RPC - *Remote Procedure Call*

Sistemas Distribuídos e Mobile

Prof. Me. Gustavo Torres Custódio
gustavo.custodio@anhembi.br

Conteúdo

Remote Procedure Call (RPC)

RMI - Java

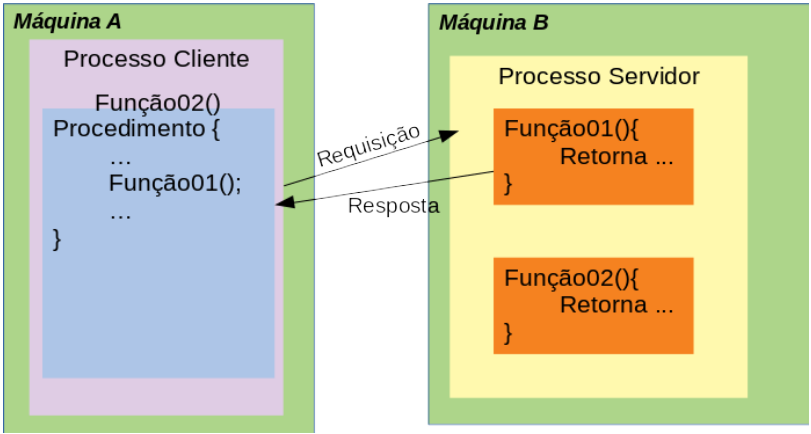
RPC - Remote Procedure Call

Remote Procedure Call (RPC)

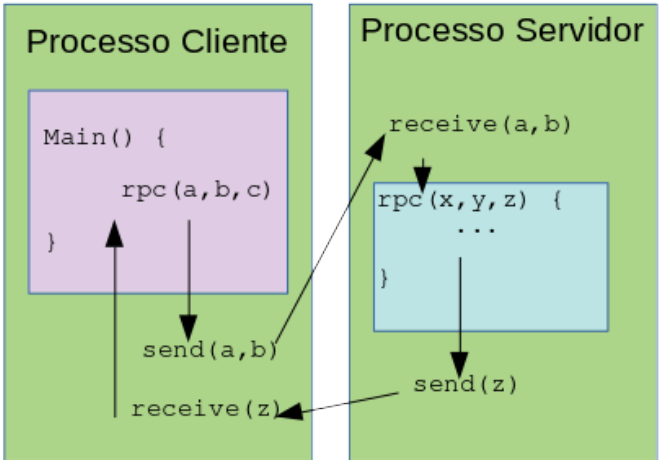
RPC

- *Remote Procedure Call* (RPC) consiste em chamar um procedimento disponível em outra máquina.
  - Os parâmetros necessários para executar o procedimento são passados junto com a chamada remota.
  - Retorno das informações para o processo “chamador”.
  - Transparência - do ponto de vista do usuário, é como se um método normal fosse chamado.

RPC


Máquina A
Processo Cliente
Função02()
Procedimento {
…
Função01();
…
}
Requisição
Resposta
Máquina B
Processo Servidor
Função01(){
Retorna ...
}
Função02(){
Retorna ...
}

RPC


Processo Cliente
Main() {
rpc(a,b,c)
}
send(a,b)
receive(z)
Processo Servidor
receive(a,b)
rpc(x,y,z) {
...
}
send(z)

Problemas

- Espaços de endereçamento diferentes (máquinas diferentes);
- Arquitetura diferentes, interferindo nos tipos de dados;
- Problemas no servidor;
- Problemas no cliente.

Chamada de Procedimento Convencional

- A passagem de parâmetros é feita **por valor** ou **por referência**.
- Por valor:
  - O parâmetro é copiado para a pilha de execução.
  - O procedimento chamado pode afetar o valor do parâmetro, mas não afeta a variável original.

Chamada de Procedimento Convencional

- Por Referência:
  - O parâmetro é um ponteiro para a variável
  - Se o parâmetro tem seu valor alterado, isso é refletido no processo chamador.

Chamada de Procedimento Remoto

- Para o SO (kernel), o envio e recebimento de mensagens é transparente, ou seja, ele não sabe que está fazendo uma RPC.
  - Transparência.
  - Portanto, deve existir um processo específico para o tratamento de RPCs.
  - Isso é feito por meio de *stubs*.
    - Basicamente possuem a função de converter chamadas remotas para locais.

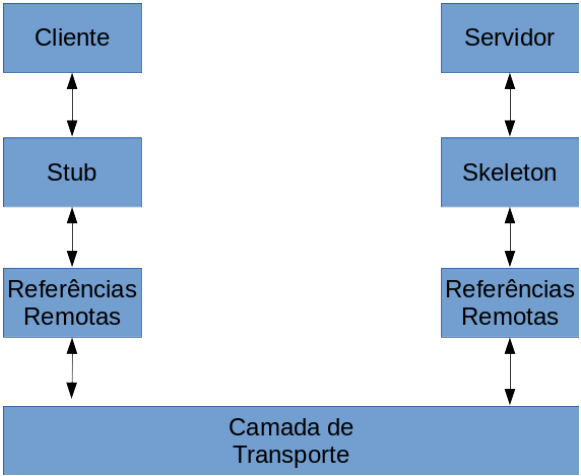
Chamada de Procedimento Remoto

- *Stub do Cliente*:
  - Empacota os parâmetros em uma mensagem e a envia para a máquina do servidor.
  - O cliente bloqueia a si mesmo em *receive* até que a resposta do servidor chegue (síncrono).

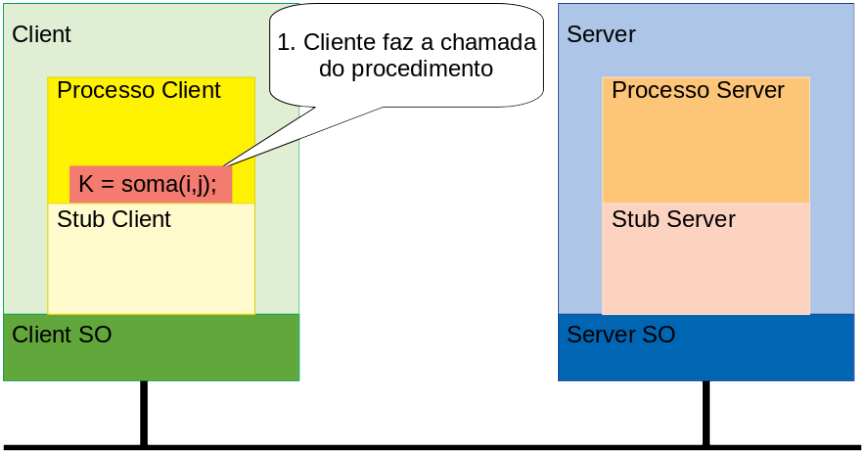
Chamada de Procedimento Remoto

- *Stub do Servidor*:
  - Quando a mensagem chega ao servidor, o SO dele passa a mensagem para o *stub* do servidor.
  - Desempacota os parâmetros da mensagem do cliente e executa o procedimento no servidor.
  - Envia mensagem de volta ao cliente por meio da instrução *send*.

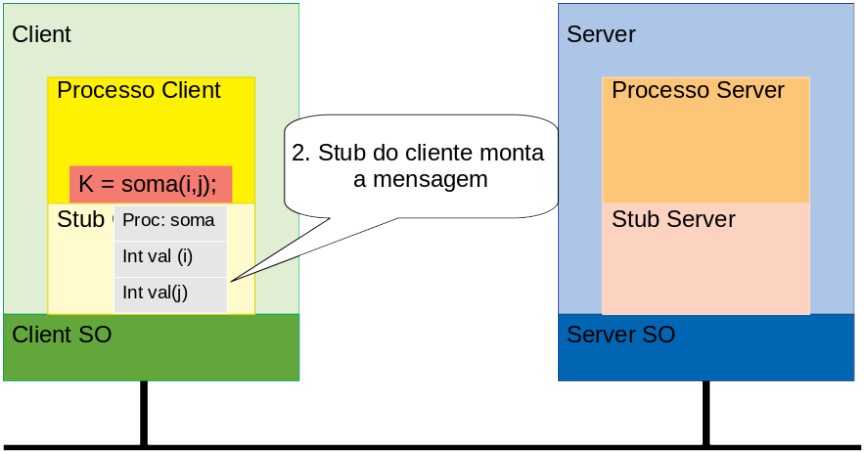
Chamada de Procedimento Remoto



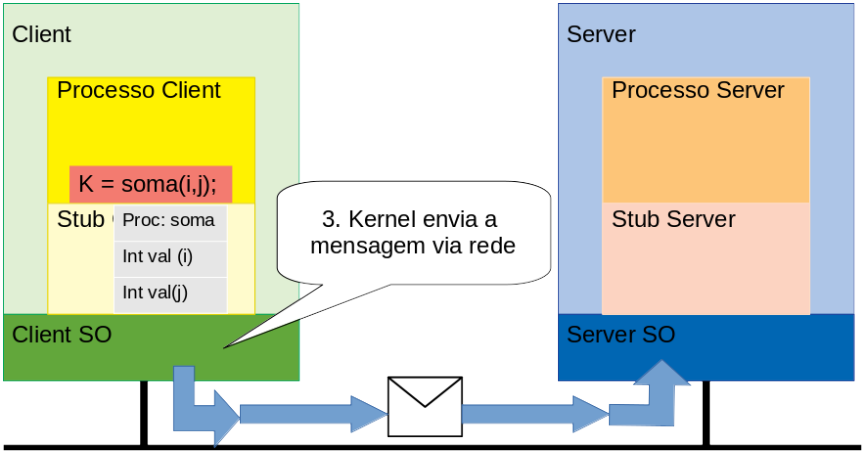
Chamada de Procedimento Remoto



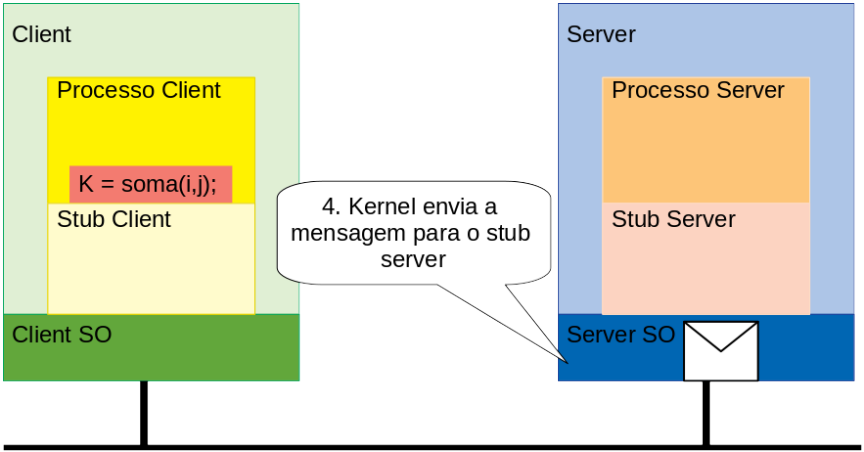
Chamada de Procedimento Remoto



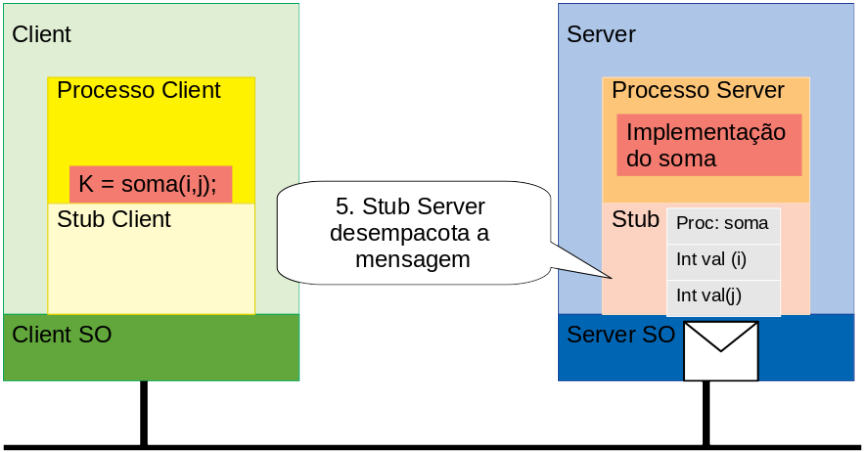
Chamada de Procedimento Remoto



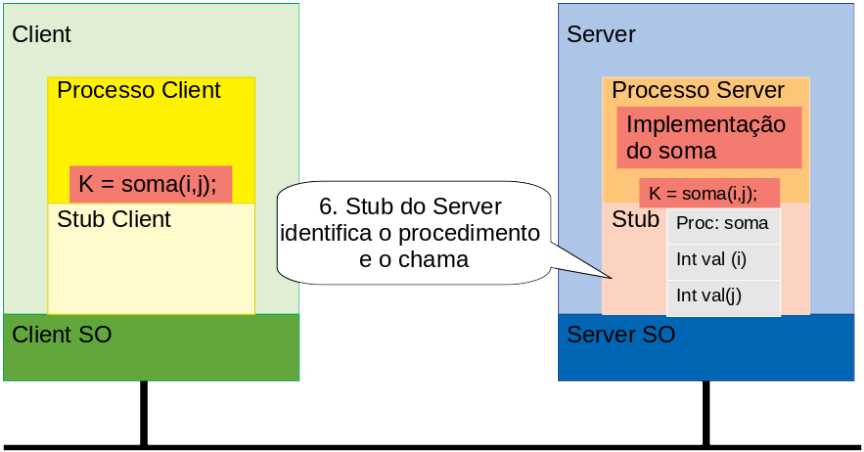
Chamada de Procedimento Remoto



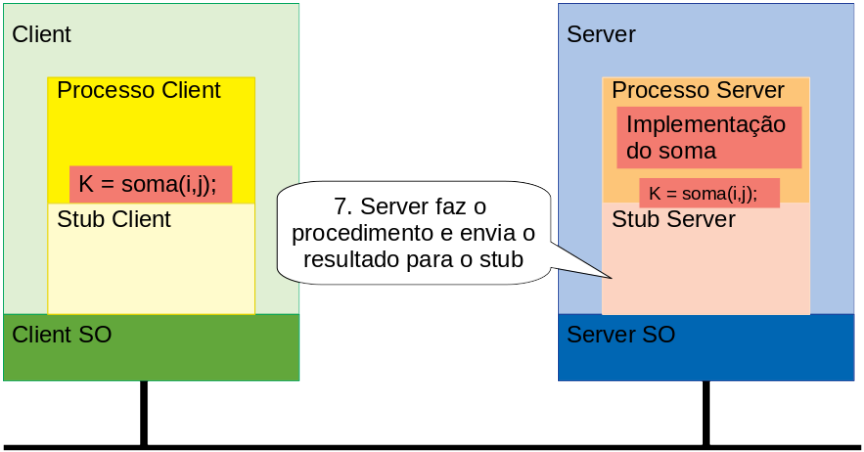
Chamada de Procedimento Remoto



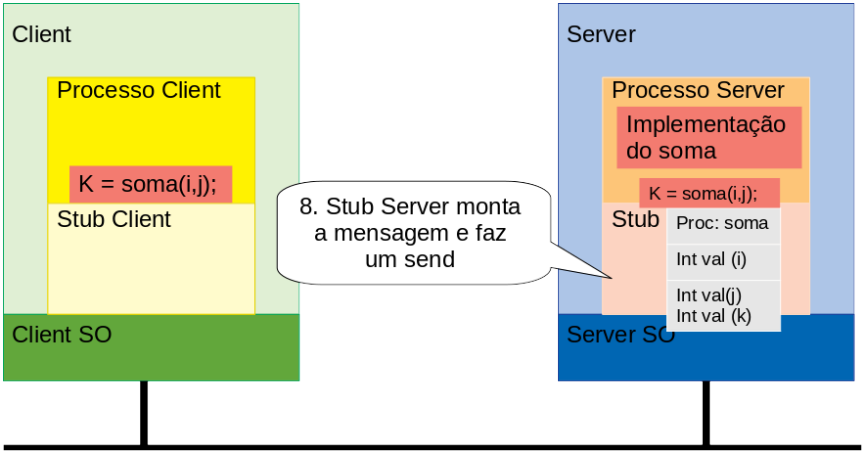
Chamada de Procedimento Remoto



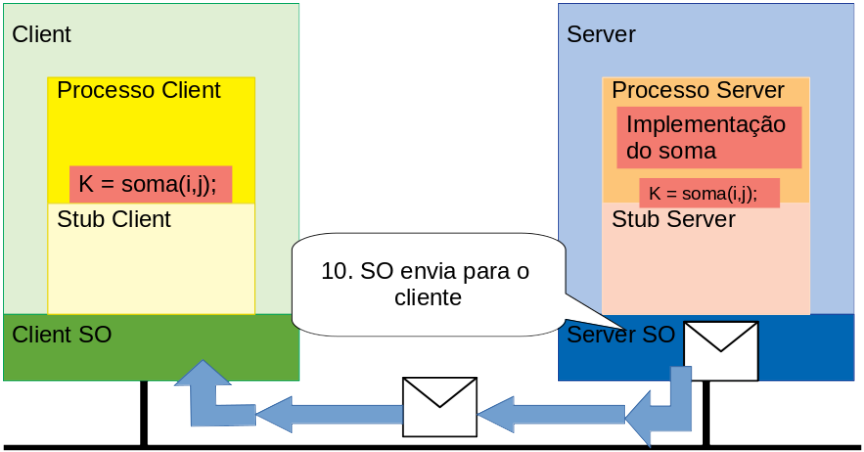
Chamada de Procedimento Remoto



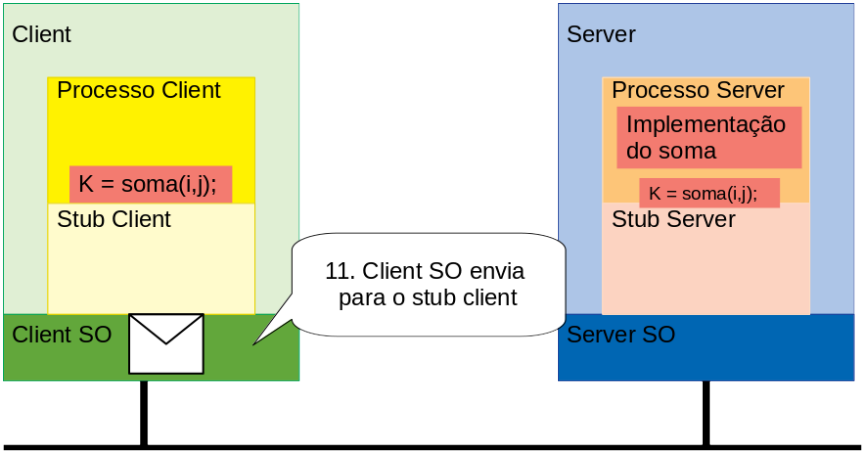
Chamada de Procedimento Remoto



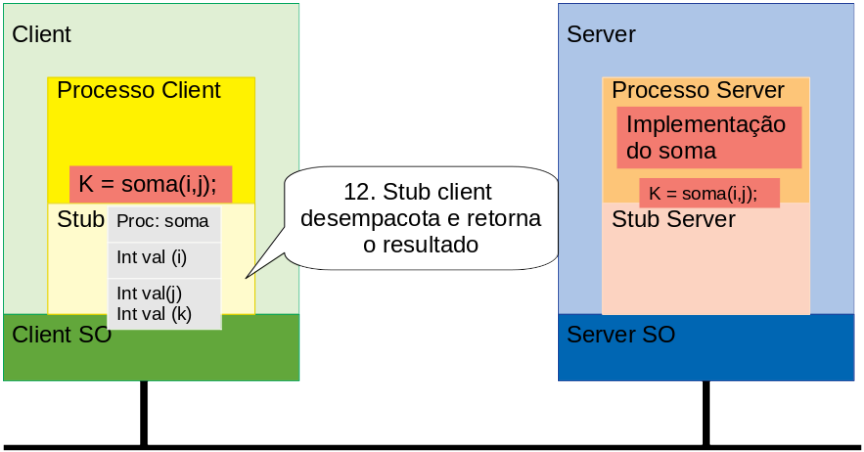
Chamada de Procedimento Remoto



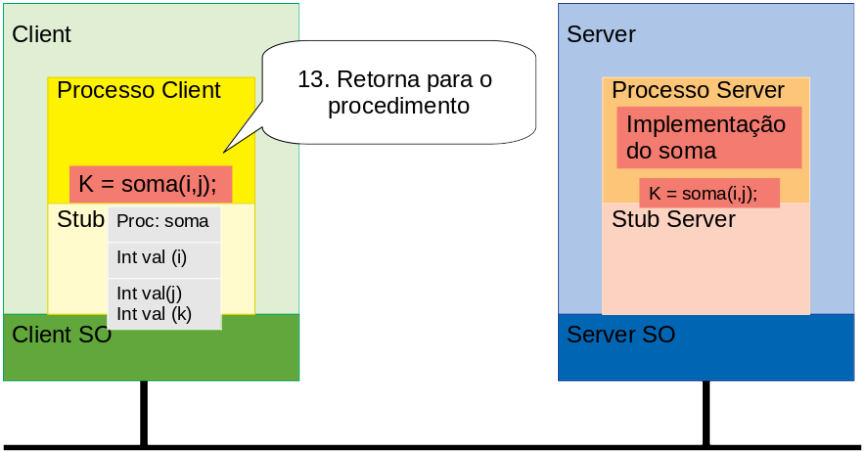
Chamada de Procedimento Remoto



Chamada de Procedimento Remoto



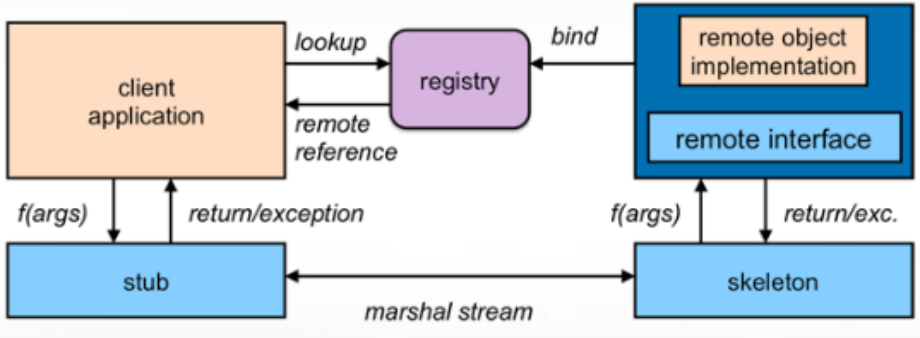
Chamada de Procedimento Remoto



Binding Dinâmico - RPC

- Para o cliente pedir para executar um método remoto, ele precisa saber a localização do servidor.
  - Uma alternativa é armazenar o endereço do servidor no programa (inflexível).
  - Outra alternativa é mapear cada serviço do servidor a uma porta específica de forma dinâmica:
    - *binding* dinâmico.

Binding Dinâmico - RPC



Binding Dinâmico - RPC

- O *binder* funciona como um *nameserver* para o RPC.
- É necessário o cliente enviar uma mensagem para o *binder* para localizar o servidor
  - *Overhead*.
- É possível o uso de múltiplos *binders*.

Binding Dinâmico - RPC

- Servidor
  - Ao ser executado, avisa ao programa de registro (*binder*) que está ativo: registro do serviço.
- Cliente
  - Quando um procedimento é chamado, este se liga ao servidor, utilizando o *binder* como intermediário.

Binding Dinâmico - RPC

- Vantagens:
  - Diversos servidores com a mesma “interface”;
  - Servidores que falham são automaticamente “desregistrados”;
  - Autenticação de usuários/clientes.
- Desvantagens:
  - *Overhead* de consulta ao *binder*;
  - Caso haja diversos *binders*, há a necessidade de atualização entre eles.

Falhas - RPC

- A concepção do RPC é deixar a programação transparente, mas as seguintes falhas podem ocorrer:
  - Cliente não acha o servidor;
  - A mensagem do cliente para o servidor foi perdida;
  - A mensagem do servidor para o cliente foi perdida;
  - O servidor sai do ar após receber uma solicitação;
  - O cliente sai do ar após ter enviado uma solicitação.

Falhas - RPC

- Cliente não acha o servidor:
  - O servidor está fora do ar;
  - Versões diferentes de stubs;
  - Soluções:
    - Retornos de variáveis “inválidas”: e.g. -1
    - Criação de exceções (perda de transparência).

Falhas - RPC

- A mensagem do cliente para o servidor foi perdida:
  - Limite de tempo de espera (*timeout*);
  - Reenvio em kernel;
  - Retorno de erro, após diversas tentativas.

Falhas - RPC

- O servidor sai do ar após receber uma solicitação:
  - Espera e reenvia / ache novo servidor;
  - Desiste e comunique falha.
    - As falhas até o momento não são distinguíveis para um cliente.

Falhas - RPC

- O cliente sai do ar após ter enviado uma solicitação:
  - Processamento órfão;
  - Gasto de tempo do servidor;
  - Soluções:
    - Reencarnação;
    - Extermínio;
    - Expiração (quantum T).

Falhas - RPC

- Extermínio:
  - Eliminar todos os órfãos.
  - Problema: encadeamento de falhas (servidor pode ter sido cliente em uma RPC).

Falhas - RPC

- Reencarnação:
  - Dividir o tempo em “épocas”.
  - Nova chamada → nova época.
  - Quando o cliente se recuperar, envia uma mensagem de broadcast inaugurando nova época.

Falhas - RPC

- Expiração:
  - Tempo máximo para servidor executar serviço.
  - Problema: mensurar o quantum.

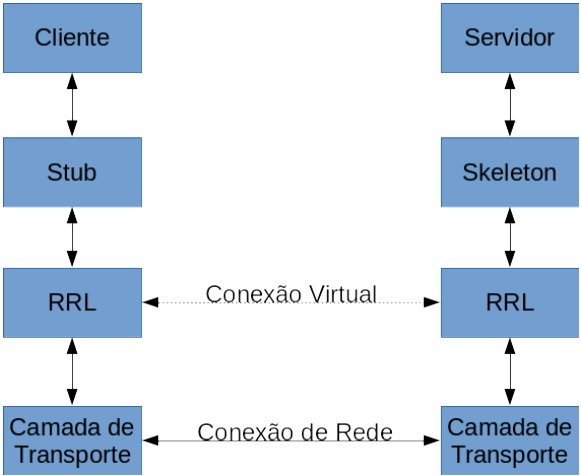
RPC - Remote Procedure Call

RMI - Java

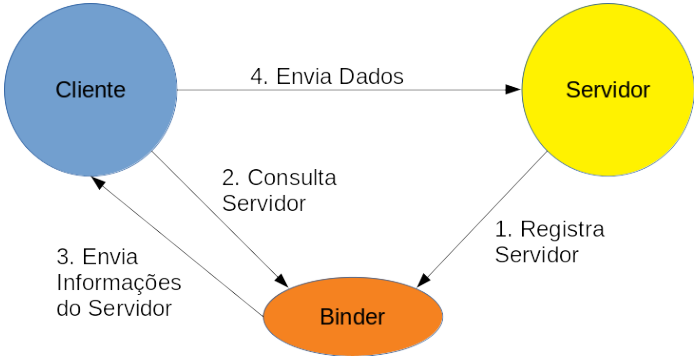
RMI - Java

- Remote Method Invocation
  - RPC da linguagem Java;
  - Orientação a Objetos;
  - 3 camadas:
    - *Stub/Skeleton*: é invocado primeiro, recebendo objetos e serializando-os (bytes).
    - RRL – *Remote Reference Layer*: cada lado possui a sua, responsável por montar a mensagem de envio/resposta entre cliente/servidor.
    - Transporte: envio das informações via rede – TCP/IP.

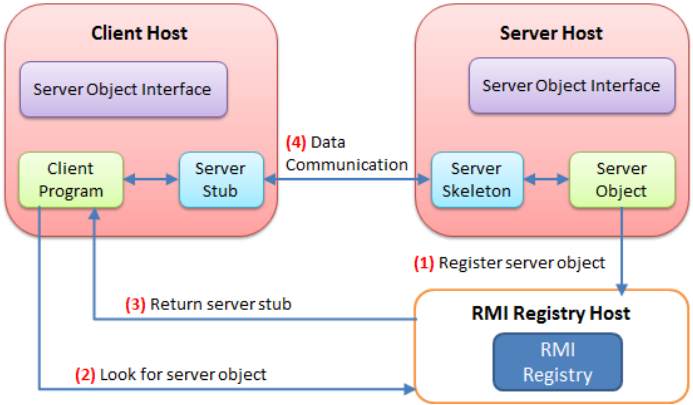
RMI - Java


Cliente
Servidor
Stub
Skeleton
RRL
Conexão Virtual
RRL
Camada de Transporte
Conexão de Rede
Camada de Transporte

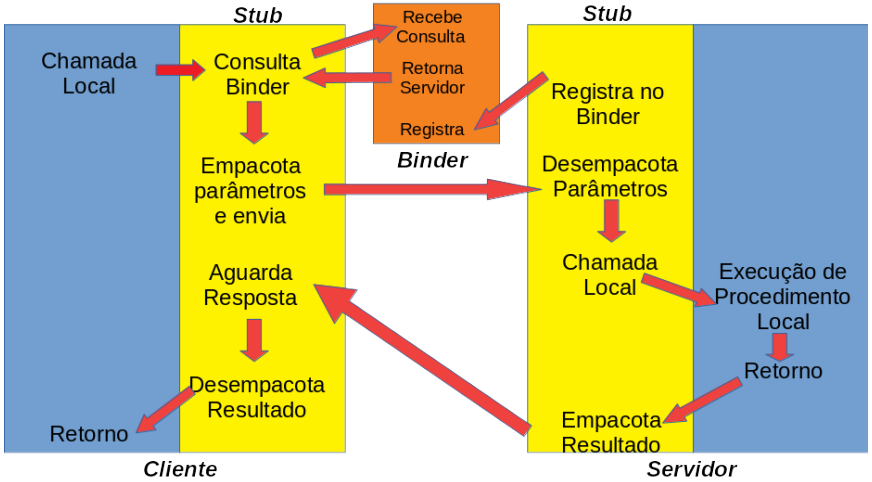
RMI - Java


Cliente
4. Envia Dados
Servidor
2. Consulta
Servidor
1. Registra
Servidor
3. Envia
Informações
do Servidor
Binder

RMI - Java


Client Host
Server Object Interface
Client Program
Server Stub
(4) Data Communication
Server Host
Server Object Interface
Server Skeleton
Server Object
(1) Register server object
RMI Registry Host
RMI Registry
(3) Return server stub
(2) Look for server object

RMI - Java


Stub
Recebe Consulta
Retorna Servidor
Registra
Stub
Chamada Local
Consulta Binder
Registra no Binder
Binder
Empacota parâmetros e envia
Desempacota Parâmetros
Aguarda Resposta
Chamada Local
Execução de Procedimento Local
Retorno
Desempacota Resultado
Retorno
Empacota Resultado
Cliente
Servidor

RMI

- Utilização de Objetos Java de forma “transparente”;
  - Uso de classe tipificada para a definição dos métodos:
    - `interface`.
  - A classe do cliente atua como proxy;
  - A implementação da classe fica no servidor.

Tutorial Java - RMI

- Servidor
  - Interface que contém os métodos a serem disponibilizados para os clientes.
- implementa interface `Remote`
  - Classe que contém a lógica dos métodos, além das rotinas de registro.
- Herda características de UnicastRemoteObject
- Implementa a interface definida
- Método `main()` para registrar o serviço

Exemplo Servidor - RMI

- Aplicação que possui um servidor que disponibiliza um método `soma()` para os clientes.
  - Servidor se auto-registra no *binder*.
  - Cliente procura pelo objeto que possui esta funcionalidade através de seu nome.
  - Cliente executa o método correspondente.

Exemplo Servidor - RMI

- Baixe o arquivo:
  - `codigos_rpc.zip`

Exercícios

- Altere o Exemplo da calculadora para fazer as operações de soma, subtração, multiplicação e divisão, de acordo com a escolha do usuário.

Referências

- Livro:
  - Sistemas Distribuídos: Princípios e Paradigmas
    - Tanenbaum
    - Cap 10 - Sistemas distribuídos baseados em objetos
- Livro:
  - Sistemas Distribuídos: Conceitos e Projetos
    - Coulouris
    - Cap 5 - Invocação Remota

Conteúdo

https://gustavotcustodio.github.io/sdmobile.html

Obrigado

gustavo.custodio@anhembi.br