Universidade Tecnológica Federal do Paraná
Departamento Acadêmico de Computação
Bacharelado em Ciência da Computação

Sistemas Distribuídos

Invocação Remota

Prof. Rodrigo Campiolo

31/08/20

Tópicos

- Introdução
- RPC
- RMI
- Atividades

Introdução

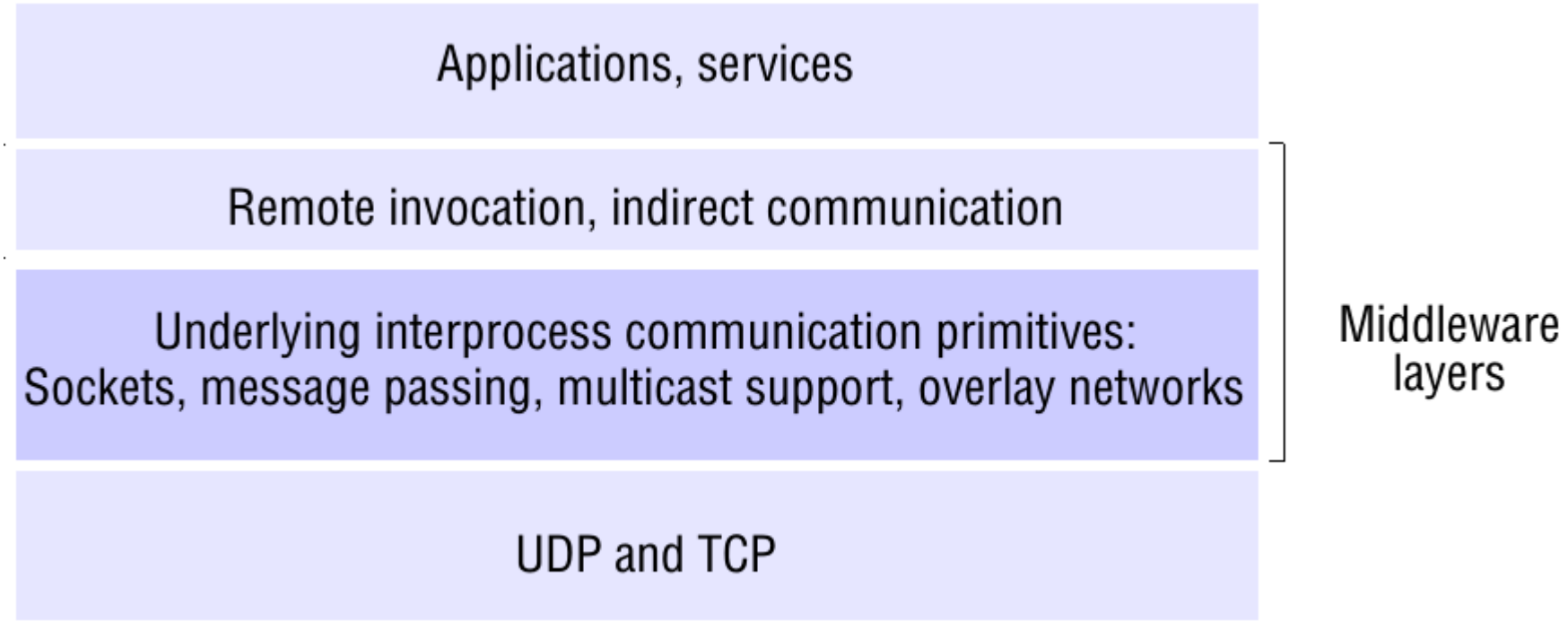
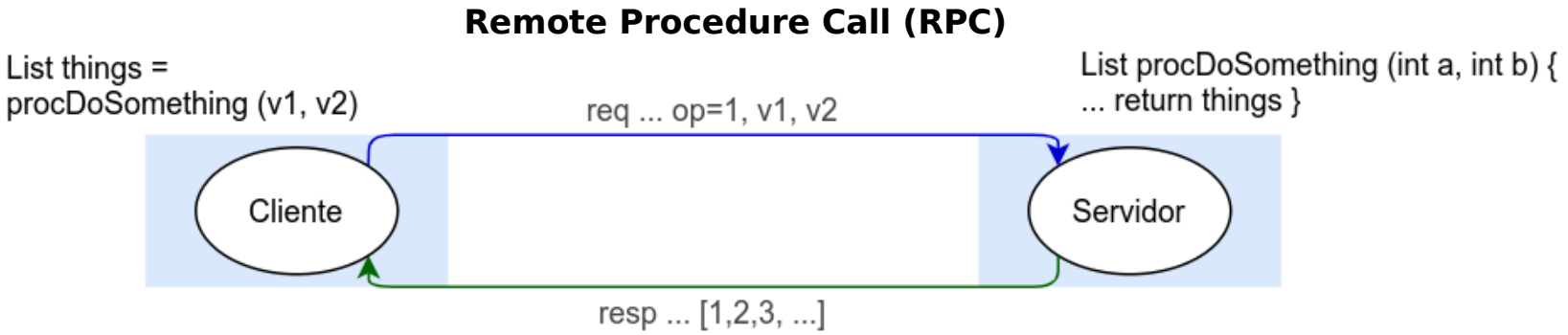


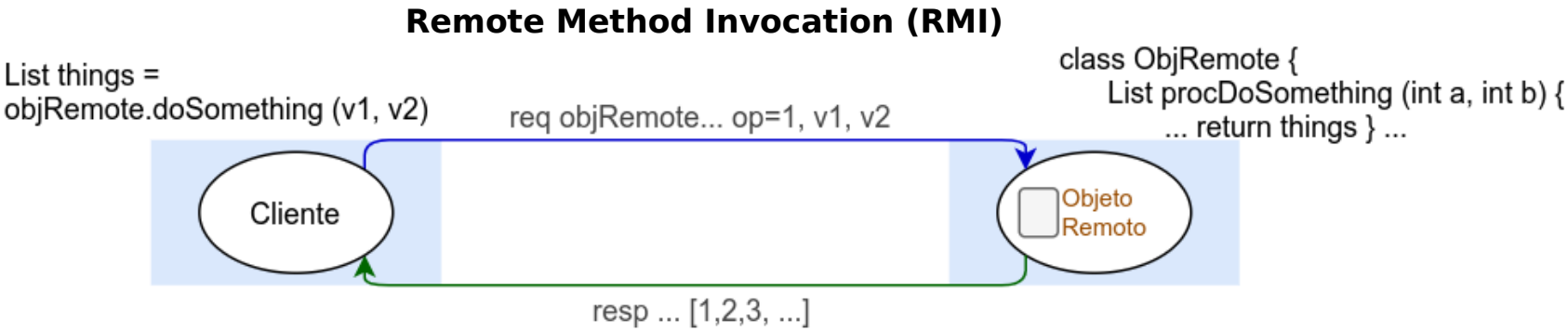
Figura 1: Camadas de Middleware

Fonte: Coulouris

Introdução

- Comunicação entre processos remotos por meio de invocação remota





Protocolos Request-Reply

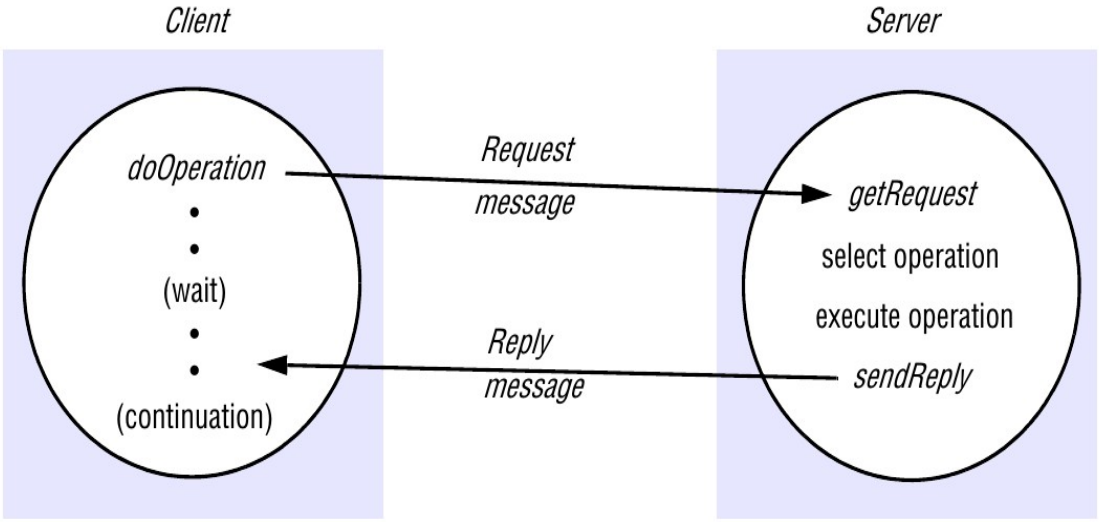


Figura 2: Comunicação Request-Reply

Fonte: Coulouris

messageType	*int (0=Request, 1= Reply)*
requestId	*int*
remoteReference	*RemoteRef*
operationId	*int or Operation*
arguments	*// array of bytes*

Figura 3: Estrutura de mensagem Request-Reply

Fonte: Coulouris

Protocolos Invocação Remota

Name	*Messages sent by*		
	Client	*Server*	*Client*
R	*Request*		
RR	*Request*	*Reply*	
RRA	*Request*	*Reply*	*Acknowledge reply*

Figura 4: Protocolos de troca de mensagens

Fonte: Coulouris

RPC

- Remote Procedure Call (RPC)
- Birrel and Nelson, 1984

- Conceitos:
  - Programação com interfaces
  - Semânticas de invocação
  - Transparência

RPC

- Programação com **interfaces**
  - Especifica os procedimentos/funções de um módulo que podem ser acessados por outros módulos.
  - As interfaces também definem as estruturas a serem usadas na comunicação.
  - Um conjunto de procedimentos/funções de um servidor geralmente é denominado de **interface de serviço**.
  - Uma interface de serviço especifica os nomes, as entradas e as saídas dos procedimentos.
  - **Interface Definition Language (IDL)** são usadas para especificar interfaces e possibilitar a invocação entre programas desenvolvidos em linguagens diferentes.

RPC

- Interfaces de Serviço – Exemplos
  - Google Cloud Print

    https://developers.google.com/cloud-print/docs/appInterfaces
  - API do Twitter
    - https://developer.twitter.com/en/docs/api-reference-index
  - Correios (WS)
    - http://ws.correios.com.br/calculador/CalcPrecoPrazo.asmx

RPC

- Semânticas de Invocação

Fault tolerance measures			*Call semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

Figura 5: Semânticas de invocação.

Fonte: Coulouris

Obs: semântica local é **exactly once**

RPC

- Transparência
  - Localização e acesso
  - Problema: falhas, latência.
  - Abortar uma invocação remota?
  - IDL

RPC

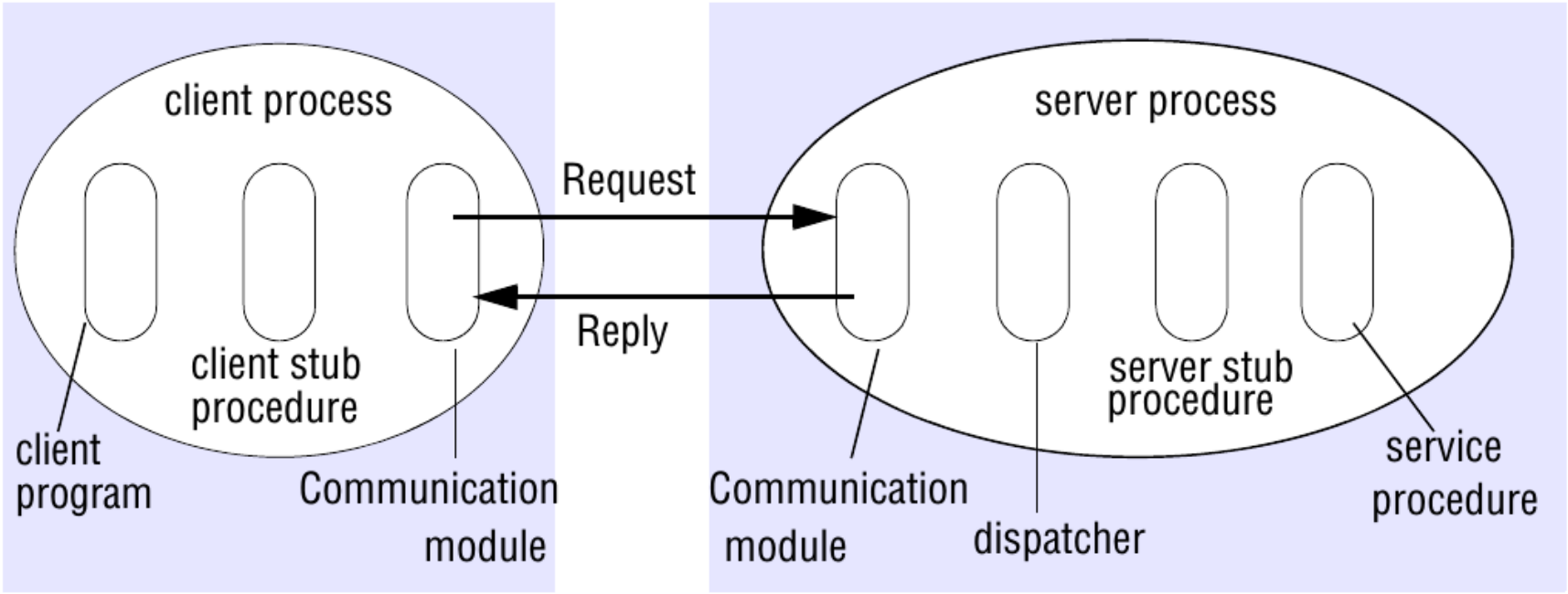


Figura 6: Invocação remota no RPC.

Fonte: Coulouris

RPC

- Exemplos
  - **Sun RPC**
  - **gRPC**
  - Apache Thrift
  - Apache Avro
  - XML-RPC e SOAP
  - RPyC
  - Microsoft RPC

RPC

- Sun RPC
  - Usado no Network File System (NFS)
  - Unix e Linux
  - Chamada sobre UDP e TCP
  - Sun XDR (IDL) e rpcgen (Compilador IDL)

```
enum color {ORANGE, PUCE, TURQUOISE};

struct list {
        string data<>;
        int key;
        color col;
        list *next;
};

program PRINTER {
        version PRINTER_V1 {
                int PRINT_LIST(list) = 1;
                int SUM_LIST(list) = 2;
        } = 1;
} = 0x2fffffff;
```

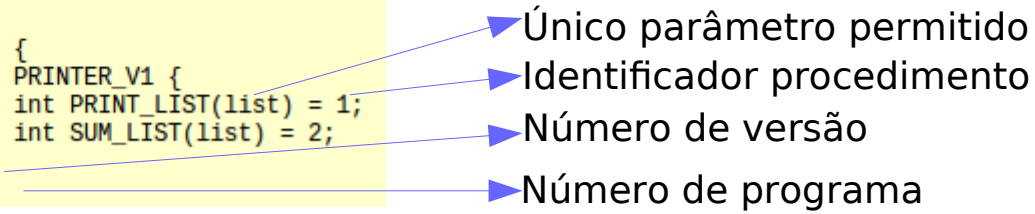


Figura 7: Definição de interface com Sun XDR

Fonte: http://www.cprogramming.com/tutorial/rpc/remote_procedure_call_start.html

RPC

- Sun RPC
  - Compilador de IDL gera:
    - Procedimentos stub do cliente
    - Procedimento principal (main), dispatcher e os procedimentos stub do servidor
    - Os procedimentos de marshalling and unmarshalling.
  - Usa um serviço de mapeamento local (*port mapper*) que registra o número do programa, número de versão e porta para os serviços locais.

RPC

- Sun RPC – Exemplo
  - llist.x

```
enum color {ORANGE, PUCE, TURQUOISE};

struct list {
        string data<>;
        int key;
        color col;
        list *next;
};

program PRINTER {
        version PRINTER_V1 {
                int PRINT_LIST(list) = 1;
                int SUM_LIST(list) = 2;
        } = 1;
} = 0x2fffffff;
```

RPC

- Sun RPC – Exemplo
  - Usando o **rpcgen**

```
$ rpcgen llist.x
```

  - Arquivos gerados:
    - llist.h : declarações
    - llist_svc.c : Serviço principal e dispatcher
    - llist_xdr.c : Marshalling and unmarshallig
    - llist_clnt.c : funções auxiliares RPC para o cliente

RPC

- Sun RPC – Exemplo
  - llist_svc_proc.c

```
int result;

/* print out a list, returning the number of items printed */
int *print_list_1_svc(list *lst, struct svc_req *req)
{
    list *ptr;

    ptr = lst;
    …
}

int *sum_list_1_svc(list *lst, struct svc_req *req)
{
    list *ptr;

    …
}
```

RPC

- Sun RPC – Exemplo
  - llist.c

```
#include "llist.h"

…

result = print_list_1(l, cl);

if (result == NULL) {
   printf("error: RPC failed!\n");
   return 1;
}

printf("client: server says it printed %d items.\n", *result);
```

RPC

- gRPC
  - Arcabouço RPC de alto desempenho e aberto.
  - Histórico
    - Google usa uma infraestrutura interna de RPC denominada Stubby.
    - Stubby é usado para interligar toda a infraestrutura de microsserviços da Google.
    - gRPC é uma implementação aberta de RPC baseada no Stubby.
    - Maiores detalhes: http://www.grpc.io/blog/principles

RPC

- gRPC
  - Características
    - Código aberto e gratuito.
    - Usa **protocol buffers** para serialização de dados.
    - Bibliotecas para diversas linguagens.
    - Suporte à autenticação, balanceamento de carga, monitoramento.
    - Possibilita conectar dispositivos, aplicações móveis e navegadores Web a serviços de *backend*.

Fonte: http://www.grpc.io/about/

RPC

- gRPC

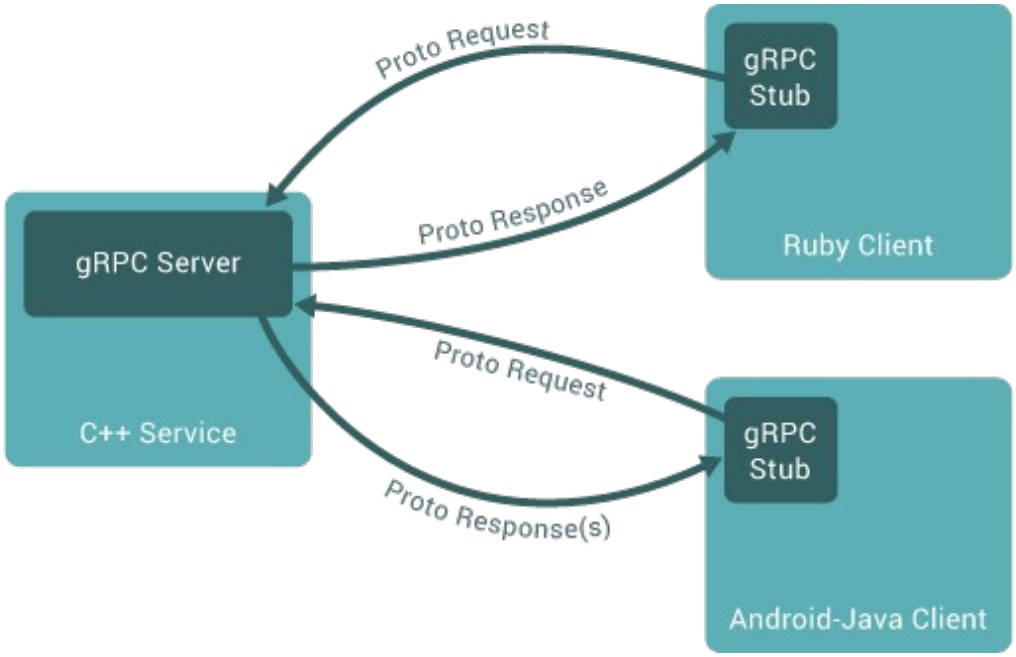


Figura: Visão geral do gRPC.

Fonte: http://www.grpc.io/docs/guides/index.html

RPC

- gRPC - Conceitos
  - Definição de serviços
    - Unary RPC
    - Server Streaming RPC
    - Client Streaming RPC
    - Bidirecional Streaming RPC
  - RPC síncrono e assíncrono
  - Deadlines/Timeouts
  - Cancelamento RPC
  - Channels (conexões para o servidor gRPC)

Fonte: http://www.grpc.io/docs/guides/concepts.html

RPC

- gRPC - Exemplo
  - helloworld.proto

```
syntax = "proto3";

// interface de serviço
service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// mensagem de solicitação
message HelloRequest {
  string name = 1;
}

// mensagem de resposta
message HelloReply {
  string message = 1;
}
```

RPC

- gRPC - Exemplo
  - Compilando a interface definida em protocol buffer

```
 # python3 -m grpc_tools.protoc  -I . --python_out=. --grpc_python_out=.
helloworld.proto
```

  - Resultado
    - helloworld_pb2_grpc.py (Stub)
    - helloworld_pb2.py (Serialização/Desserialização)

RPC

- gRPC – Exemplo
  - greeter_server.py

```
from concurrent import futures
import time

import grpc

import helloworld_pb2
import helloworld_pb2_grpc

_ONE_DAY_IN_SECONDS = 60 * 60 * 24

class Greeter(helloworld_pb2_grpc.GreeterServicer):

  def SayHello(self, request, context):
    return helloworld_pb2.HelloReply(message='Hello, %s!' % request.name)

def serve():
  server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
  helloworld_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)
  server.add_insecure_port('[::]:50051')
  server.start()
  try:
    while True:
      time.sleep(_ONE_DAY_IN_SECONDS)
  except KeyboardInterrupt:
    server.stop(0)

if __name__ == '__main__':
  serve()
```

RPC

- gRPC – Exemplo
  - greeter_client.py

```
from __future__ import print_function

import grpc

import helloworld_pb2
import helloworld_pb2_grpc


def run():
  channel = grpc.insecure_channel('localhost:50051')
  stub = helloworld_pb2_grpc.GreeterStub(channel)
  response = stub.SayHello(helloworld_pb2.HelloRequest(name='you'))
  print("Greeter client received: " + response.message)


if __name__ == '__main__':
  run()
```

Atividades

- Faça uma implementação usando Sun RPC ou gRPC para um serviço remoto de gerenciamento de músicas/livros que possibilita adicionar, remover e consultar músicas/livros.

RMI

- Invocação Remota de Método (*Remote Method Invocation - RMI*)
- Passagem de referências de objetos
- Cliente invoca um método de um objeto no servidor.

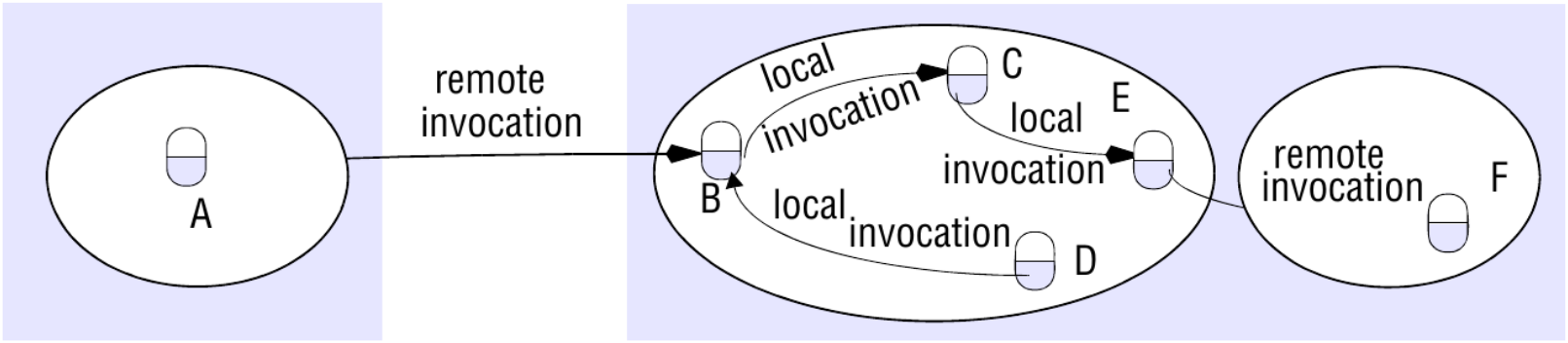


Figura 8: Invocações de método remoto

Fonte: Coulouris

RMI

- Referências para objetos remotos
  - Identificador único para um objeto no sistema distribuído.
  - Podem ser passadas como parâmetros

Figura 9: Representação para objeto remoto

Fonte: Coulouris

RMI

- Interface remota
  - Especifica os métodos que podem ser invocados remotamente
  - Exemplo (Java RMI)

```
public interface Calculadora extends Remote {
    public int soma (int a, int b) throws RemoteException;
    public int subtrai (int a, int b) throws RemoteException;
    public int divide (int a, int b) throws RemoteException;
    public int multiplica (int a, int b) throws RemoteException;
} //Calculadora
```

Figura 10: Interface para uma calculadora remota

RMI

- Questões
  - Ações distribuídas
  - Coleta de lixo distribuída
  - Exceções distribuídas

RMI - Implementação

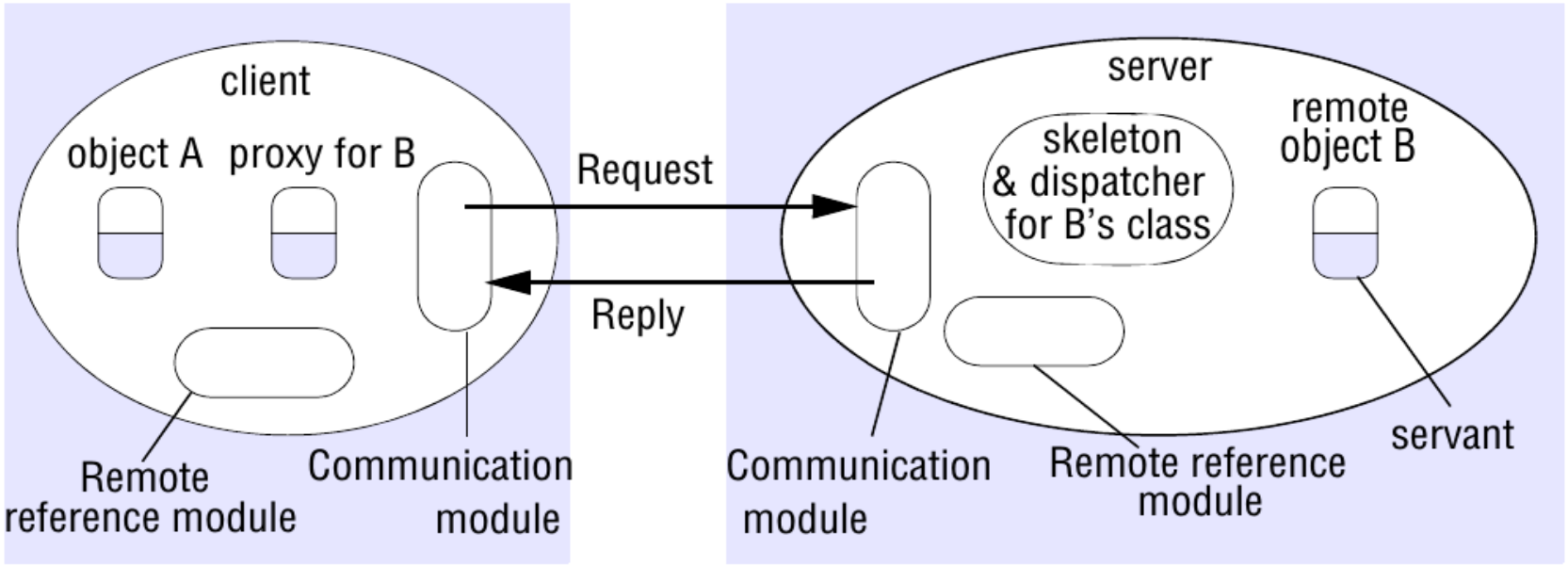


Figura 11: Componentes para a comunicação RMI

Fonte: Coulouris

RMI - Implementação

- **Módulo de comunicação**: controla o processo de solicitação e resposta (tipo da mensagem, identificação da solicitação e referência do objeto remoto)

- **Módulo de referência remoto**: resolução de referências locais e remotas e criação de referências remotas. Os processos mantém uma **tabela de objetos remotos** que associam referências remotas e *proxies* locais.

RMI - Implementação

- **Servant**: Instância de uma classe que provê o corpo da implementação de um objeto remoto.
- **Proxy**: Realiza marshalling e unmarshalling das requisições remotas no cliente.
- **Skeleton**: Realiza unmarshalling and marshalling das requisições no servidor.
- **Dispatcher**: Seleciona o método no skeleton.

RMI – Implementação

- Há um *dispatcher* e um *skeleton* associado a cada objeto remoto.
- Há um *proxy* para cada referência remota mantida no cliente.
- **Invocação dinâmica**: realiza a solicitação sem o uso direto de um *proxy*. Por exemplo, um cliente recebe a referência de um objeto remoto durante a execução.
- Alternativa à invocação dinâmica: Download dinâmico de classes.

RMI - Implementação

- Programa servidor
  - Classes para dispatcher, skeletons, interfaces, implementação das interfaces (servants).
  - Inicializar os objetos remotos.
  - Registrar os objetos remotos ou um objeto remoto com métodos Fábrica.
  - O registro pode ser realizado em um **Binder**.

RMI - Implementação

- Programa cliente
  - Localiza e obtém a referência remota (geralmente por meio de um Binder).
  - Armazena a referência em um objeto e realiza as invocações como se fosse um objeto local.

RMI - Implementação

- **Ativadores**: Processos que inicializam processos no servidor quando objetos remotos são solicitados.
- **Objetos persistentes**: objeto que continua ativo entre ativações de processos.
- **Serviço de localização**: auxilia clientes a localizar objetos remotos a partir da referência remota.
- **Coleta de lixo distribuída**: garantir que um objeto remoto seja removido se não é mais referenciado. Possibilidade de manter a contagem de referência.

RMI - Implementação

- **Lease** (Arrendamento): Concessão de uso de um recurso por um período de tempo.
- **Callbacks**: O servidor informa a seus clientes que um evento ocorreu.

RMI Java - Exemplo

- Acessar o exemplo de serviço calculadora no Moodle.

  Exemplos → RMI Java e Python - Calculadora remota

Atividades

- Faça uma implementação usando Java RMI para um serviço remoto de gerenciamento de músicas/livros que possibilita adicionar, remover e consultar músicas/livros.

Atividades

- Especifique uma interface para um serviço de impressão.
- Especifique uma interface para um serviço de chat que possibilite que usuários se comuniquem em grupo ou pares. Os usuários podem entrar e sair do grupo a qualquer momento. Podem criar novos grupos. Todos os grupos criados são públicos, logo qualquer usuário pode acessar a lista e acessar os grupos. O serviço não possui um mecanismo de segurança.

Referências

COULOURIS, George F; DOLLIMORE, Jean; KINDBERG, Tim; BLAIR, Gordon. **Sistemas distribuídos: conceitos e projeto**. 5. ed. Porto Alegre: Bookman, 2013.