`MAX` and `Second_End_Count`$= 0$

Poiché sia verificata la Pre-condizione, 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.

Soluzione al Problema - Concorrenza - Esempio (4)

Caso 4: Accesso non consentito

Pre-condizione: `Free=False` and `Current_Direction`$\neq D_T$

Poiché 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

Pre-condizione: `Free=False` and `Current_Direction`$= D_T$ and `Access_Number=MAX` and `Second_End_Count`> 0

Poiché 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 crrente 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`.

Soluzione al Problema - Tecnologie

- Scelta delle tecnologie che meglio si adattano alle scelte di progetto.
- Interessante l'utilizzo di tecnologie eterogenee
  - È difficile pensare ad un Sistema Distribuito realizzato con tecnologie omogenee.
  - Possibilità di utilizzare tecnologie specifiche per singola componente
    - Nella solouzione che ho adottato, ho utilizzato il linguaggio Ada per codificare le componenti che rappresentano le Regioni;
    - Utilizzo del linguaggio Scala per la realizzazione di Name Server, Biglietteria e Controller Centrale;
    - Utilizzo di Javascript e HTML per la realizzazione dell'interfaccia grafica.

Conclusioni