Service Oriented Architecture
(Unix)

UNIVERSITÀ DI PISA

CdL Magistrale in Ingegneria Informatica

Progetto di Sistemi Operativi e Programmazione Distribuita

a.a. 2010-11

Agostino Polizzano

admin@agostinopolizzano.info

Contents

1 Presentazione sintetica del progetto

Il progetto realizza una infrastruttura che implementa una architettura "*Service Oriented*".

Il sistema consiste di un **Registro dei Servizi**, uno o più **Service Provider** e uno o più **Client**.

Vidiamo più in dettaglio le funzioni di ciascun componente.

1.1 Service Provider e Servizi

Un Service Provider è un processo che mette a disposizione alcuni servizi per i Client che ne fanno richiesta. Un **servizio** può essere visto come una funzione che:

- ha un nome (che identifica univocamente il servizio nel sistema);
- prende dei parametri in ingresso
- fornisce dei parametri in uscita

Un servizio è quindi descritto da una stringa contenente il nome e da una lista ordinata di parametri, ognuno dei quali contiene un tipo e una direzione (IN o OUT).

Il **Service Provider** è un processo di tipo server che sta in attesa di richieste di utilizzo di un servizio da parte dei client. Quando una richiesta arriva, il Service Provider fa il parsing della richiesta e di tutti i parametri, ne verifica la correttezza, e la processa. Al termine manda la risposta al client. La risposta contiene i parametri di uscita (se sono previsti).

Il service provider è un programma concorrente; esso serve tutte le richieste in arrivo in parallelo, se necessario utilizzando delle primitive semaforiche per regolare l'accesso alle risorse condivise interne (dipendentemente dal tipo di servizio).

1.2 Registro dei Servizi

Il **Registro dei Servizi** è un processo che si occupa di registrare la corrispondenza tra servizio e Service Provider. Infatti, ServiceProvider differenti possono supportare servizi diversi oppure lo stesso servizio. Ogni Service Provider è identificato univocamente dalla coppia *<indirizzo IP, porta>*. Il Registro dei Servizi associa il nome del servizio con il Service Provider. Un Client che vuole usufruire di un servizio prima chiede al Registro dei Servizi quali Service Provider forniscono un determinato servizio (in realtà un Client potrebbe anche evitare questa richiesta qualora questo conoscesse già l'indirizzo del Service Provider che fornisce il servizio desiderato); quindi indirizza la propria richiesta direttamente al Service Provider indicato dal Registro. Poichè più di un Service Provider può fornire lo stesso servizio, il Registro dei Servizi usa una qualunque politica FIFO per selezionare uno dei Service Provider corrispondenti al servizio

richiesto. Il registro 'e dinamico: all'inizio, nessun servizio è registrato. Un Service Provider registra un servizio mandando un opportuno messaggio al Registro; successivamente, un Service Provider può volersi cancellare dal Registro, oppure de-registrare un suo servizio.

1.3 Client

Il Client che ha bisogno di utilizzare un servizio si può prima si rivolgere al Registro dei Servizi, chiedendo l'indirizzo e la porta del Service Provider a cui rivolgersi. Successivamente interagisce direttamente con il Service Provider.

2 Aggiunta di un nuovo servizio

Di seguito verrà spiegato come implementare un nuovo servizio che un generico Service Provider può mettere a disposizione dei Clients.

2.1 Implementazione del servizio

Il servizio sarà completamente implementato in un header file, che supponiamo sia NameService.h e dovrà essere inserito nella cartella *services* del Service Provider.

Il file NameService.h dovrà avere struttura analoga alla seguente:

```
#ifndef _NAMESERVICE_H_
#define _NAMESERVICE_H_

#include ...
#include ...

void name_service_body (void**& in, void**& out) {

        //codice che esplicita il servizio

}

Service* NameService = new Service (name_service, name_service_body);

#endif
```

ove ***in*** è un puntatore ai parametri in ingresso i quali sono così strutturati (tale sarà il modo in cui il Service Provider passerà al Servizio i parametri ricevuti da Client, quindi bisogna progettare il servizio con questo presupposto):

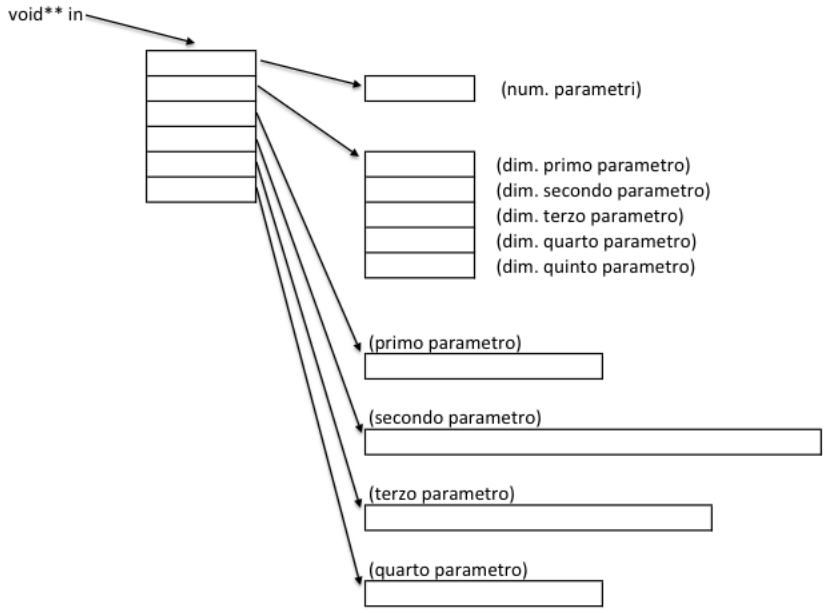

Fig.1 - schema parametri in ingresso

quindi:

```
int num_par_in=*(int*)*(in) //numero dei parametri in ingressi
int dim_par_i = *((int*)*(in+1)+i) //dimensione (in bytes)
                                   //dell'i-esimo parametro in
                                   //ingresso (i=0,...,n con n
                                   //pari al numero dei
                                   //parametri in ingresso)
//...
void* par_i = *(in+2+i) //puntatore al parametro i-esimo
//...
```

Osservazione: il tipo dei parametri in ingresso deve essere conosciuto a priori in quanto saranno quelli che ci si aspetta che siano per il servizio che si sta implementando. Il Client che invoca il servizio in questione dovrà inviare il numero dei parametri in ingresso che il servizio prevede e del tipo opportuno. Eventuali errori da parti del Client, quindi parametri in numero, tipo o dimensione errata, dovranno essere gestiti dal servizio stesso. Errori nell'implementazione del servizio potrebbero compromettere la stabilità, oltre che del servizio stesso, anche del

Service Provider che lo ospita. La situazione di errore può essere segnalata semplicemente facendo terminare il servizio ponendo out=NULL (il Service Provider è progettato per gestire questa situazione di errore che verrà opportunamente segnalata al Client senza correre il rischio di breakdown per il Service Provider).

Per quanto concerne i parametri in uscita è sufficiente costruirli in modo analogo per quanto visto per i parametri in ingresso (tali parametri saranno puntati da ***out***)

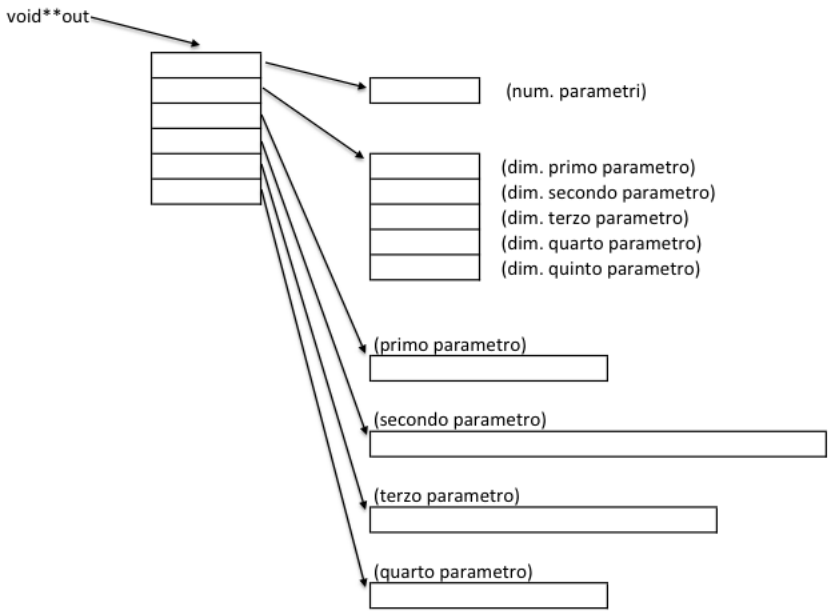

Fig.2 - schema parametri in uscita

Quindi, ad esempio, se il servizio che si intende implementare avrà 3 parametri di uscita, si dovrà procedere in modo analogo al seguente:

```
out = new void*[5]; //2 + numero parametri in uscita
*(out) = new int;
*(int*)*(out) = 3; //numero dei parametir in uscita
*(out + 1) = new int[3]; //vettore della dimensione di
                         //ciascun parametro in uscita

*((int*)*(out + 1) + i) = xx; //dimensione del parametro
                    //i-esimo in uscita (i=0,...,n con n pari
```

```
                    //al numero dei parametri in uscita)
//...
*(out+2+i) = new char[xx]; //si alloca la memoria
                    //per l'i-esimo parametro in uscita
//...

//quindi bisogna inserire nella in tale memoria allocata
//l'i-esimo parametro in uscita
```

Osservazione: all'interno dell'header file NAMESERVICE.H si possono implementare altre strutture dati o altri metodi utili al sevizio stesso, ad esempio:

```
#ifndef _NAMESERVICE_H_
#define _NAMESERVICE_H_

#include ...
#include ...

class item_aux {
        //...
};

struct struct_aux {//...
};

int func_aux () {
        //...
}

void name_service_body (void**& in, void**& out) {

        //codice che esplicita il servizio
                //item_aux aux1;
                //struct_aux aux2;
                //func_aux();

}

Service* NameService = new Service (name_service, name_service_body);

#endif
```

2.2 Inclusione del servizio in un Service Provider

A questo punto, implementato il servizio, questo è pronto per essere inserito tra i servizi offerti dal ServiceProvider. Per compire questa operazione è sufficiente includere l'header file relativo al servizio in questione (NAMESERVICE.H) nel main del ServiceProvider ed invocare il metodo *AddService(NameService)* del ServiceProvider ove NomeService è il puntatore all'oggetto *NameService* creato nell'header file del servizio.