Luca Cabibbo

Architettura dei Sistemi Software

Invocazione remota

dispensa asw430

ottobre 2024

The core idea of RPC is to hide the complexity of a remote call. Many implementations of RPC, though, hide too much.

Sam Newman

- Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 23, Invocazione remota
- Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. **Distributed Systems**: Concepts and Design, fifth edition. Pearson, 2012.
- Tanenbaum, A.S. and Van Steen, M. **Distributed Systems**: Principles and Paradigms, second edition. Pearson, 2007.
- Richardson, C. **Microservices Patterns**: With examples in Java. Manning, 2019.

- Obiettivi e argomenti

- Obiettivi
  - introdurre l'invocazione remota e discutere la sua implementazione
  - discutere la semantica dell'invocazione remota – e come differisce dalla semantica dell'invocazione locale
  - discutere alcune varianti dell'invocazione remota
- Argomenti
  - invocazione remota
  - implementazione dell'invocazione remota
  - semantica dell'invocazione remota
  - ulteriori aspetti e varianti dell'invocazione remota
  - discussione

* Invocazione remota

- L'*invocazione remota* è una delle principali astrazioni di programmazione distribuita implementate dal middleware
  - è una versione "remota" (distribuita) dell'invocazione "locale" di operazioni
  - consente interazioni di tipo client/server (di tipo richiesta/risposta e sincrone) tra componenti
  - è implementata da molte tecnologie di middleware
    - ad es., RPC, RMI, nelle tecnologie a componenti e nelle tecnologie a servizi
  - discutiamo gli aspetti fondamentali dell'invocazione remota – ma senza far riferimento a tecnologie specifiche

Chiamata di procedure remote (RPC)

- La *chiamata di procedure remote* (*Remote Procedure Call* o *RPC*) è un'astrazione di programmazione distribuita fondamentale
  - è una delle prime soluzioni di middleware realizzate
  - consente a un componente (*client*) di chiamare (invocare) una "procedura" (*operazione*) di un componente remoto (*server*), affinché questi esegua l'operazione richiesta

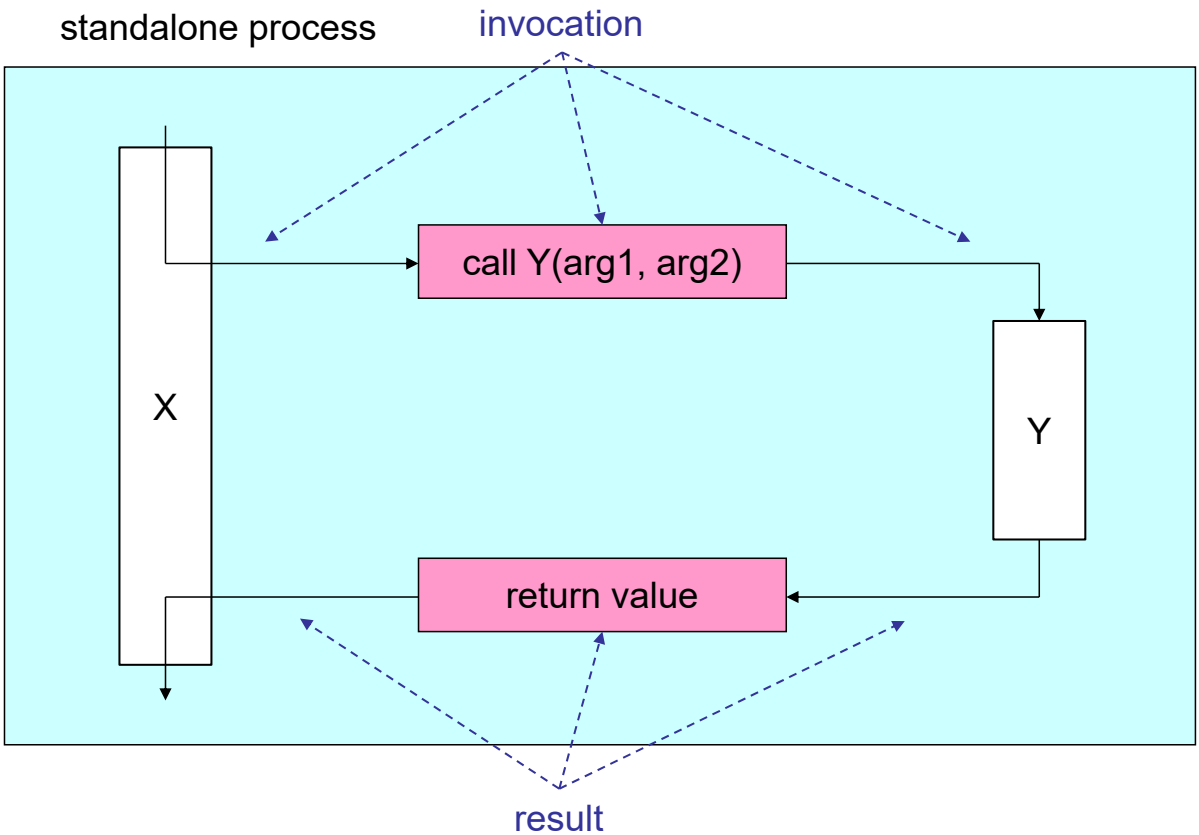
Remote Procedure Call

- *RPC* (*Remote Procedure Call*)
  - In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network), which is coded as if it were a normal (local) procedure call, without the programmer explicitly coding the details for the remote interaction.
    That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote. […]
    In the object-oriented programming paradigm, RPC calls are represented by remote method invocation (RMI).
    [Wikipedia, 2020]

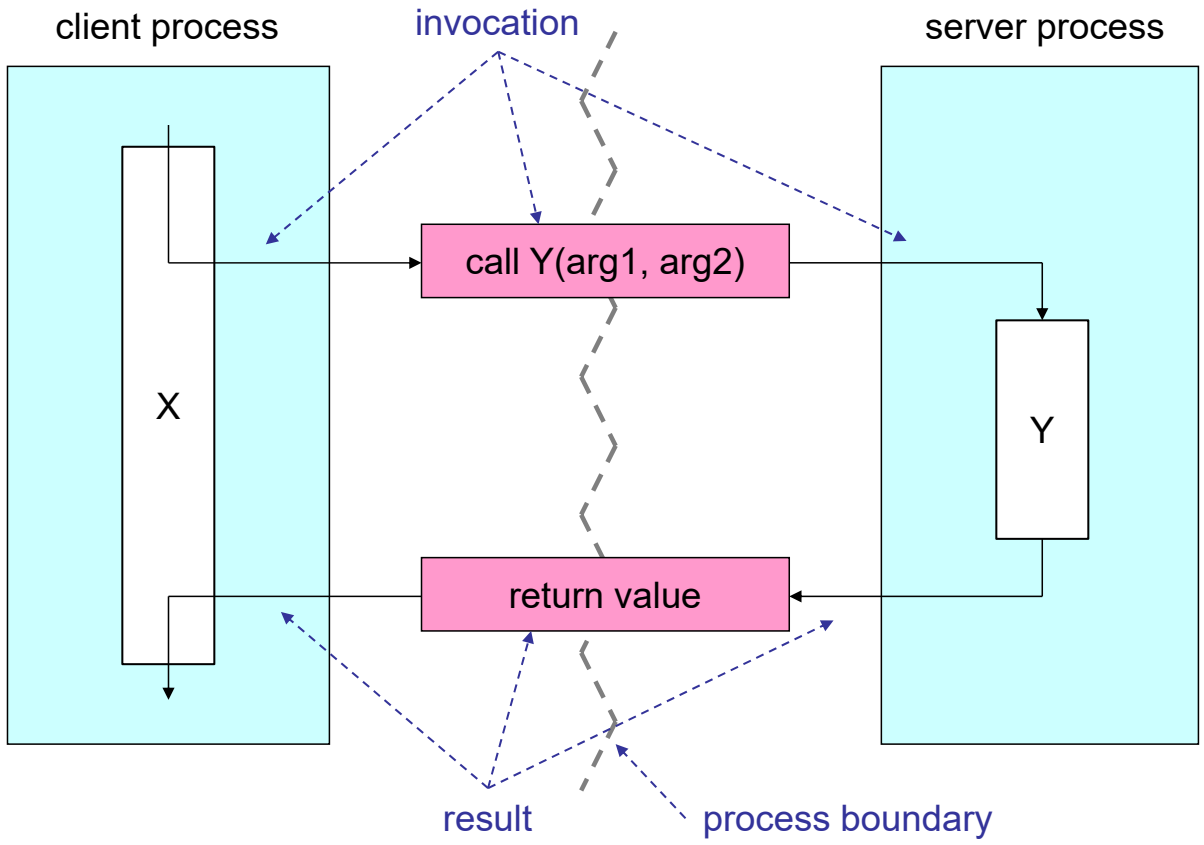
Invocazione di metodi remoti (RMI)

- L'*invocazione di metodi remoti* (*Remote Method Invocation* o *RMI*) è un'estensione OO di RPC
  - un servizio di comunicazione tra *oggetti distribuiti* o *oggetti remoti* – componenti distribuiti realizzati con tecnologie a oggetti e in esecuzione in processi separati
  - a parte l'adozione di un modello a oggetti, un'invocazione di metodo remoto (in RMI) corrisponde essenzialmente a una chiamata di procedura remota (in RPC)
  - nel seguito parleremo semplicemente di "operazioni" e di "invocazioni" "remote"

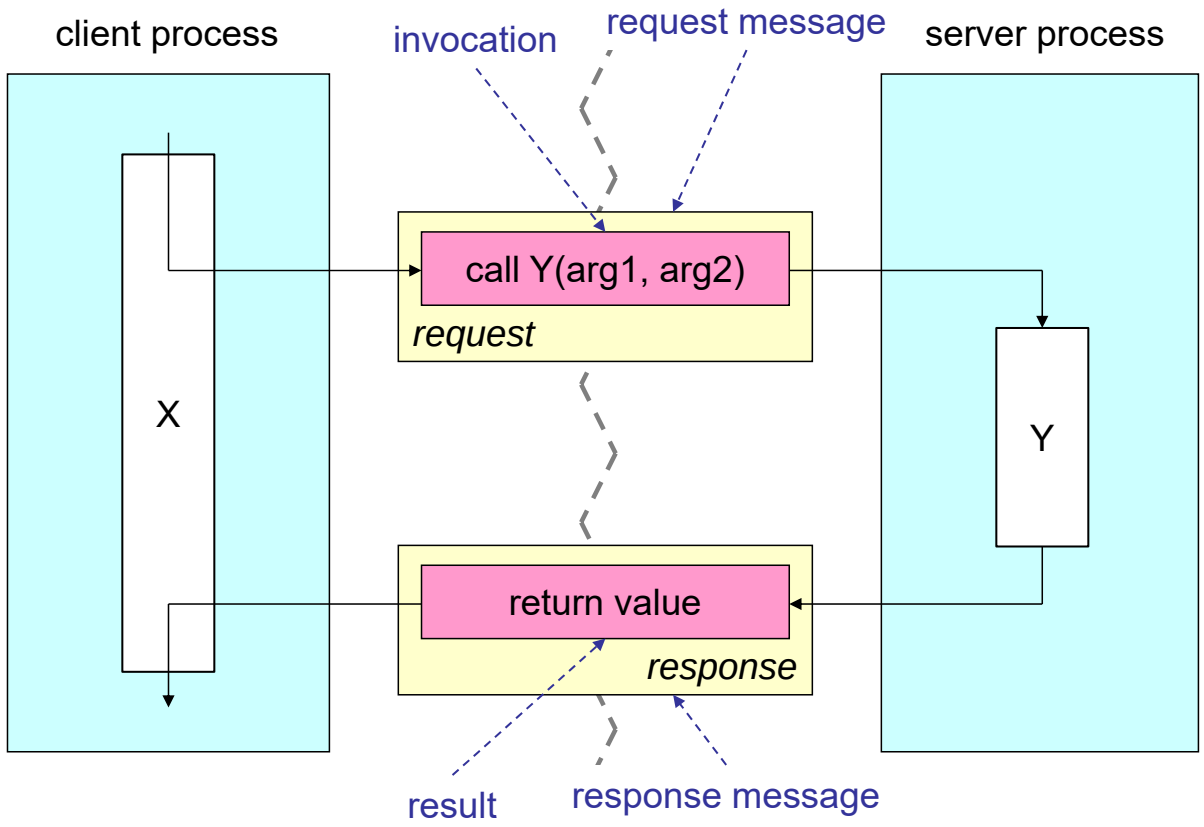
Un'invocazione "locale"



Un'invocazione remota



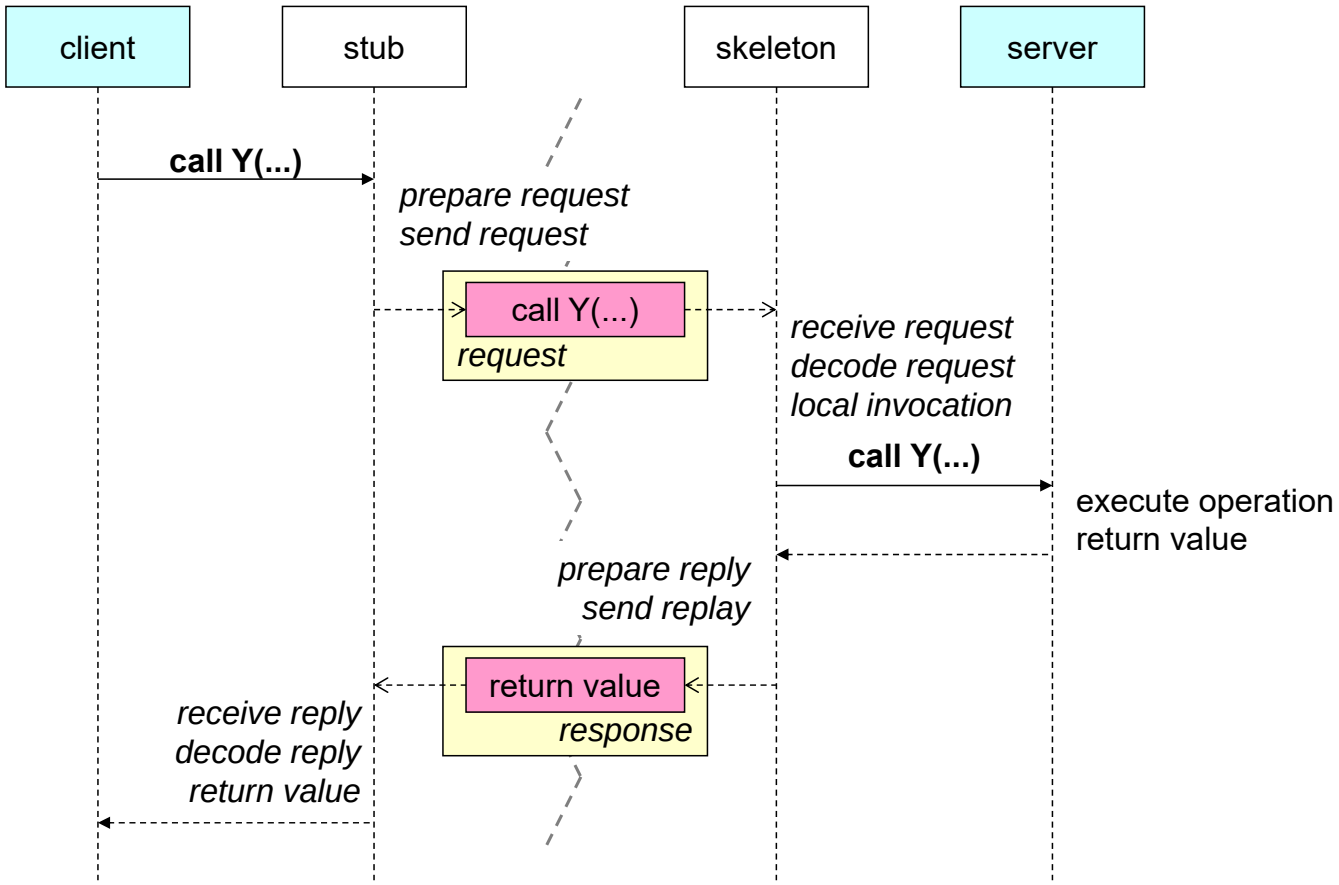
Un'invocazione remota



Paradigma dell'invocazione remota

- Un'invocazione remota
  - sintatticamente, l'operazione remota **Y** viene chiamata da **X** come se fosse un'invocazione locale
  - in realtà, l'operazione **Y** vive nel processo server – che è separato dal processo client in cui vive **X**
  - l'esecuzione di un'invocazione remota comprende certamente le attività consuete – legame dei parametri, esecuzione dell'operazione e restituzione dei risultati
  - l'implementazione è basata su un protocollo richiesta-risposta – che prevede lo scambio di un messaggio di richiesta e di uno di risposta
  - il client e il server interagiscono in modo sincrono

* Implementazione dell'invocazione remota



Invocazione remota e proxy remoti

- Nell'invocazione remota, la comunicazione tra i componenti *client* e *server* avviene tramite un connettore realizzato come una coppia di proxy remoti
  - *stub* – o *proxy lato client*
  - *skeleton* – o *proxy lato server*
  - un *proxy* [GoF] fornisce un surrogato o un segnaposto per un altro oggetto – per controllarne l'accesso
  - i proxy remoti nascondono al programmatore il fatto che la comunicazione sia distribuita

Invocazione remota, client e server

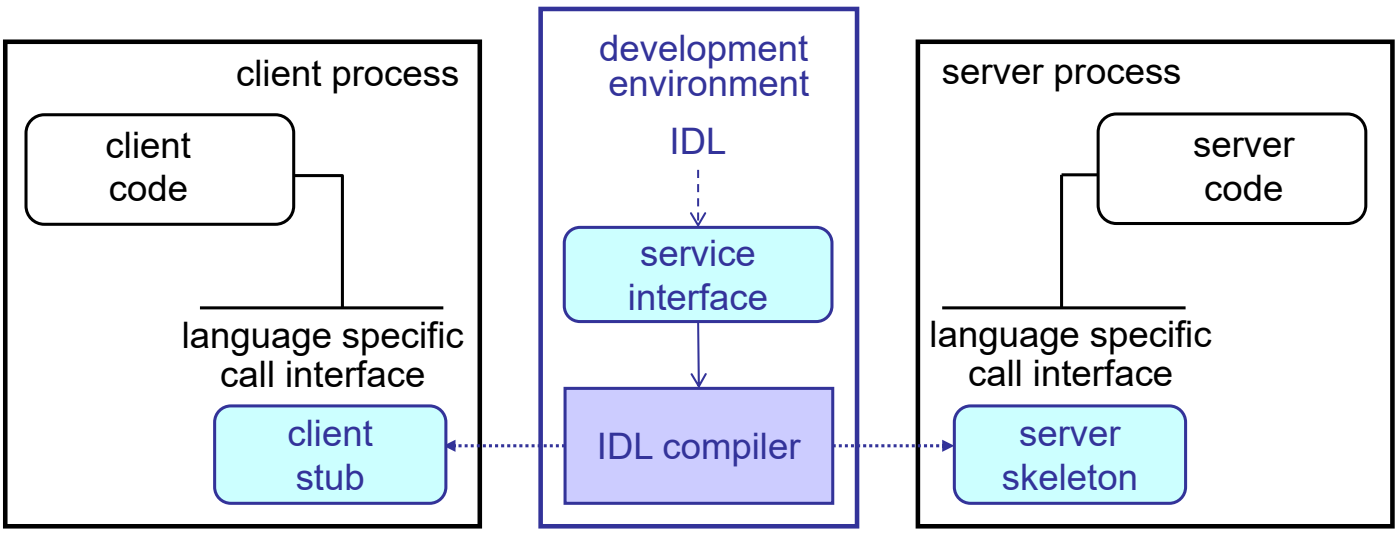
- Prima di andare avanti, è importante comprendere la terminologia utilizzata
  - va notata in particolare la distinzione tra i seguenti termini – e le relazioni tra di essi
    - componente client e componente server
    - proxy lato client e proxy lato server
    - processo client e processo server
  - quando è scritto semplicemente "client" e "server", senza qualificazioni, bisogna intendere "componente client" e "componente server"

Invocazione remota e proxy remoti

- I due proxy remoti implementano un protocollo richiesta-risposta per l'invocazione remota
  - le responsabilità principali dei proxy remoti
    - inviare e ricevere i messaggi di richiesta e risposta
    - preparare e decodificare i messaggi di richiesta e di risposta
  - con RPC e RMI, i proxy vengono generati automaticamente
  - l'implementazione di questi proxy fa in genere riferimento a dei moduli di comunicazione generali

Generazione automatica dei proxy

- La generazione automatica dei proxy remoti è basata su
  - un linguaggio per la definizione di interfacce – *Interface Definition Language* (*IDL*)
  - un *compilatore d'interfacce*



Moduli di comunicazione

- L'implementazione dei proxy remoti è basata su degli opportuni *moduli di comunicazione*
  - questi moduli implementano il protocollo richiesta-risposta sottostante all'invocazione remota
  - si occupano soprattutto dello scambio dei messaggi di richiesta e di risposta
  - hanno un ruolo fondamentale nella definizione della *semantica dell'invocazione remota*

* Semantica dell'invocazione remota

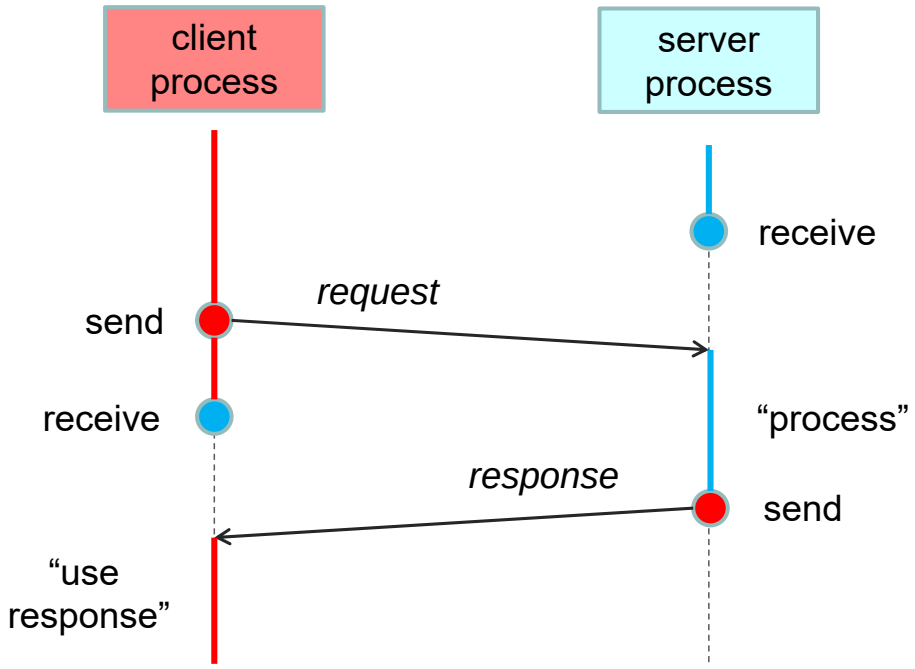
- L'invocazione di un'operazione remota viene scritta, sintatticamente, allo stesso modo di un'invocazione "locale"
  - ma la semantica di un'invocazione remota è la stessa di un'invocazione "locale"?
  - no!!! ecco alcune importanti differenze
    - le operazioni sono eseguite in processi e spazi degli indirizzi separati
    - si possono verificare dei problemi di comunicazione
    - il legame dei parametri e dei risultati è gestito diversamente
  - discutiamo ora questi aspetti

- Problemi di affidabilità

- Nei sistemi distribuiti si possono verificare diversi problemi di affidabilità
  - fallimenti nella comunicazione
  - fallimenti nei processi coinvolti
- A causa di questi (o di altri) problemi, un'invocazione remota potrebbe terminare, per un client, con la ricezione di un'*eccezione remota*
  - che cosa vuol dire?
  - chi può ricevere un'eccezione remota? e chi la può sollevare?

Problemi di affidabilità

- Questo è uno scenario di successo nell'invocazione remota



  - in quali casi si può verificare un'eccezione remota?

Problemi di affidabilità

- Per semplicità, consideriamo solo la possibilità di perdere messaggi (richieste o risposte)
  - l'uso di un protocollo di rete a livello di trasporto "affidabile" (come TCP) non garantisce l'affidabilità della comunicazione remota
  - i moduli di comunicazione devono considerare esplicitamente la possibilità di perdere messaggi nella gestione di un'invocazione remota
  - i moduli di comunicazione possono essere realizzati per tollerare la perdita di alcuni messaggi – la loro implementazione ha impatto sulla semantica dell'invocazione remota

- Semantica dell'invocazione remota

- Si consideri un client C che effettua un'invocazione remota di un'operazione O di un servizio (server) S
  - a tal fine, verranno scambiati alcuni messaggi in rete e verranno svolte alcune attività – tra cui, forse, l'esecuzione di O
  - alla fine, per C ci sono due possibilità
    - C riceve un *risultato*
    - C riceve la segnalazione di un'*eccezione remota*
  - a fronte di questi possibili esiti, che cosa può capire C di quanto è effettivamente successo nel sistema?
  - la semantica dell'invocazione remota (*call semantics*) riguarda appunto ciò che può succedere durante un'invocazione remota

Semantica dell'invocazione remota

- I proxy remoti – e in particolare i loro moduli di comunicazione (*mdc*) – possono gestire la perdita dei messaggi di richiesta e di risposta sulla base di tre opzioni (ciascuna opzione richiede anche le precedenti)
  - il proxy lato client ripete il messaggio di richiesta
  - il proxy lato server filtra richieste duplicate
  - il proxy lato server ritrasmette risposte

Semantica dell'invocazione remota

- Queste opzioni danno luogo a un ventaglio di semantiche differenti per l'invocazione di operazioni remote
  - il proxy lato client non ripete richieste (*maybe*)
  - il proxy lato client ripete richieste, il proxy lato server non filtra duplicati e semmai chiede al server di rieseguere l'operazione (*at least once*)
  - il proxy lato client ripete richieste, il proxy lato server filtra duplicati e ritrasmette le risposte (senza far rieseguire l'operazione al server) (*at most once*)

Semantica dell'invocazione remota

- Possibili semantiche per l'invocazione remota
  - che cosa può capire il client se riceve un risultato?
  - e se riceve un'eccezione remota?
  - *maybe* – il proxy lato client non ripete richieste
  - *at least once* – il proxy lato client ripete richieste, il proxy lato server non filtra duplicati e semmai chiede al server di rieseguire l'operazione
  - *at most once* – il proxy lato client ripete richieste, il proxy lato server filtra duplicati e ritrasmette le risposte (senza chiedere al server di rieseguire l'operazione)
- La semantica dell'invocazione "locale" è invece *exactly once*
  - l'operazione viene eseguita esattamente una volta
  - un'invocazione termina sempre con la restituzione di un risultato

Discussione

Semantica	Opzioni	Se il client riceve un risultato, l'operazione è stata eseguita	Se il client riceve un'eccezione remota, l'operazione è stata eseguita	Note
Maybe	il proxy lato client non ripete richieste	esattamente 1 volta	0 o 1 volta	
At least once	il proxy lato client ripete richieste, il proxy lato server non filtra duplicati e chiede al server di riesegue l'operazione	1 o più volte	0 o più volte	
At most once	il proxy lato client ripete richieste, il proxy lato server filtra duplicati e ritrasmette le risposte (senza chiedere al server di rieseguire l'operazione)	esattamente 1 volta	0 o 1 volta	maggiore tolleranza ai guasti rispetto a maybe
Exactly once	invocazione locale	esattamente 1 volta	non è possibile	invocazione locale

Discussione

- Dunque, tre possibili semantiche per l'invocazione remota
  - nessuna corrisponde alla semantica *exactly once* dell'invocazione "locale"
  - differiscono in quello che può accadere durante un'invocazione remota
    - l'operazione remota può essere eseguita dal server zero, una o anche più volte
  - differiscono in quello che il client può comprendere alla fine dell'invocazione remota
    - quando al client viene restituito il controllo, insieme a un risultato oppure a un'eccezione remota

Discussione

- Quando si usa uno specifico servizi di middleware, è importante capire qual è la semantica dell'invocazione remota che implementa, nonché le sue implicazioni – ad esempio
  - se la semantica è *at-least once*, allora l'invocazione di un'operazione può causare delle esecuzioni multiple
    - in alcuni casi (quali?) è opportuno implementare l'operazione in modo idempotente
  - se invece la semantica è di tipo *at-most once* (o *maybe*)
    - l'idempotenza delle operazioni remote non è in genere necessaria (oppure no?)

Discussione

- La tattica della ripetizione dei messaggi di richiesta in assenza di una risposta dal server – usata nelle semantiche *at-least once* e *at-most once* – è sempre efficace?
  - questa tattica consente di gestire l'eventuale perdita dei messaggi di richiesta e di risposta – ma è efficace solo se i problemi di comunicazione sono transienti
  - può consentire di gestire eventuali problemi di indisponibilità del processo server – ma solo se questi problemi sono transienti
  - tuttavia, se questi problemi sono duraturi, questa tattica può addirittura peggiorare la situazione, anziché migliorarla
    - ad es., se il processo server è lento a rispondere per un problema di carico (ci sono molti thread attivi), la ripetizione delle richieste, soprattutto con la semantica *at-least once*, può peggiorare le cose (i thread attivi possono aumentare)
    - potrebbero essere preferibili altre tattiche, per sostenere l'isolamento dei guasti

- Prestazioni

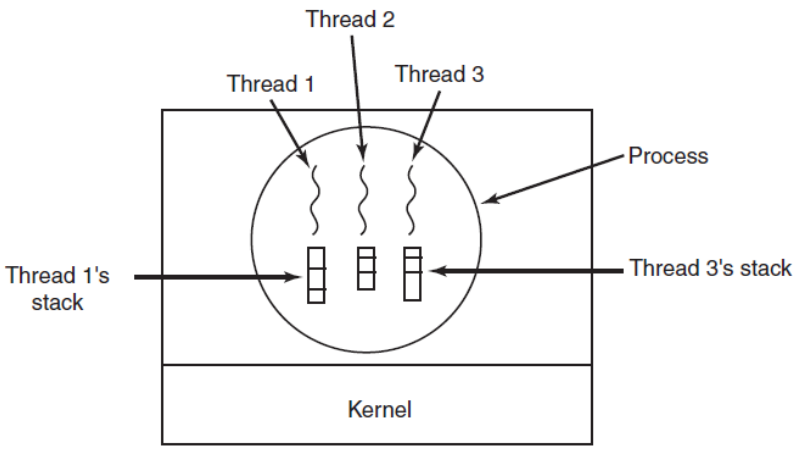
- Invocazioni remote e invocazioni "locali" differiscono anche rispetto alle prestazioni
  - il protocollo richiesta-risposta su cui si basa l'invocazione remota introduce degli overhead (quali?)
  - per minimizzare la penalizzazione sulle prestazioni, è spesso utile minimizzare il numero di invocazioni remote (come?)

- Concorrenza

- Le operazioni remote possono essere eseguite in modo concorrente tra loro
  - un'operazione remota viene di solito eseguita in un pop-up thread distinto del componente remoto
  - l'esecuzione di un'operazione remota viene gestita in uno stack dedicato alla chiamata remota
  - è possibile che più invocazioni remote vengano eseguite in modo concorrente
  - è possibile che più operazioni remote eseguite in modo concorrente operino su risorse condivise

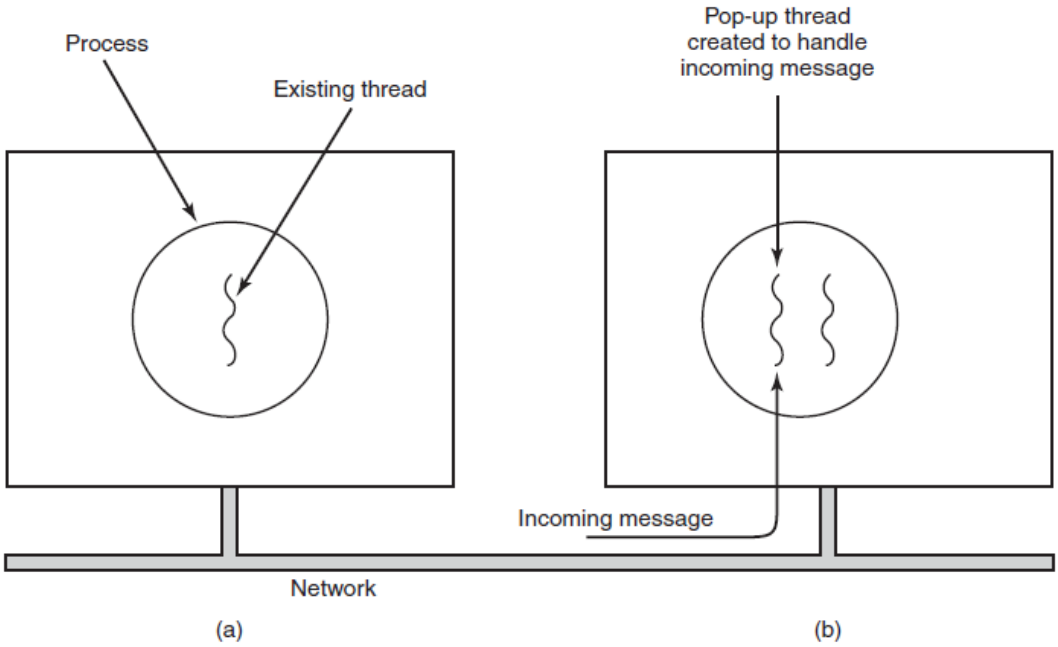
Parentesi: Processi e thread

- Nei sistemi operativi
  - un *processo* è un'istanza di un programma in esecuzione
  - un *thread* è un'entità che può essere assegnata a un processore ed eseguita dal processore
  - ogni processo ha almeno un thread di esecuzione
  - i thread di un processo sono eseguiti in modo concorrente e condividono con il processo le sue risorse



Parentesi: Pop-up thread

- In un processo server, si parla di *pop-up thread* se, ogni volta che il server riceve una richiesta, il processo server crea un nuovo thread (pop-up thread) per gestire quella richiesta



- Legame dei parametri

- Nell'invocazione remota si adotta di solito il legame dei parametri per *valore-risultato*
  - in un'operazione, si fa distinzione tra parametri di input (in) e risultati/parametri di output (out)
  - i parametri di input e output vengono legati in momenti distinti
- Esempio: calcola le radici di un'equazione di secondo grado
  - intestazione dell'operazione (omettendo i tipi)
    - calcolaRadici(in a, in b, in c, out x1, out x2)
  - invocazione dell'operazione
    - calcolaRadici(1, 4, 2, radice1, radice2)
- In molti linguaggi, è disponibile solo in una versione limitata
  - double sqrt(double x)
  - r = sqrt(125)

Legame di dati strutturati e oggetti

- Una differenza importante tra la semantica dell'invocazione remota e dell'invocazione "locale" è nel modo in cui avviene il legame di dati strutturati – come oggetti o record
  - in un'invocazione "locale", viene di solito passato un *riferimento* all'oggetto
    - questo rende possibile il verificarsi di effetti collaterali sugli oggetti passati come parametri
  - in un'invocazione remota, viene invece passata una *rappresentazione serializzata* dello stato dell'oggetto
    - questo disabilita la possibilità di avere effetti collaterali sugli oggetti passati come parametri (o restituiti come risultati)

Serializzazione

- *Serialization*
  - In computer science, serialization is the process of translating a data structure or an object state into a format that can be stored (for example, in a file) or transmitted (for example, across a network connection link) and reconstructed later (possibly in the same or in a different computer environment).
    When the resulting series of bits is reread according to the serialization format, it can be used to create a semantically identical clone of the original object.
    [Wikipedia, 2020]

Legame di dati strutturati e oggetti

- Un parametro che è un oggetto viene passato come una rappresentazione serializzata dell'oggetto – che viene trasmessa e poi ricostruita nel processo remoto
  - **alpha(value-object)**

call alpha(

```
{
    "type" : "asw.model.Employee",
    "id" : "410",
    "firstName" : "John",
    "lastName" : "Smith",
    "dept" :
    {
        "type" : "asw.model.Department",
        "id" : "992",
        "name" : "Engineering"
    }
}
```

)

Legame di riferimenti remoti

- In RMI, è possibile passare anche dei parametri che sono dei "riferimenti remoti" a oggetti remoti
  - **beta(remote-object)**

call beta(*remote reference to remoteObject*)

  - in questo caso, l'operazione remota potrà effettuare a sua volta delle invocazioni remote sull'oggetto remoto passato come parametro

- Discussione

- Le invocazioni remote vengono scritte sintatticamente, nel codice, in modo simile a quelle “locali” – tuttavia, la loro semantica è differente, per vari motivi
  - per la possibilità di fallimenti nella comunicazione remota – e per il modo in cui vengono gestiti
  - per il tempo richiesto dalla gestione di una chiamata
  - per la possibile concorrenza delle chiamate remote
  - per come viene effettuato il legame dei parametri – e per gli effetti collaterali che si possono verificare (o non verificare)
  - per aspetti legati alla sicurezza e alla disponibilità (che non sono stati discussi)

* Ulteriori aspetti e varianti dell’invocazione remota

- Discutiamo ora alcuni ulteriori aspetti e varianti dell’invocazione remota
  - invocazione remota “asincrona”
  - trasparenza dalla locazione

- Invocazione remota “asincrona”

- Alcune varianti dell’invocazione remota – utili soprattutto per operazioni di lunga durata e dunque con una latenza alta
  - *invocazione (remota) asincrona*
    - la chiamata è non bloccante – e restituisce immediatamente un oggetto (*promise* oppure *future*) per il risultato
  - *invocazione (remota) sincrona differita*
    - il chiamante effettua una prima chiamata (non bloccante) per attivare l’operazione remota – e poi una seconda chiamata per accedere al risultato dell’operazione
  - *callback*
    - il chiamante, al momento dell’invocazione remota (asincrona), specifica un’operazione (*callback*) che l’oggetto remoto può invocare per comunicare il risultato

- Trasparenza dalla locazione

- Nell’invocazione remota, il client deve conoscere la locazione in rete del server per l’operazione richiesta – ad es., indirizzo IP (oppure nome di dominio nel DNS) e porta
  - nei casi più semplici, la locazione dei servizi è definita staticamente – nel client potrebbe essere specificata in un file di configurazione
  - le cose sono spesso più complesse – soprattutto nei sistemi software moderni virtualizzati o sul cloud, perché i servizi possono essere replicati e possono cambiare locazione dinamicamente
  - può allora essere utile un’infrastruttura di comunicazione per rendere i client indipendenti dalla locazione fisica dei server – un broker oppure un servizio di service discovery

* Discussione

- L'invocazione remota è un'astrazione di programmazione distribuita fondamentale, basata sull'invocazione di operazioni
  - consente a un componente (nel ruolo di client) di chiamare/invocare un'operazione di un componente remoto (nel ruolo di server), affinché il componente remoto server esegua l'operazione richiesta dal client
  - questo stile di comunicazione è sostenuto dal pattern architetturale *Broker* [POSA]