RPC e RMI

Programação Objetos distribuidos

Objetivos

- Nesta aula iremos apresentar o primeiro modelo de tecnologia de desenvolvimento em sistemas distribuídos, a invocação remota. Serão discutidos o modelo primitivo de requisição/resposta, RPC e o RMI/Java. Os conceitos são fundamentais para o entendimento do desenvolvimento de aplicações em ambientes distribuídos.

Plano de Aula

- RPC e RMI
  - Introdução
  - Chamada de Procedimento Remoto
  - Invocação a Método Remoto

RPC e RMI

- **Introdução**

O **RPC** representa um importante avanço intelectual na computação distribuída, com o objetivo de tornar a programação de sistemas distribuídos semelhante à programação convencional;

- O **RMI** está intimamente relacionada à RPC, mas é estendida para o mundo dos objetos remotos distribuídos;
- Aqui iremos apresentar de forma genérica como cada um destes exemplos de **invocação remota** funcionam;

RPC e RMI

- **Chamada de Procedimento Remoto (RPC)**

```
main(){
        char nome[5];
        int idade = 31;

        printf("Informe seu nome:");
        gets(nome);
        printf("Informe sua idade:");
        scanf("%d",&idade);
        fontedaJuventude(idade);
        printf("Nome:%s\n",nome);
        printf("Idade:%d\n",idade);
        envelhecer(&idade);
        printf("Nome:%s\n",nome);
        printf("Idade:%d\n",idade);
        system("pause");
}
void fontedaJuventude(int id){
      id = id + 1;
}
void envelhecer(int *id){
      *id = *id + 1;
}
```

RPC e RMI

- **Chamada de Procedimento Remoto (RPC)**
  - Antes de examinarmos a implementação em sistemas de RPC, vamos ver três questões importantes para entender esse conceito:
    - O estilo de programação promovido pela RPC: programação com interfaces;
    - A semântica de chamada associada à RPC;
    - O problema da transparência e como ele se relaciona com as chamadas de procedimento remoto;

RPC e RMI

- **Chamada de Procedimento Remoto (RPC)**
  - **Programação com Interfaces**
    - As modernas linguagem de programação promovem a **modularização** e **comunicação entre estes módulos**;
    - Para controlar as possíveis interações entre os módulos, é definida uma ***interface*** que especifica os **procedimentos** e as **variáveis** que podem ser acessadas a partir de **outros módulos**;
    - Em um modelo **cliente-servidor** geralmente surge o termo, ***interface de serviço***: se refere a especificação dos procedimento oferecidos por um servidor, definindo os tipos de argumentos de cada um dos procedimentos;
    - Uso de Linguagens de Definição de Interfaces: As **IDLs** são projetadas para permitir que procedimentos implementados em diferentes linguagens invoquem uns aos outros;

RPC e RMI

- **Chamada de Procedimento Remoto (RPC)**
  - **Semântica de chamada RPC**
    - Promove a confiabilidade das invocações remotas vistas pelo ativador/chamador;
    - **Semântica talvez**: a chamada de **rpc** pode ser executada uma vez ou não ser executada. Ele surge quando nenhuma medida de tolerância a falhas é aplicada;
    - **Semântica pelo menos uma vez**: o invocador recebe um resultado (o procedimento foi executado pelo menos uma vez) ou recebe uma exceção;
    - **Semântica no máximo uma vez**: o chamador recebe um resultado – no caso em que o chamador tem certeza da execução – ou recebe uma exceção informando-o de que nenhum resultado foi recebido.

RPC e RMI

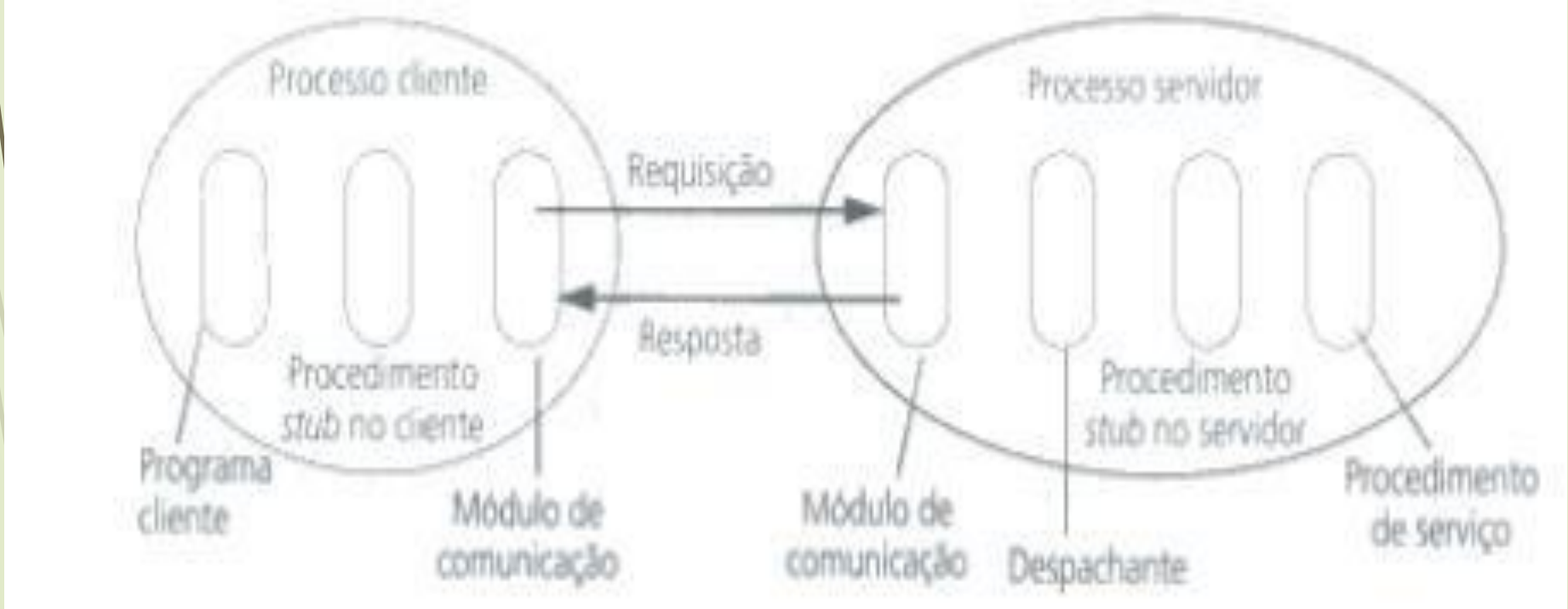
- **Chamada de Procedimento Remoto (RPC)**
  - **Semântica de chamada RPC**

Medidas de tolerância a falhas			Semântica de chamada
Reenvio da mensagem de requisição	Filtragem de duplicatas	Reexecução de procedimento ou retransmissão da resposta	
Não	Não aplicável	Não aplicável	*Talvez*
Sim	Não	Executa o procedimento novamente	*Pelo menos uma vez*
Sim	Sim	Retransmite a resposta	*No máximo uma vez*

- **Chamada de Procedimento Remoto (RPC)**
  - **Transparência**
    - Objetivo Básico do RPC:Chamada Local = Chamada Remota;
    - Uma séria de ações transparentes ao Programador (empacotar/desempacotar, trocas de mensagens, etc);
    - Transparência de localização e de acesso, ocultando o local físico do procedimento;
    - **Perigoso**: Chamadas de Procedimentos Remotos são mais vulneráveis a falhas (envolvem rede, outro computador, outro processo);
    - Consenso atual: Chamadas locais e remotas devem se tornar **transparente** em sintaxe de uma chamada local ser igual a uma remota, mas a **diferença** entre as duas devem ser expressas em suas **interfaces**;

RPC e RMI

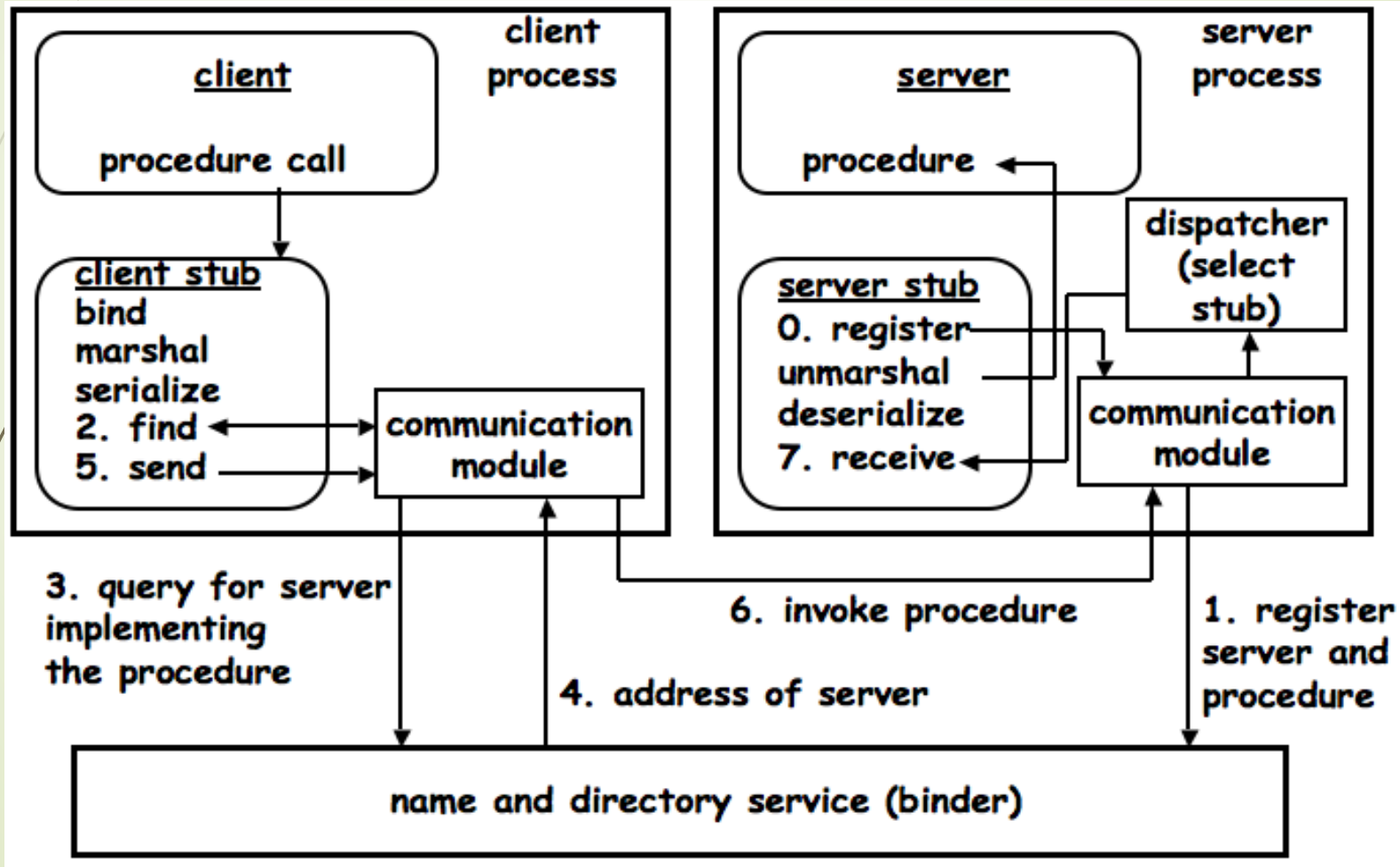
- **Chamada de Procedimento Remoto (RPC)**
  - **Implementação**



- **Chamada de Procedimento Remoto (RPC)**
  - **Implementação**
    - **Cliente** mantém um ***procedimento stub*** para cada procedimento da interface de serviço;
    - O ***procedimento stub*** se comporta como local para o cliente;
    - Na realidade o ***procedimento stub***: empacota o identificador de procedimento e os argumento e os envia através do **módulo de comunicação**;
    - O **Servidor** mantém um **despachante** que seleciona um dos ***procedimentos stub*** do servidor (identificador de procedimento);
    - O ***procedimento stub*** no servidor desempacota os argumentos da mensagem de requisição e chama o **procedimento de serviço**;
    - Após a execução, os **valores de retorno** são empacotados e enviados ao cliente;
    - Os **procedimentos de serviço** implementam os procedimentos da **interface de serviço**.

RPC e RMI

- Chamada de Procedimento Remoto (RPC)
  - Implementação



RPC e RMI

- **Invocação a Método Remoto (RMI)**
  - No **RMI** um objeto **chamador** pode invocar um método em um objeto potencialmente **remoto**;
  - Semelhanças com RPC:
    - Suporta a programação com interfaces;
    - Construída sobre protocolos de requisição-resposta;
    - Oferecem o mesmo nível de transparência (as interfaces expõe a natureza e complexidade distribuída do RMI);
  - Diferenças positivas
    - Usa o poder da OO (objetos, classes, heranças, etc);
    - Complementa o conceito de ID de um objeto para referências exclusivas, podendo os objetos serem usados como parâmetros;

RPC e RMI

- **Invocação a Método Remoto (RMI)**

- Vamos relembrar a Orientação a Objetos
- **Modelo de Objeto** através da invocação de métodos;
  - Objeto ->conjunto de dados e métodos;
  - Comunicação -> um objeto se comunica com outro
  - Padrão forte ->encapsulamento;
- **Referências de Objeto**
  - Objetos são manipulados através das suas referências;
  - Uma variável não contém um objeto e sim uma referência;
  - Um referência pode ser atribuído a variáveis, passado como argumentos e retornados como resultados;

• Invocação a Método Remoto (RMI)

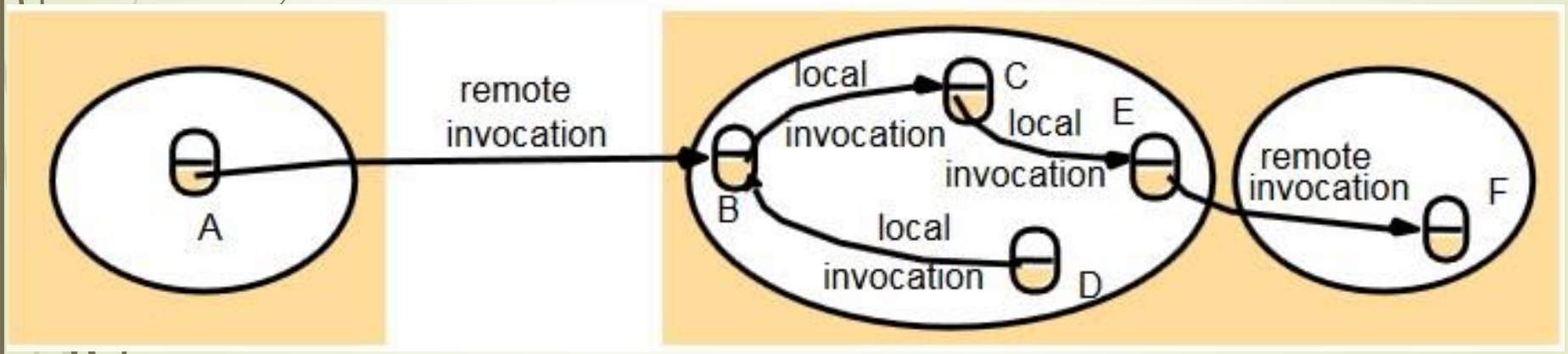
- **Interfaces**
  - Definição da assinatura de um conjunto de métodos;
  - Tipos de argumentos, valores de retorno e exceções;
- **Ações**
  - Uma ação é iniciada por um objeto invocando um método em outro objeto;
  - Uma invocação pode levar a mais invocações a métodos em outros objetos;
- **Exceções**
  - As exceções proporcionam uma maneira clara de tratar com condições de erro, sem complicar o código;
- **Coleta de lixo**

- **Invocação a Método Remoto (RMI)**
  - **Objetos Distribuídos**
    - Os Sistemas de Objetos Distribuídos podem adotar o modelo cliente-servidor;
    - Os objetos são gerenciados pelo servidor e os clientes invocam seus métodos usando RMI;
    - Podemos ter invocações encadeadas (objetos servidores podem se tornar clientes de objetos em outros servidores);
    - O acesso direto aos estados de um objeto é proibitivo (encapsulamento);

- **Invocação a Método Remoto (RMI)**

– Modelo de Objetos Distribuídos

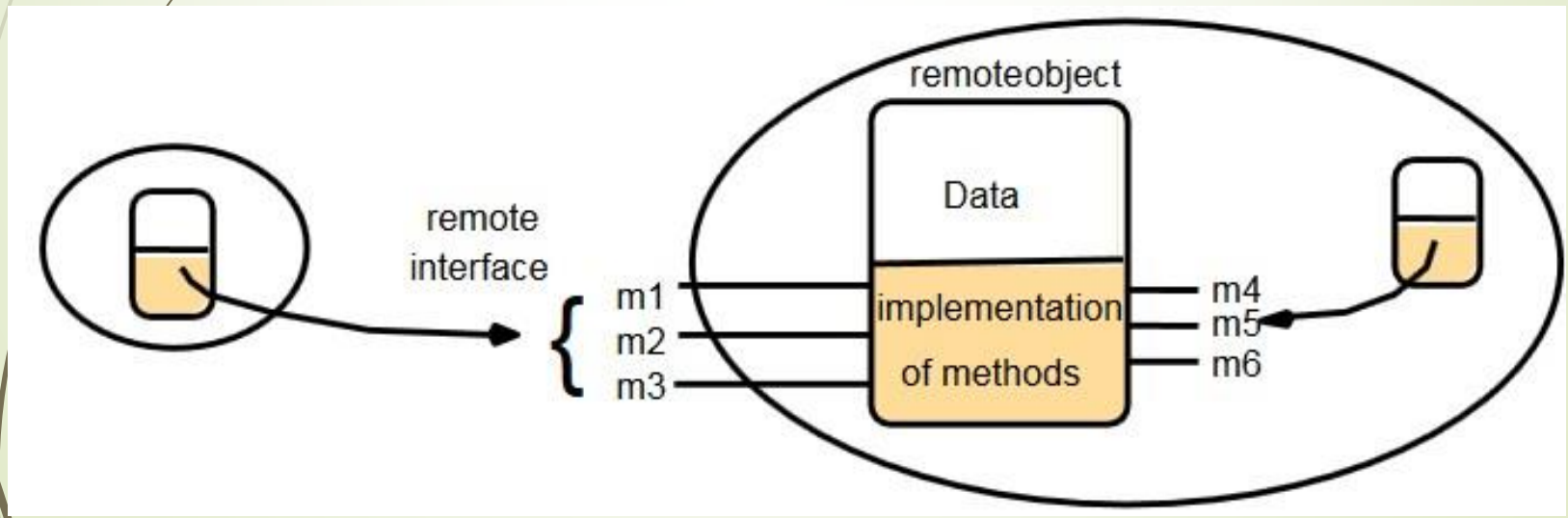
– As invocações a métodos entre objetos em diferentes processos, sejam no mesmo computador ou não, são conhecidas como ***invocações a métodos remotos***. Quando ocorrem em um mesmo processo, chamamos de ***invocações a métodos locais***.



• Invocação a Método Remoto (RMI)

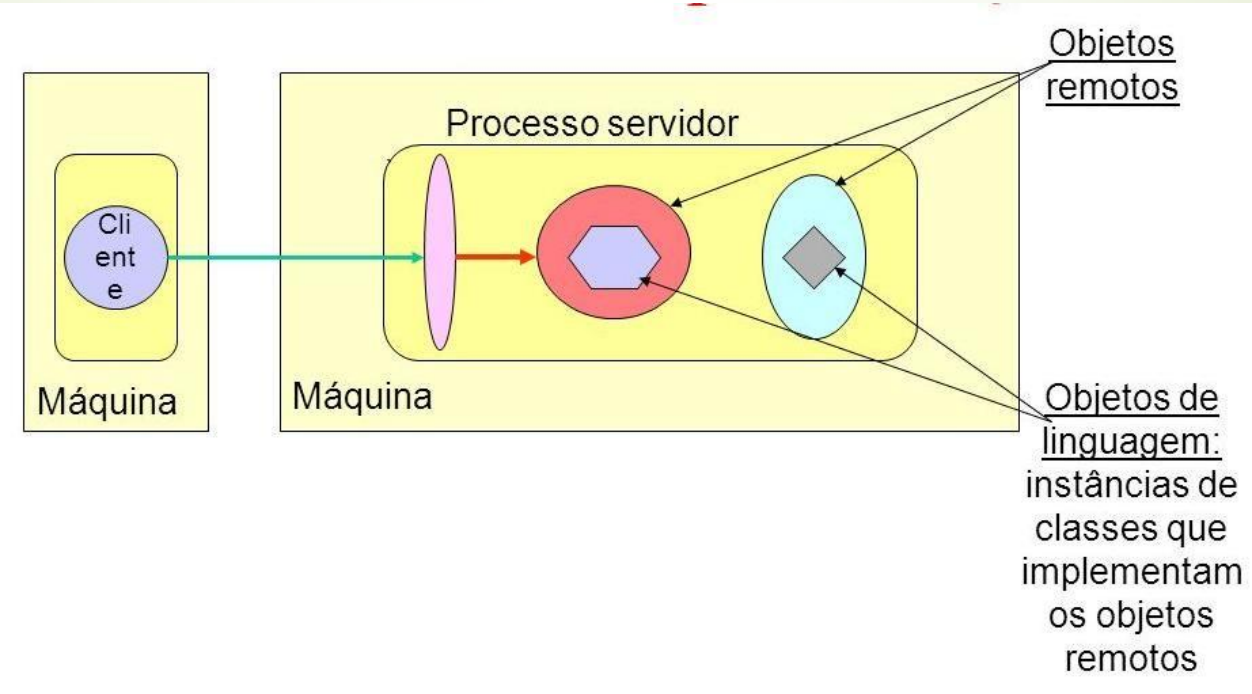
- Interfaces Remotas

- A classe de um objeto remoto implementa os métodos de sua interface remota. Objetos em outros processos só podem invocar os métodos pertencentes à interface remota. Os objetos locais podem invocar qualquer método (local ou remoto).



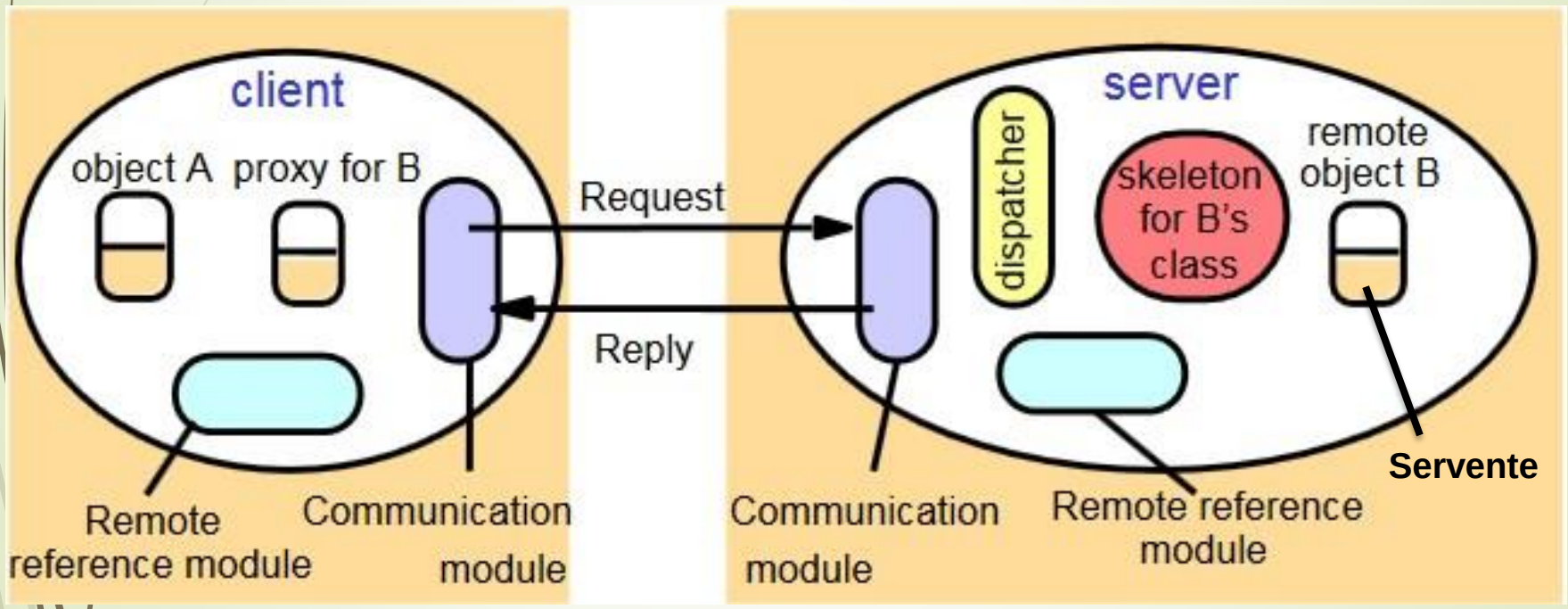
RPC e RMI

- **Invocação a Método Remoto (RMI)**
  - **Ações em um Sistema de Objetos Distribuídos**
    - As ações ocorrem como no modelo não distribuído: através da invocação a métodos. Porém, num sistema distribuído os objetos podem estar em diferentes processos ou computadores.



RPC e RMI

- **Invocação a Método Remoto (RMI)**
  - **Implementação**



RPC e RMI

- **Invocação a Método Remoto (RMI)**
  - **Implementação**
    - O **Módulo de Comunicação** transmite mensagens de requisição-resposta, entre clientes e servidores (mensagem: tipo de mensagem, sua requestID e a referência remota);
    - O **Módulo de Referência Remota** promove a transformação entre referências de objeto local e remoto e pela criação de referência de objeto remoto (uso da tabela de objetos remotos);
    - A **tabela de objetos remotos** inclui:
      - Uma entrada para todos os objetos remotos mantidos pelo processo. O Objeto Remoto B será registrado no servidor;
      - Uma entrada para cada proxy local. O proxy de B será registrado na tabela do cliente;
    - O **Servente** é uma instância de uma classe que fornece o corpo de um objeto remoto;

RPC e RMI

- **Invocação a Método Remoto (RMI)**
  - **Implementação**
    - O **Software RMI** é uma camada de software – *middleware* – entre os objetos e os módulos de comunicação e de referência remota. As funções dos seus componentes são:
    - **Proxy**: comporta-se como um objeto local para o invocador. Ele não executa uma invocação local, ele a encaminha em uma mensagem para um objeto remoto. Ele oculta os detalhes de referência de objeto remoto, do empacotamento dos argumento e desempacotamento dos resultados;
    - **Despachante**: recebe a mensagem de requisição do módulo de comunicação e utiliza o *OperationID* para selecionar o método apropriado no esqueleto;
    - **Esqueleto**: Um método de esqueleto desempacota os argumentos da mensagem de requisição e invoca o método correspondente no **Servente**;