Università degli Studi di Bologna
Facoltà di Ingegneria

Corso di
Reti di Calcolatori M

Chiamate di Procedura Remota:
Java RMI, Deployment e Scalabilità

Luca Foschini

Anno accademico 2014/2015

Specifica (vi ricorda qualcosa??)

Sviluppare **un servizio di nomi distribuito e coordinato** partendo dal RegistryRemoto sviluppato nell'esercitazione svolta (vedi es. 7 del corso di Reti T/L-A, A.A. 2010/2011).

Si ipotizzi che sia disponibile una molteplicità di RegistryRemoti che siano, da un punto di vista logico, in esecuzione su host (e registrati presso rmiregistry) diversi. In tale scenario si vuole realizzare un **FrontEnd per RegistryRemoti** che permetta i seguenti comportamenti:

- i servitori devono poter registrare la propria disponibilità di servizio **presso uno qualsiasi dei RegistryRemoti**;
- il **FrontEnd** deve mantenere una **vista di tutte le registrazioni** (**periodicamente aggiornate**) eseguite presso i vari RegistryRemoti;
- i clienti devono poter ottenere i riferimenti remoti necessari per il servizio di cui hanno bisogno **attraverso il FrontEnd**.

Specifica: il FrontEnd

Il **FrontEnd** è realizzato come server RMI e implementa le seguenti operazioni (si realizzino interfacce con scope diversi per clienti e RegistryRemoti):

- **ricerca del primo riferimento** registrato con un **nome logico** dato;
- **ricerca di tutti i riferimenti** registrati con uno stesso **nome logico**;
- **registrazione di un RegistryRemoto**, dato il solo **riferimento remoto**.

e mantiene due strutture dati:

- una con **i riferimenti ai RegistryRemoti**;
- una con **le corrispondenze nome logico/riferimento remoto per tutti i servitori remoti**.

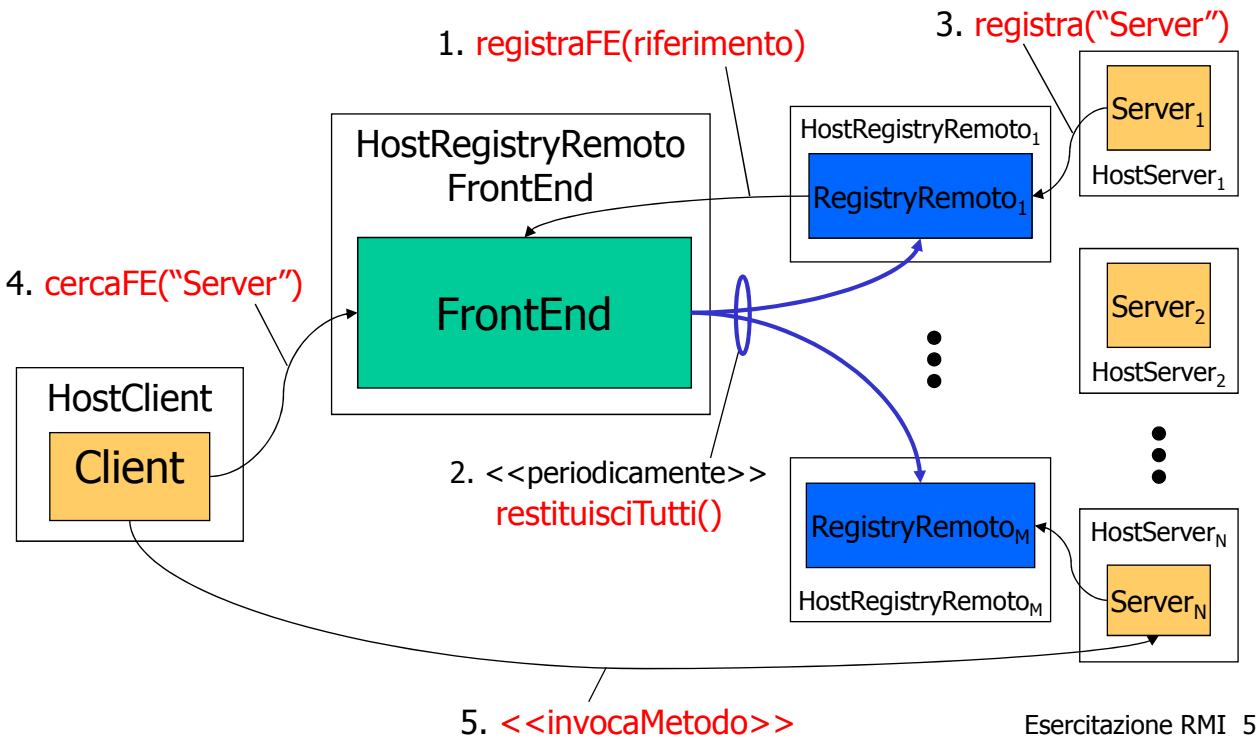
Inoltre, il main del FrontEnd **aggiorna** periodicamente (a polling) **le corrispondenze nome logico/riferimento remoto**.

Metodi remoti

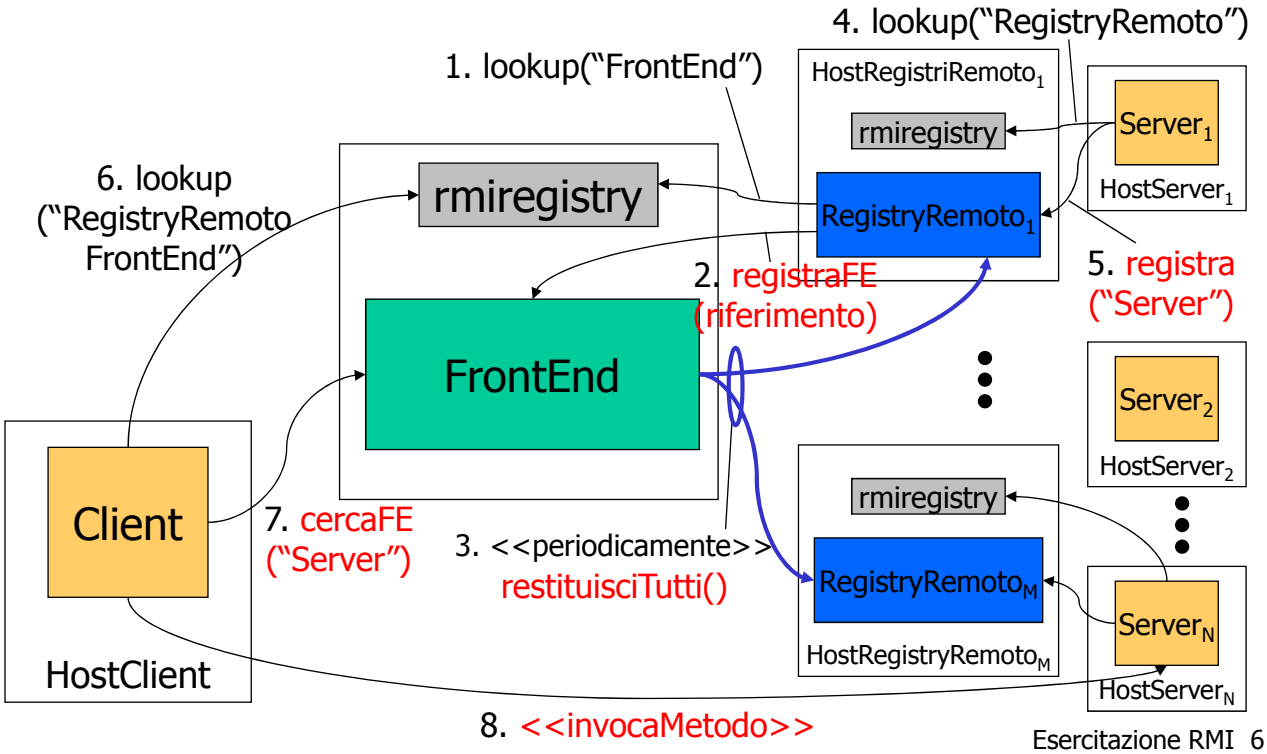
Il progetto RMI si basa su **due interfacce remotizzabili** (una per i client e una per i RegistryRemoti) in cui vengono definiti i **metodi invocabili da client e RegistryRemoti**:

- Il metodo **`cercaFE`** accetta come parametro d'ingresso **il nome del servizio**, quindi restituisce il primo riferimento al servizio richiesto, oppure *null* se il servizio non è disponibile.
- Il metodo **`cercaTuttiFE`** accetta come parametro d'ingresso **il nome del servizio**, quindi restituisce un array contenente tutti i riferimenti corrispondenti al servizio richiesto, oppure *null* se il servizio non è disponibile.
- Il metodo **`registraFE`** accetta come parametro d'ingresso **il riferimento al RegistryRemoto** e lo inserisce nell'opportuna struttura dati.

Architettura di riferimento



Architettura di riferimento: dettagli implementativi



Classi in gioco

Il progetto RMI si compone, oltre alle classi degli oggetti Server, le ulteriori classi :

- **Un'interfaccia remota *RegistryRemotoClient*** (contenuta nel file *RegistryRemotoClient.java*) in cui vengono definiti i metodi invocabili dai clienti (cerca, cercaTutti);
- **Un'interfaccia remota *RegistryRemotoServer*** (contenuta nel file *RegistryRemotoServer.java*) che estende ***RegistryRemotoClient*** aggiungendo i metodi invocabili dai servitori (restituisciTutti, aggiungi, eliminaPrimo, eliminaTutti);
- Una **classe per la realizzazione del RegistryRemoto** (***RegistryRemotoImpl*** contenuta nel file *RegistryRemotoImpl.java*), che implementa i metodi di RegistryRemotoServer invocabili in remoto.

NOTA: possibilità di usare interfacce remote diverse con scope diversi in base al **ruolo** di utilizzo dell'oggetto remoto!!

Particolare attenzione nella realizzazione di Server e Client in modo che effettuino la registrazione e la ricerca del riferimento all'oggetto Server, presso il RegistryRemoto (invece che sull'rmiregistry locale).

Classi in gioco (ancora…)

Inoltre, si progettino le classi:

- **FrontEnd** (contenuta nel file FrontEnd.java), che implementa i metodi del invocabili in remoto e presenta l'interfaccia di invocazione:

FrontEnd [registryPort]

Si modifichi inoltre opportunamente:

- Il main del Client in modo da interrogare il FrontEnd al posto del RegistryRemoto;
- Il main del RegistryRemoto in modo da registrare il riferimento del RegistryRemoto presso il FrontEnd.

Deployment (singolo host)

Le scelte di deployment sono **cruciali** per garantire **buone performance** al crescere del sistema.

Inizialmente, proviamo ad emulare **su un singolo host** lo scenario di deployment mostrato nell'architettura di riferimento facendo uso di più `rmiregistry` attivati su **porte diverse** e si eseguano in modo consistente i programmi **Client** e **Server** lanciandoli con il parametro `rmiRegistryPort` in modo che accedano all'rmiregistry corretto.

Deployment (singolo host)

Si provi poi a caricare i **Server** sfruttando i **Client** come generatori di traffico.

Quale è il modo migliore per testare le **prestazioni del sistema**? Quali sono i **test significativi**?

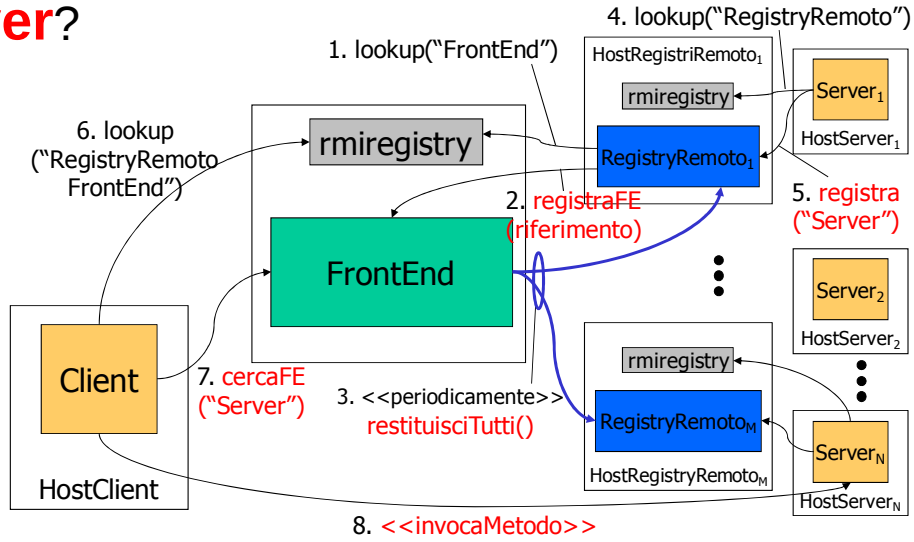
Come arrivare alla **saturazione delle risorse**?

Effettuare un po' di prove variando il **numero di istanze** di Client (per singolo oggetto Server) e la **frequenza di invio di richieste** da parte dei singoli Client. Cosa incide maggiormente?

SCALABILITÀ

Deployment (con più host)

Si provi quindi a realizzare scenari di deployment più complessi, **su un numero di host maggiore**.
Come organizziamo il deployment? Su quali nodi attivare i vari **servizi di registry**, i **Client** e i **Server**?



Deployment (con più host)

Effettuate le stesse prove di performance pensate per il caso di deployment su singolo nodo.

Quali **differenze** riscontrate?

NOTA BENE: la comparazione **DEVE** essere **quantitativa** ☺, non solo **qualitativa** ☹