Remote Procedure Call

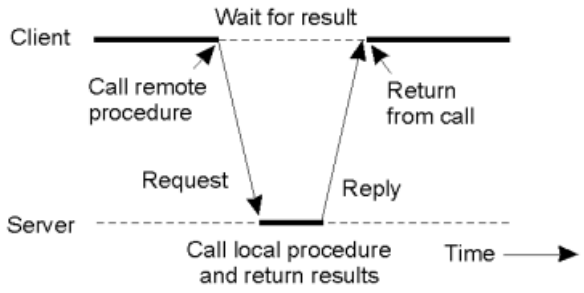
Le procedure remote (RPC) sono una estensione al distribuito del normale protocollo di chiamata di procedura.

Vantaggi	Svantaggi
• Hanno una semantica nota: chiamata di procedura • Sono facili da implementare: vicine al modello client-server	• Realizzate dal programmatore: tutto è esplicito (si può ovviare usando modelli a componenti più sofisticati) • Sono statiche: scritte nel codice dei programmi (si può ovviare usando chiamate indirette) • Non c'è concorrenza: sono bloccanti (si può ovviare con un uso appropriato dei thread o chiamate asincrone)

6.1 Modello RPC

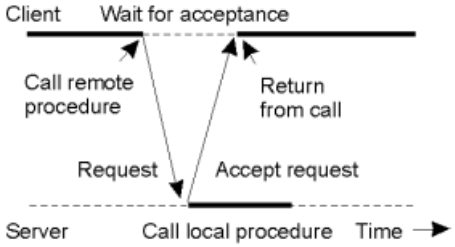
Il modello RPC maschera il modello client-server.
La differenza principale sta nella gestione delle informazioni scambiate: nel modello RPC si utilizzano i parametri.

6.1.1 Modello sincrono

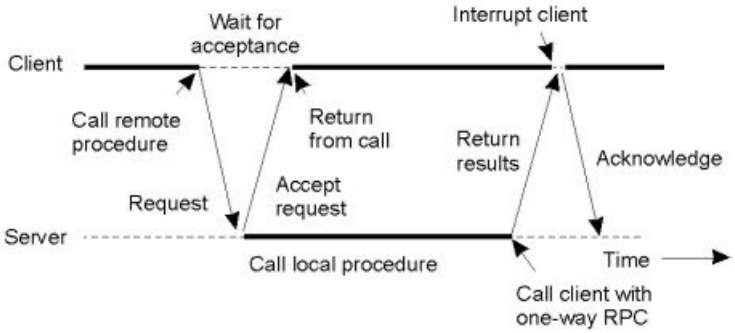


1. **Richiesta RPC dal client**: Il client chiama una procedura o un metodo in modo locale, come se fosse una chiamata di funzione normale. I parametri della chiamata di procedura, insieme a eventuali dati aggiuntivi, vengono incapsulati in un pacchetto di dati da inviare attraverso la rete al server.
2. **Ricezione della richiesta da parte del server**: Il server riceve il pacchetto contenente la richiesta RPC. Il server decodifica il pacchetto per estrarre i parametri della chiamata di procedura e identificare la procedura richiesta.
3. **Esecuzione della procedura**: Il server decodifica il pacchetto per estrarre i parametri della chiamata di procedura e identificare la procedura richiesta. Una volta completata l'esecuzione della procedura, il server genera una risposta che può includere eventuali risultati o valori di ritorno. La risposta viene incapsulata in un pacchetto di dati da inviare al client.
4. **Trasmissione della risposta**: Il pacchetto contenente la risposta viene inviato al client attraverso la rete.
5. **Ricezione della risposta da parte del client**: Il client riceve il pacchetto contenente la risposta RPC. Il client decodifica il pacchetto per estrarre la risposta restituita dalla procedura remota.
6. **Ritorno alla chiamata di procedura locale**: Il client riceve i risultati come se fosse una chiamata di procedura locale e può continuare l'esecuzione del programma utilizzando tali risultati.

6.1.2 Modello asincrono



Interazione tra client e server utilizzando una RPC asincrona, senza risposta. Il client rimane in attesa solo fino a quando il server non riceve la richiesta.



Interazione tra client e server utilizzando una RPC asincrona, con riposta posticipata utilizzando una seconda RPC asincrona (una call-back). Il client rimane in attesa solo fino a quando il server non riceve la richiesta. Una volta che il client riceve la risposta dal server, fa un interrupt e manda un ack al server.

6.2 Architettura RPC

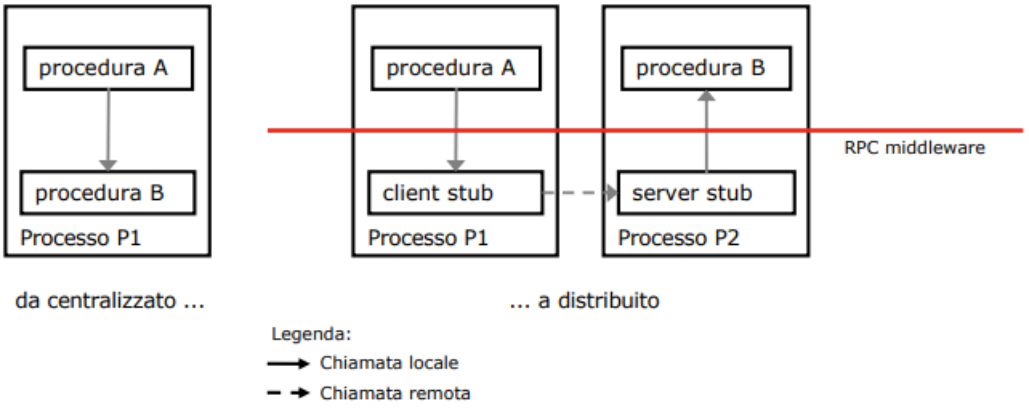
Una Remote Procedure Call (RPC) non può essere diretta, deve essere gestita da un middleware che si preoccupa di trasformare le chiamate locali in chiamate remote.

6.2.1 Stub

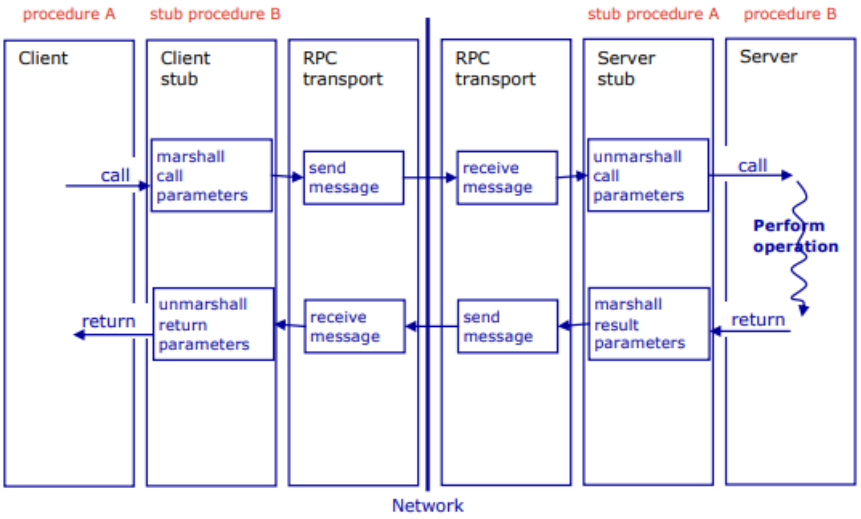
Nell'architettura RPC, uno stub è un componente software che agisce come un rappresentante locale della procedura remota sul lato del client. Quando un'applicazione client invoca una procedura remota, lo stub locale viene chiamato al posto della procedura remota stessa. Il compito principale dello stub è quello di incapsulare i parametri della chiamata di procedura, trasmetterli attraverso la rete al server remoto e gestire la comunicazione con il server.

In sintesi:

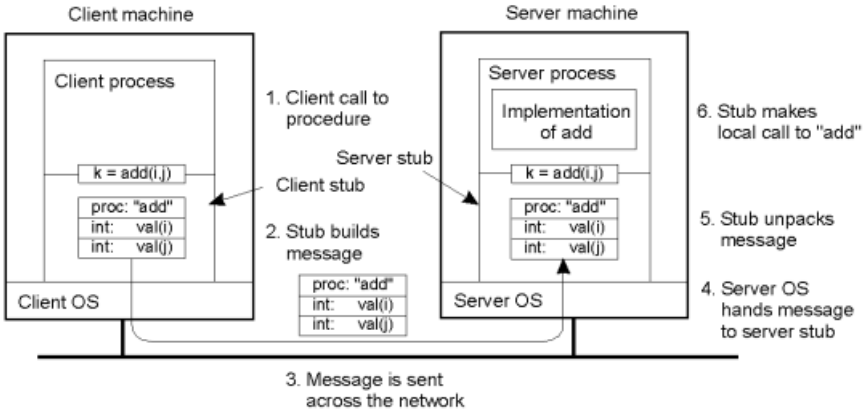
- Simula i comportamenti locali per chiamante e chiamato;
- Realizzare la comunicazione tra processi remoti.



6.3 Esecuzione di una RPC



6.3.1 Passaggio dei valori mediante parametri

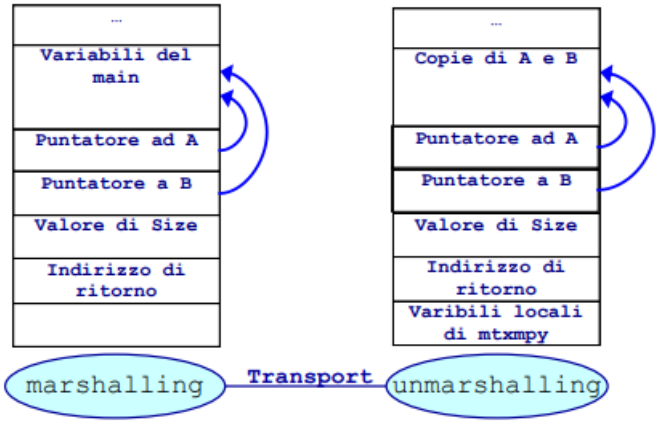


1. **Chiamata di procedura dal client**: Il client invoca una procedura remota come se fosse una chiamata di funzione locale.
   Ad esempio, può chiamare una funzione `add(i, j)`.
2. **Interazione con lo stub client**: Quando il client chiama la procedura remota, in realtà chiama una funzione fornita dallo stub locale.
   Lo stub locale riceve la chiamata di procedura e i relativi parametri, come `i` e `j`.
   Lo stub incapsula i parametri della chiamata di procedura in un formato adatto per la trasmissione sulla rete.
   Questo può includere la serializzazione dei dati in un formato compatibile con la comunicazione di rete.
   Esempio: prodotto tra matrici

```
void main() {
        Matrix A, B, C;
        int size = ...;
        ...
        C = call mtxmpy(A, B, size);
        ...
    }
```

lo stack dopo la chiamata

3. **Trasmissione dei parametri**: Una volta che i parametri sono stati incapsulati, lo stub trasmette il pacchetto contenente i parametri attraverso la rete al server remoto.
4. **Ricezione dei parametri da parte del server**: Il server riceve il pacchetto contenente i parametri della chiamata di procedura. Sul lato server, c'è uno stub remoto che interagisce con il server per ricevere i parametri trasmessi.
5. **Decodifica dei parametri da parte dello stub remoto**: Lo stub remoto decodifica il pacchetto ricevuto per estrarre i valori dei parametri. Questo processo può includere la deserializzazione dei dati nel formato originale. L'operazione si chiama ***marshalling/unmarshalling*** dei dati.



6. **Esecuzione della procedura sul server**: Una volta che i parametri sono stati decodificati, il server esegue la procedura remota utilizzando tali valori.