Laboratorio
Programmazione concorrente e distribuita

Simone Albertini
11 giugno 2013

Come si lavora oggi

- Ognuno sulla sua postazione
- Svolgere l'esercizio sulla traccia
- Si possono scambiare opinioni, consultare materiale, fare domande.
- Tempo: 3 ore.

http://bit.ly/13xtUwX

Problema

- *Realizzare un sistema client-server per registrare i ri sultati dello spoglio di una elezione*
- *Concettualmente uguale al primo esercizio di laboratorio ma qui lo strutturiamo in un contesto diverso.*

Requisiti funzionali

- I Client
  - Inviano i loro risultati
  - Possono chiedere chi sta vincendo
  - Possono chiedere i risultati
- Il server
  - Aggrega i risultati inviati dai client
  - Risponde alle loro richieste

Pattern Proxy

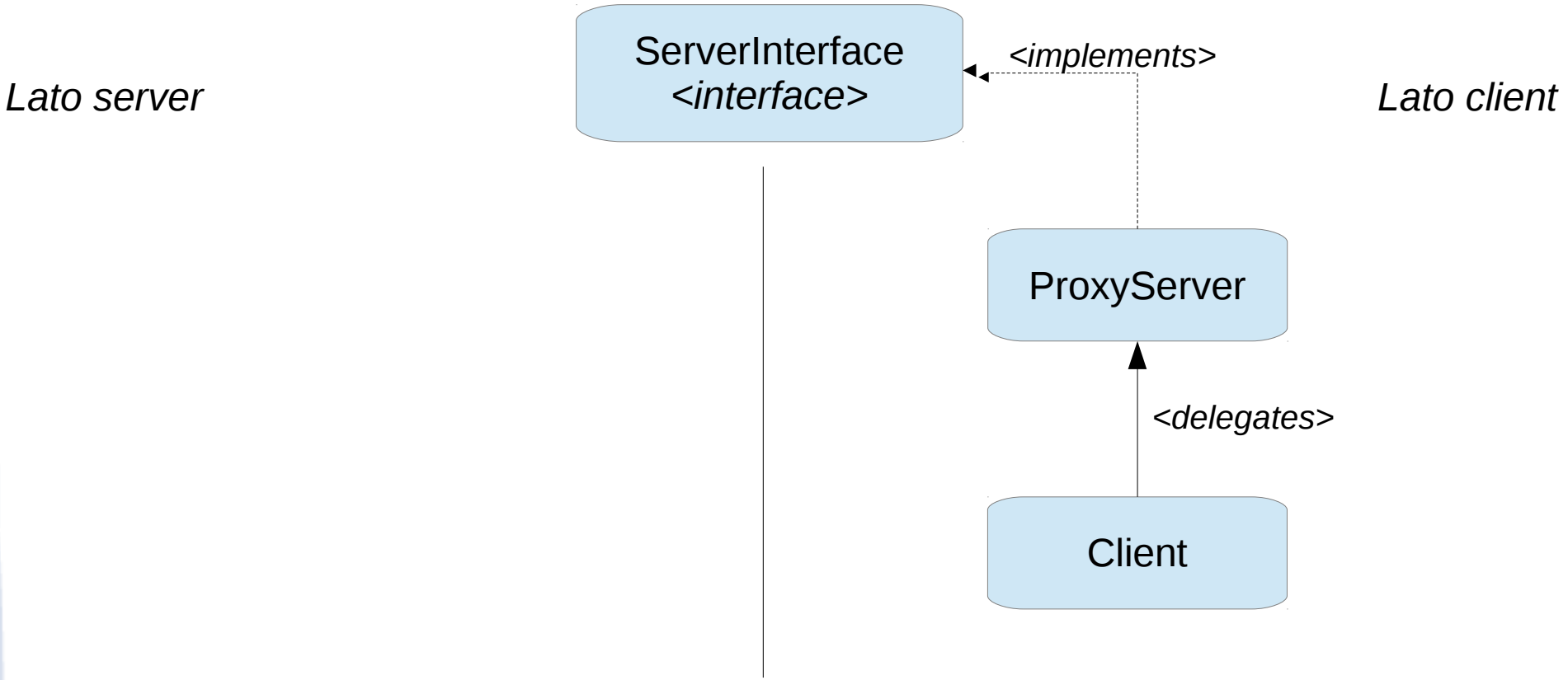
Lato server

ServerInterface
<interface>

Lato client

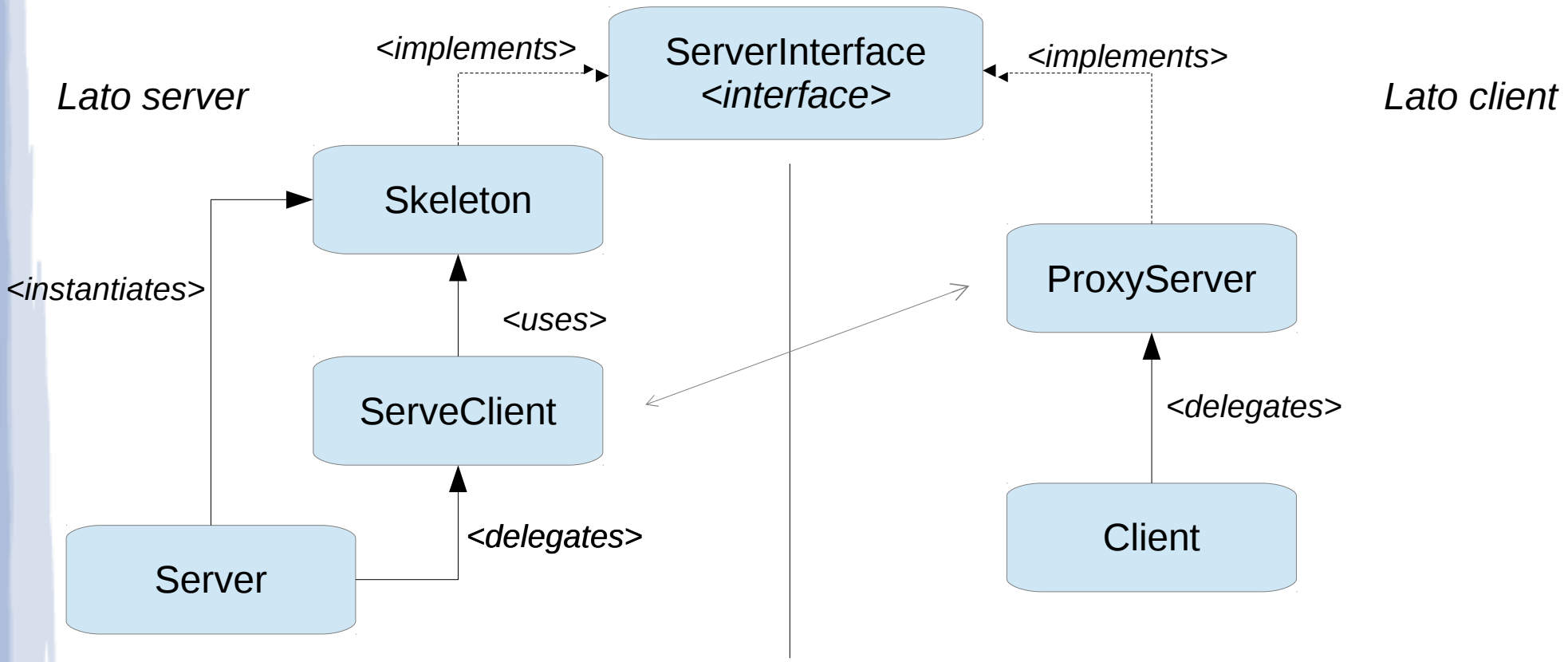
Si definisce una interfaccia che funge di fatto da API lato client e definisce le funzionalità offerte dal sistema.

Pattern Proxy



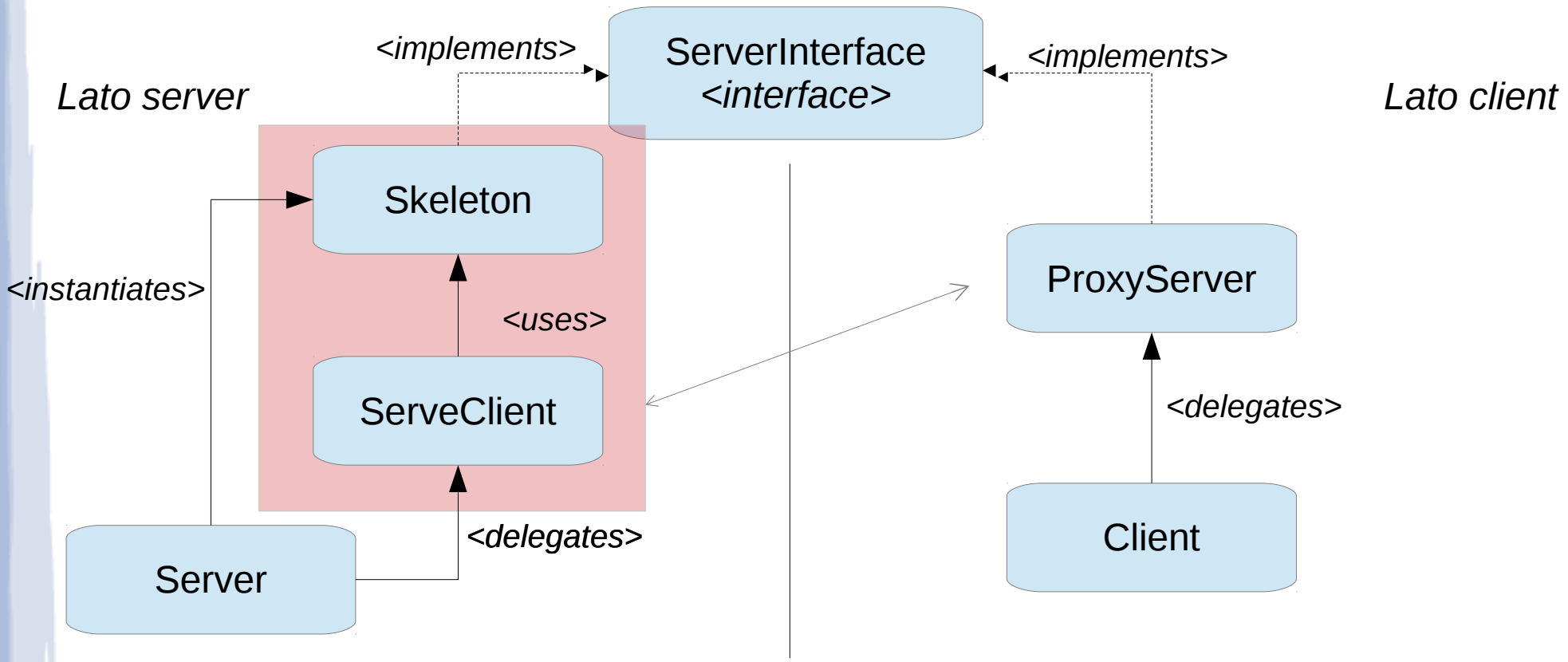
Lato client, il ProxyServer implementa le funzionalità da ServerInterface e implementa il protocollo di comunicazione

Pattern Proxy



Lato server, uno skeleton implementa le funzionalità. Viene condiviso tra i thread che gestiscono la comunicazione coi client

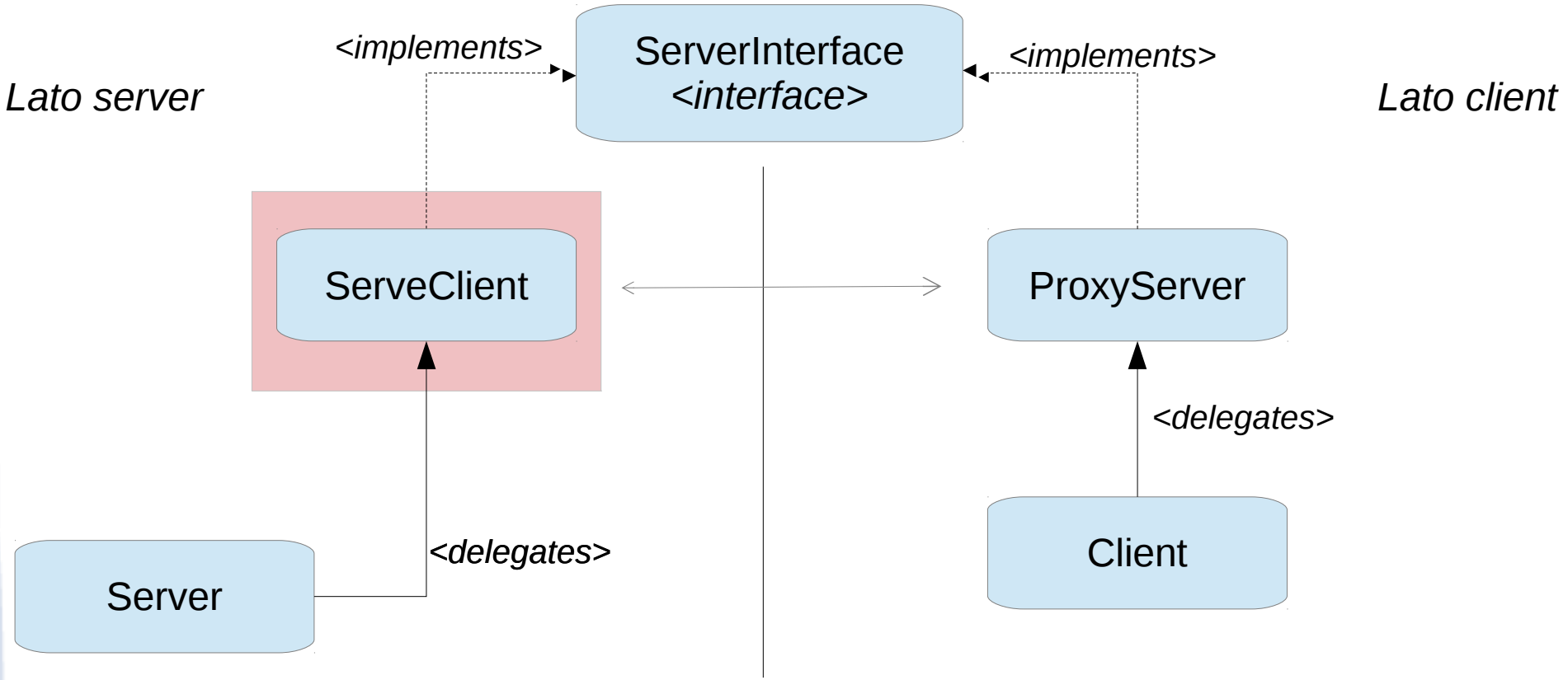
Pattern Proxy



In problemi semplici, questi due elementi possono essere uniti in un unico elemento.

In questo esercizio è consigliato farlo.

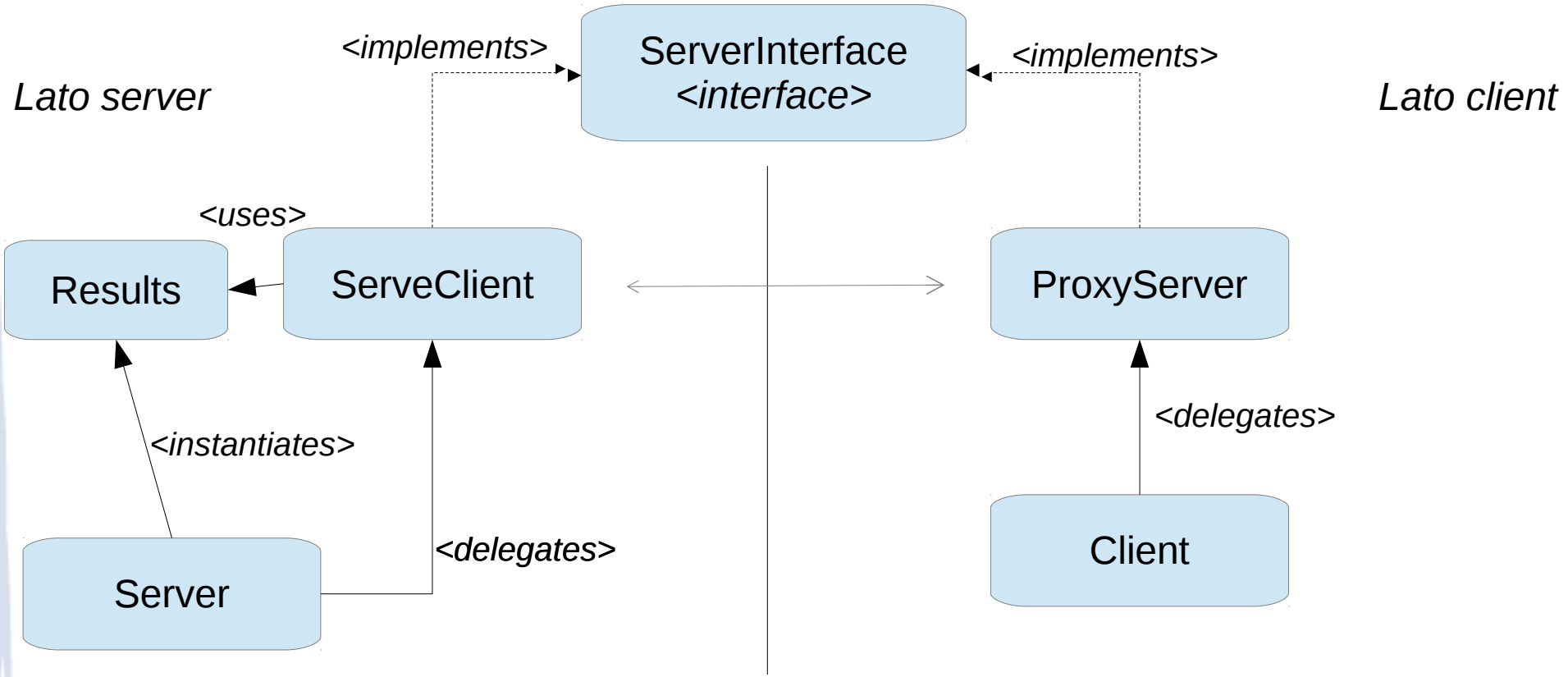
Pattern Proxy



In problemi semplici, questi due elementi possono essere uniti in un unico elemento.

In questo esercizio è consigliato farlo.

Pattern Proxy



L'oggetto che tiene il conteggio dei voti (già implementato) è istanziato dal server e condiviso tra i thread

Protocollo di comunicazione

- Tutto tramite oggetti String scritti sullo stream della socket.

1) Add results
   - Client invia: <add> party votes
   - Server ritorna: nulla
   - Esempio:
     - <add> partito 5

       Deve aggiungere 5 voti a “partito”

Protocollo di comunicazione

2) Winner

- Client invia: <winner>
- Server ritorna: nomePartito
- Deve restituire sullo stream il nome del partito che ha più voti

3) Results

- Client invia: <results>
- Server ritorna: partito1 voti1 partito2 voti2 [ecc]
- Ciascun partito con il suo relativo totale dei voti in una uncia stringa separati da spazi.

Protocollo di comunicazione

4) Close

- Client invia: <close>
- Server risponde: nulla
- Il server chiude la comunicazione così come il client.