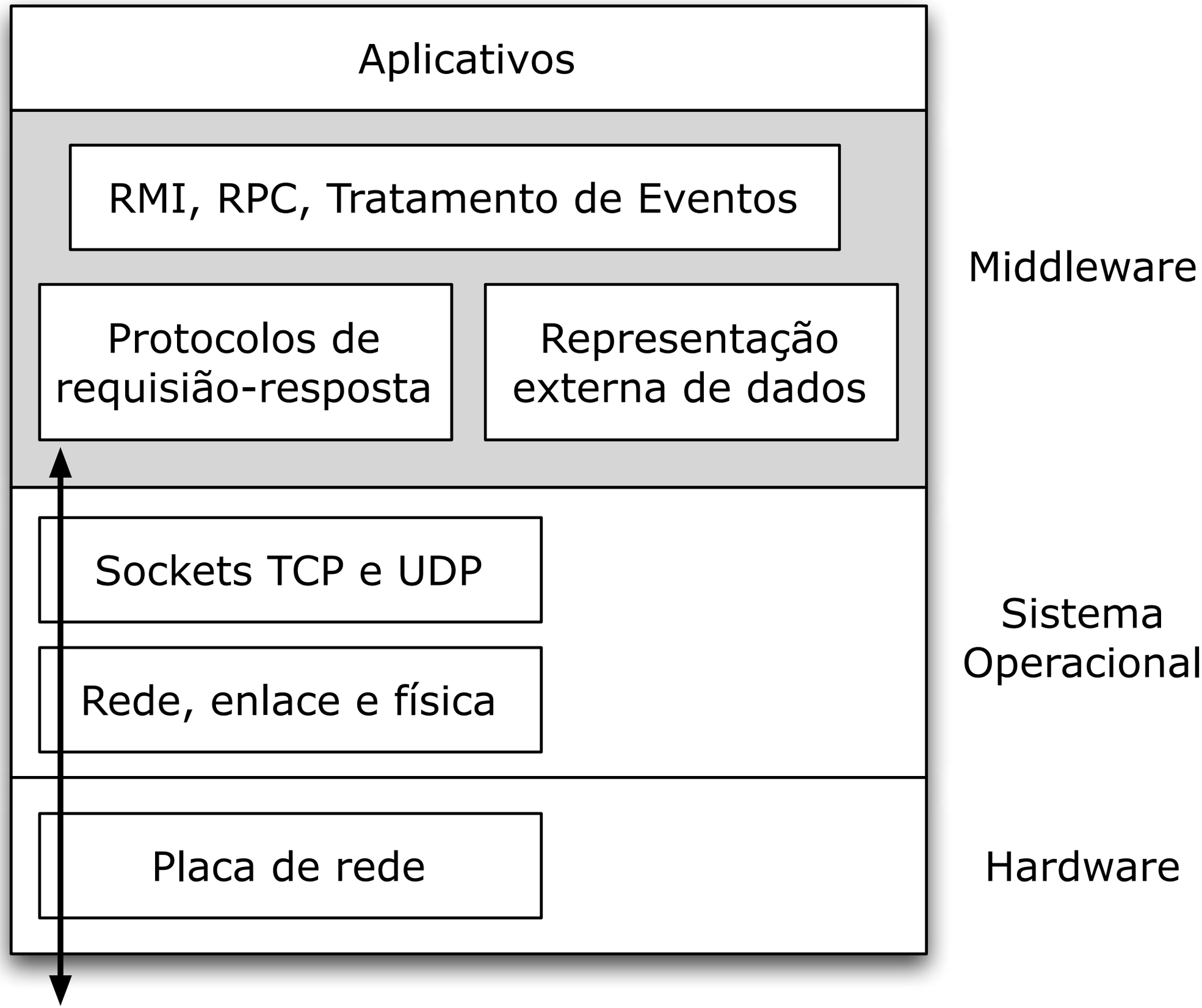
Objetos Distribuídos e Invocação Remota

Middleware

- Modelos de programação para aplicações distribuídas, em geral, são implementadas em um middleware.
  - transparência de localização;
  - independência dos protocolos de comunicação, sistema operacionais e hardware.


Aplicativos
RMI, RPC, Tratamento de Eventos
Protocolos de requisição-resposta
Representação externa de dados
Sockets TCP e UDP
Rede, enlace e física
Placa de rede
Middleware
Sistema Operacional
Hardware

Interfaces

- O uso de interfaces nos permite controlar as interações entre módulos de um sistema.
- A interface de um módulo especifica, de forma precisa e explicita, suas funcionalidades que podem ser acessadas a partir de outro módulo.

Interfaces em SD

- Não devem especificar acesso a dados (somente a serviços)
- Precisam identificar se os parâmetros são de entrada ou saída.
  - parâmetros de entrada são passados para o módulo remoto pelo envio dos valores dos argumentos na mensagem de requisição (parâmetros de saída são retornados na mensagem de resposta)
  - pode-se usar passagem por referência?

Notação

- Interfaces de Serviço
  - especifica os procedimentos disponíveis em um servidor
- Interfaces Remotas
  - especifica os métodos de um objeto que estão disponíveis para acesso remoto

Linguagens de Definição de Interfaces

- Uma IDL (*interface description language*) é uma linguagem para especificação de interfaces
  - permite que programas implementados em linguagens diferentes invoquem procedimentos/ métodos uns dos outros.
- Exemplos
  - IDL para RMI: IDL do CORBA, Java RMI
  - IDL para RPC: XDR da Sun, WSDL

Objetos Distribuídos

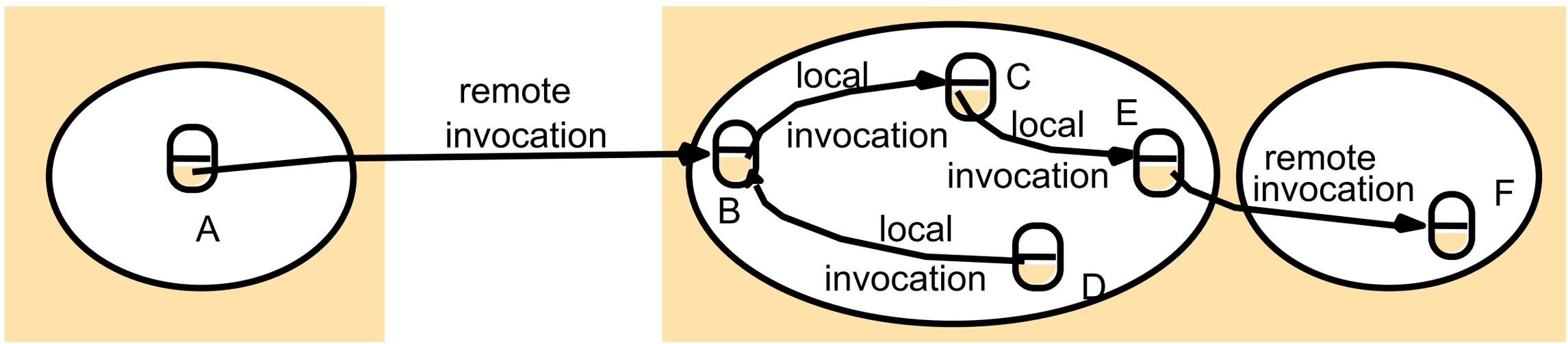
- A distribuição de objetos em diferentes processos em um SD é uma extensão natural do modelo POO.
  - Um objeto pode invocar a execução de um objeto remoto usando RMI.
  - Objetos podem ser replicados para se aumentar o grau de tolerância a falhas e o desempenho do sistema.

Encapsulamento

- O estado de um objeto não deve ser acessado diretamente por um objeto remoto
  - pode ser acessado somente através de um método apropriado.
- O uso de RMI significa que um objeto pode ser acessado de forma concorrente.

Invocação Remota (RMI)

- Um processo inclui um conjunto de objetos. Alguns deles podem ser acessados local e remotamente, outros, apenas localmente.



Modelo de Objetos Distribuídos

- Referência de objeto remoto
- Interface remota
- Coleta de lixo distribuída
  - baseada em contagem de referência
  - arrendamento (leasing)
- Exceções
  - a invocação de método remoto deve ser capaz de levantar exceções (timeout, perda de mensagens, falha no processo remoto, etc)

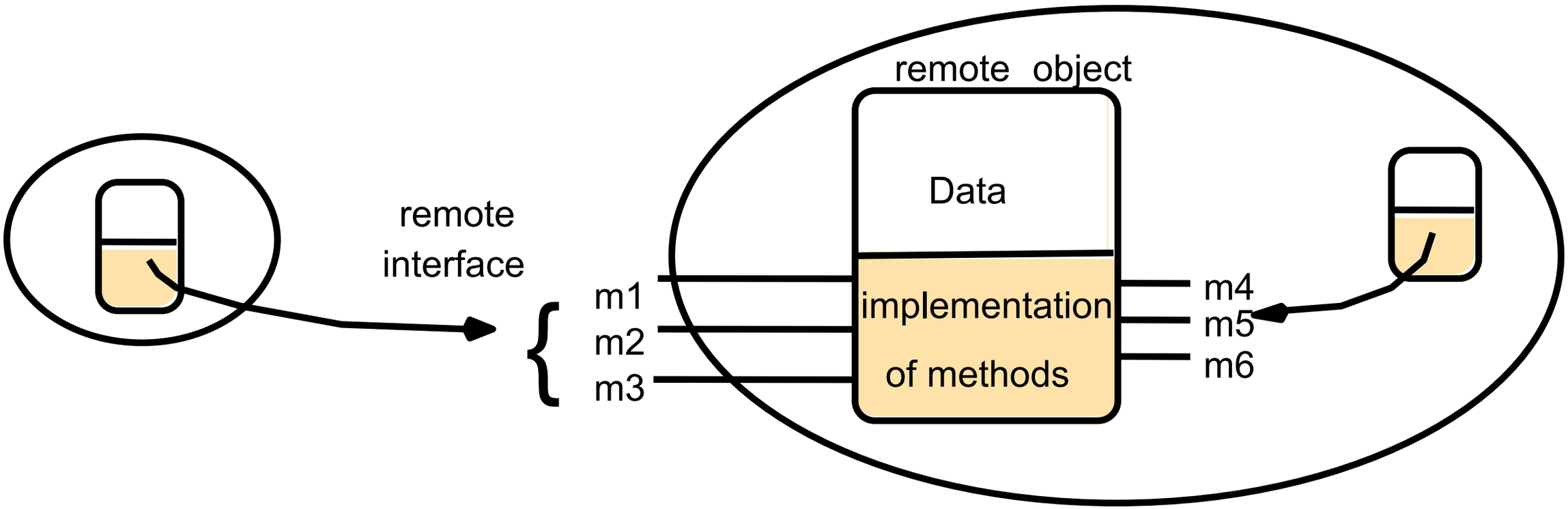
Referências de Objetos Remotos

- É um identificador que pode ser usado por todo um SD para se referir a um objeto único
- Permite que o objeto receba invocações remotas
- Podem ser passadas como argumentos

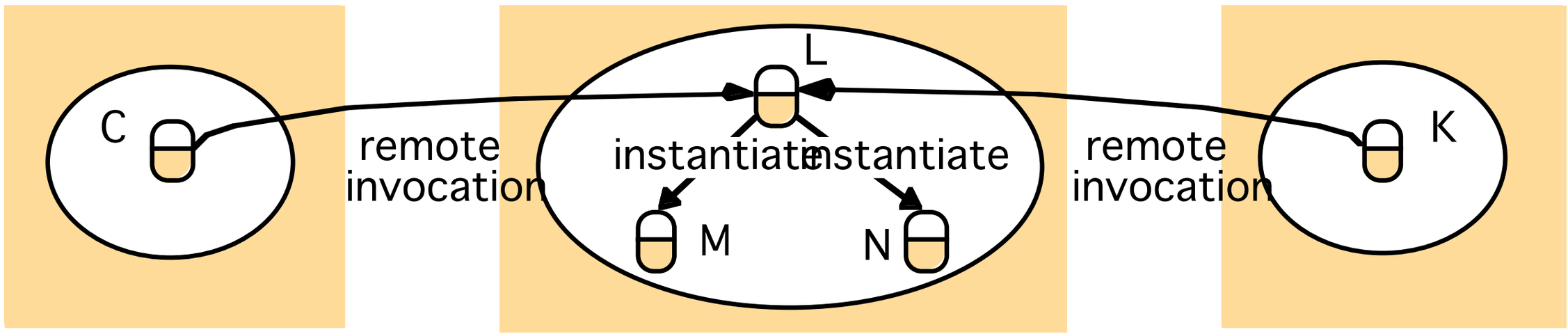
Interfaces Remotas

- Todo objeto remoto deve definir uma interface remota que especifica quais de seus métodos podem ser invocados remotamente
- A classe de um objeto remoto implementa os métodos da sua interface remota
- As classes podem ser implementadas em qualquer linguagem para a qual exista um compilador da IDL utilizada.

Interfaces Remotas



Instanciação de Objetos Remotos



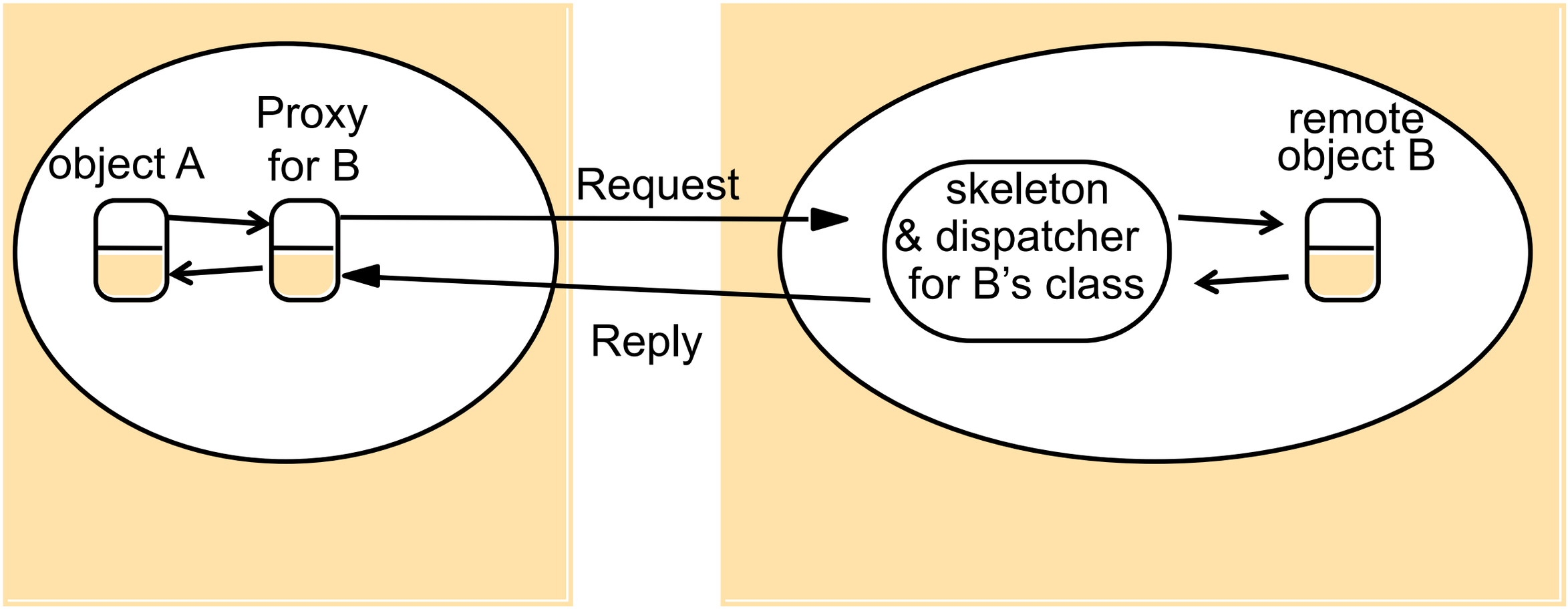
Desafios no Projeto de RMI

- Semântica de invocação
- Nível de transparência
  - a sintaxe de uma invocação remota é a mesma de uma invocação local
  - a diferença entre objetos locais e remotos deve ser expressa apenas nas interfaces

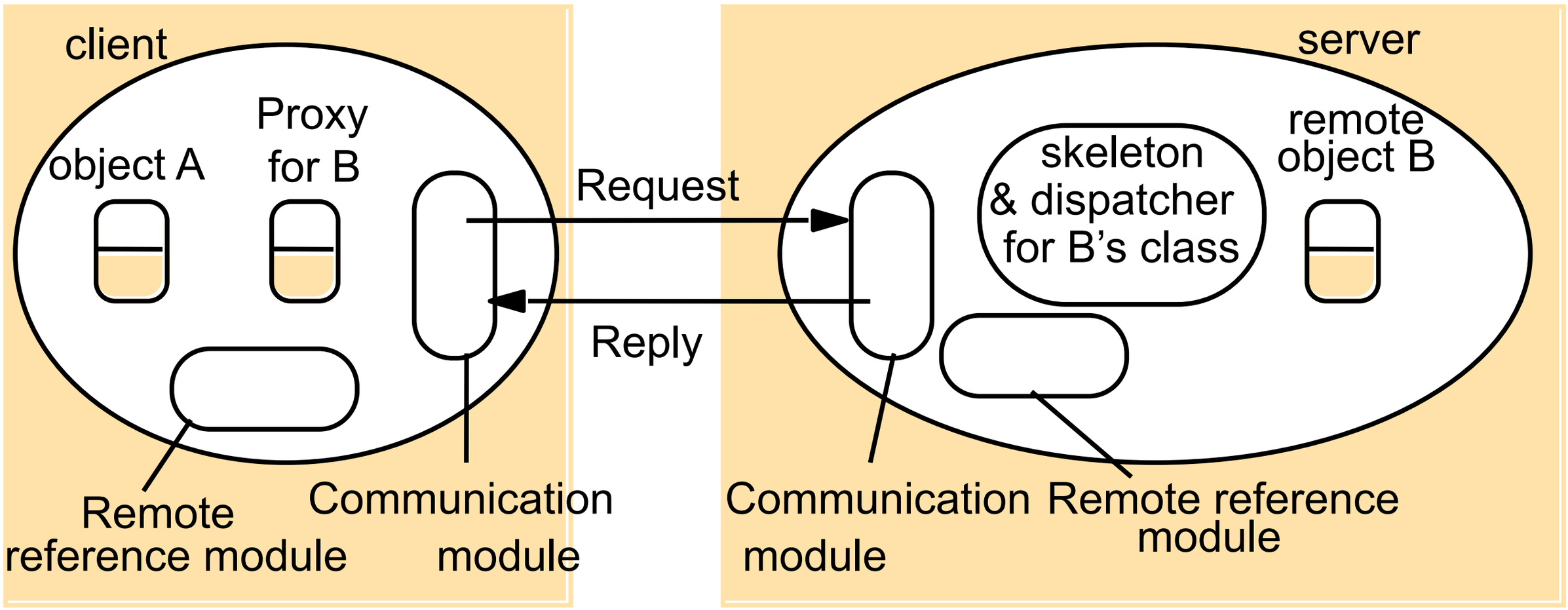
Garantias na Invocação Remota

- Retentativa de mensagem de requisição
  - retransmitir a requisição até que se obtenha uma resposta
- Filtragem de duplicatas
  - eliminar requisições duplicadas no servidor
- Retransmissão de resultados
  - manter histórico das respostas (resultado pode ser retransmitido sem que o método seja excutado novamente)

Implementação de RMI



Implementação de RMI

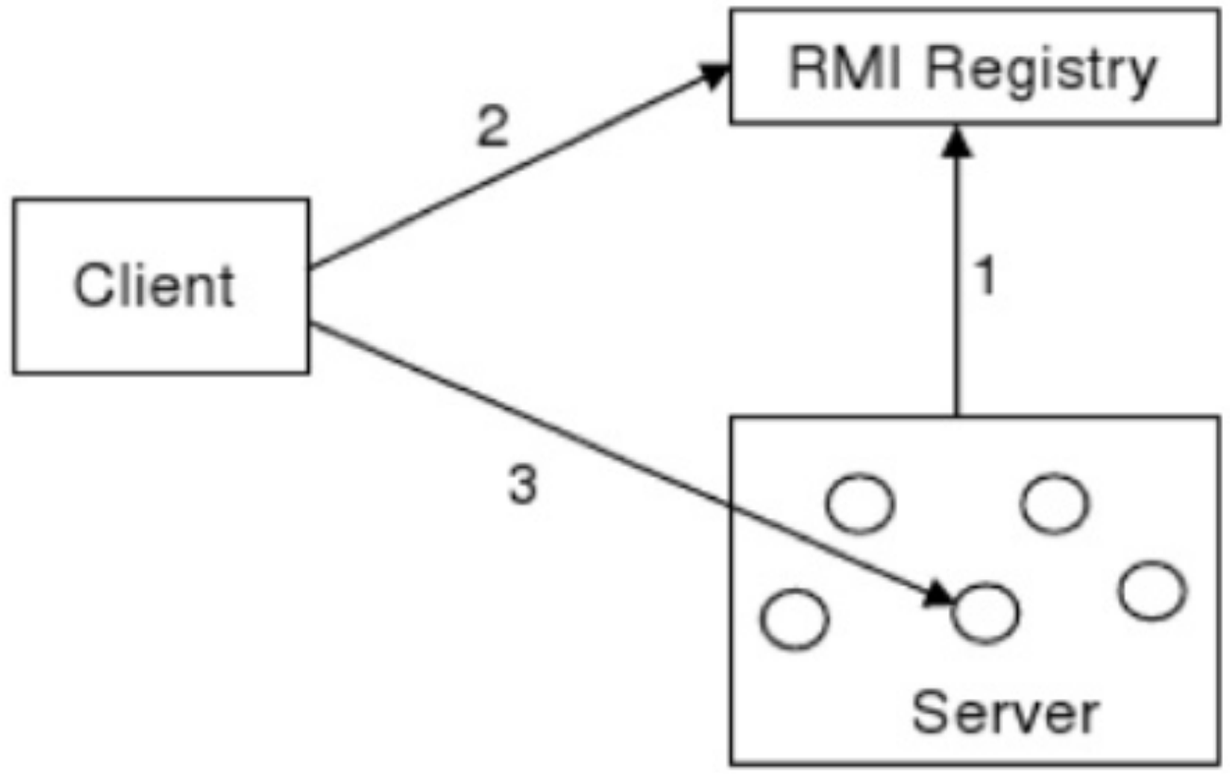


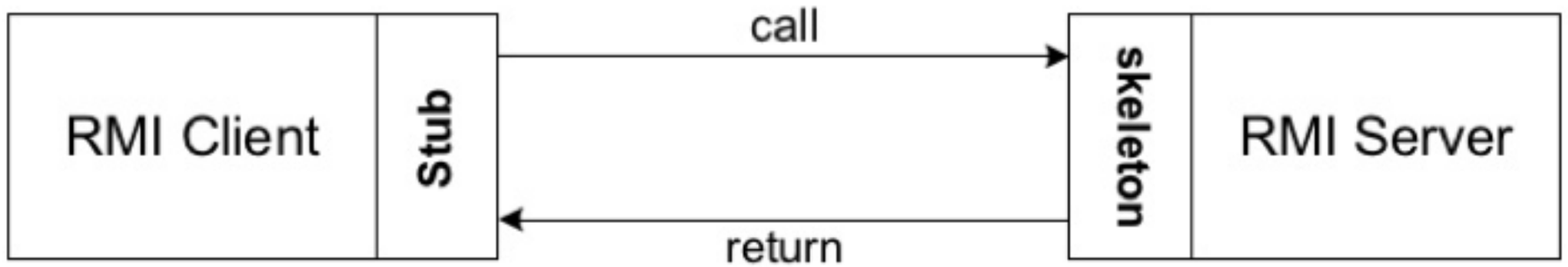
Estudo de Caso: Java RMI

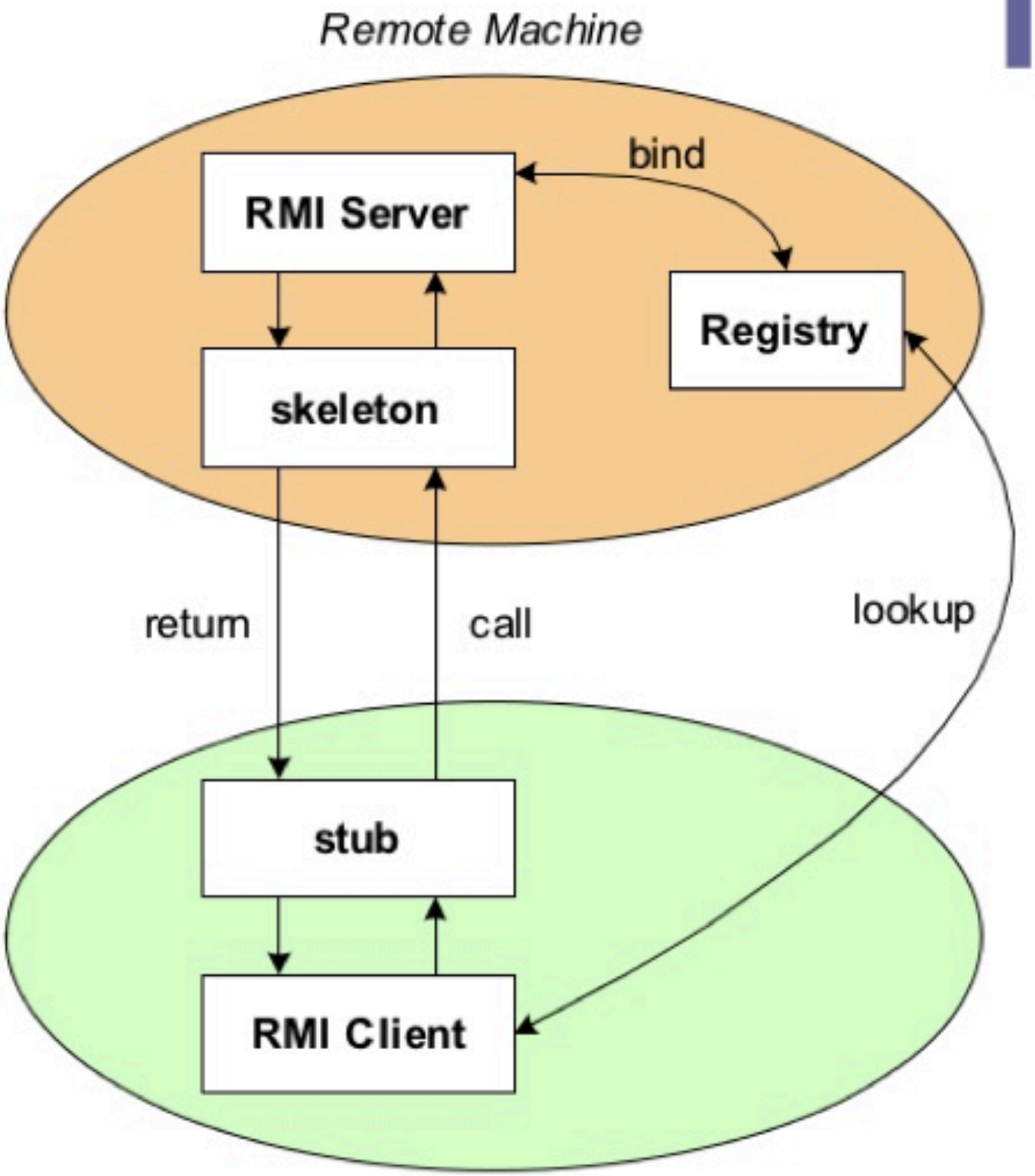
- Interfaces Remotas
  - public interface <Nome> extends Remote {...}
  - os métodos devem disparar uma exceção denominada RemoteException
- Passagem de parâmetros
  - passagem de objetos remotos
  - passagem por valor (objetos serializáveis e não-remotos → novo objeto criado no destino)

Java RMI

- Download de classes
  - se o destino não possui a classe de um objeto passado por valor ou a classe proxy de um objeto remoto, o código será transferido entre as máquinas virtuais (automaticamente)
- RMIregistry (vinculador/binder)
  - uma instância em cada servidor que contenha objetos remotos
  - mapeia nomes em referências de objetos remotos


RMI Registry
2
Client
1
3
Server


call
RMI Client
Stub
skeleton
RMI Server
return


Remote Machine
RMI Server
bind
Registry
skeleton
return
call
lookup
stub
RMI Client
Local Machine

Criando um aplicativo com Java RMI

1. Definir a interface Remota
2. Implementar o servidor
3. Implementar o cliente
4. Compilar (stubs são gerados automaticamente)
5. Iniciar o RMIregistry e depois o servidor
6. Iniciar o(s) cliente(s)