Unichristus
Centro Universitário Christus

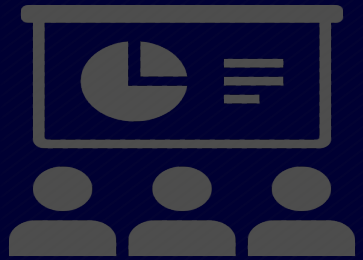
SISTEMAS DISTRIBUÍDOS

Modelos de Arquitetura

Lattes - linkedin
euristenhojr@gmail.com
http://www.unichristus.edu.br/

Euristenho Queiroz de Oliveira **Júnior**
Especialista em Engenharia de Software
MSc em Engenharia de Software

AGENDA



1. Apresentação
2. Livros
3. Modelos
4. RPC
5. RMI
6. Próxima Aula
7. Referências

FORMAÇÃO ACADÊMICA

- Graduado em Telemática/Telecomunicações - IFCE (2002 - 2008)
- Especialista em Engenharia de Software - FA7 (2011 - 2013)
- MSc em Engenharia de Software - UFPE (2011 - 2015)

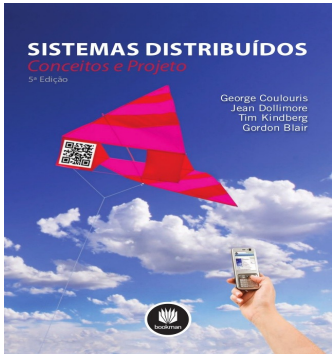
CURRÍCULO PROFISSIONAL

- Atuei 4 anos na empresa privada
- 9 anos no ambiente Público
- Atualmente Líder Técnico de 45 Projetos de Tecnologia na SEPOG/PMF

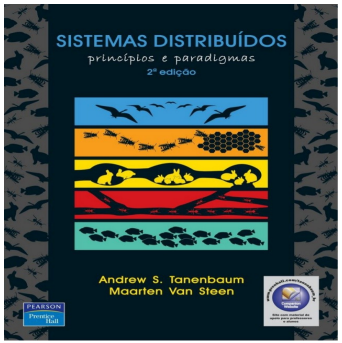
DOCÊNCIA

- Professor Substituto das Disciplinas de Sistemas de Informação – FA7 (2011 - 2012)
- Professor da Especialização em Sistemas WEB – FJN (2011 - 2012)
- Professor de Bancas de graduação em Sistemas de Informações – FA7 (2012)
- Professor dos Cursos de Tecnologia da Unifanor (2015 - ATUAL)
- Professor da Unichristus (2018 - ATUAL)

- **Sistemas Distribuídos - Conceitos e Projeto** - 5ª Ed. 2013 - George Coulouris, Tim Kindberg, Jean Dollimore



- **Sistemas Distribuídos, Princípios e Paradigmas** - 2ª Ed. 2007 - Andrew S. Tanembaum, Maarten Van Steen



OBJETIVOS

- Apresentar os conceitos básicos da computação distribuída e seus desafios como Heterogeneidade; Segurança; Tolerância a Falhas; Escalabilidade; Concorrência; Coordenação e Sincronização de processos; Comunicação interprocessos.
- Desenvolver competências e habilidades que auxiliem o profissional de Ciência da Computação a implementar os conceitos de sistemas distribuídos no desenvolvimento de sistemas de informação.
- Conhecer a aplicação desses conceitos em estudos de Casos que abordam arquiteturas e tecnologias modernas como RMI, CORBA e Web Services.

Introdução

- Modelos Físicos
- Modelos de arquitetura para sistemas distribuídos
- Modelos fundamentais

Qual o impacto do sistema distribuído se basear somente na troca de mensagens entre os dispositivos?

Quais as dificuldades e ameaças para os sistemas distribuídos?

- **Dificuldades e ameaças para os sistemas distribuídos**:
  - **Variados modos de uso**: variações de carga de trabalho, requisitos especiais de algumas aplicações etc
  - **Ampla variedade de ambientes de sistema**
  - **Problemas internos**: relógios não sincronizados, atualizações conflitantes de dados, diferentes modos de falhas de hardware e de software envolvendo os componentes individuais de um sistema
  - **Ameaças externas**: ataques à integridade e ao sigilo dos dados, negação de serviço, etc.

- Será visto a forma de **agrupar as características e problemas** nos sistemas distribuídos através do uso de **modelos descritivos**
- **Modelos**:
  - **Físicos**: capturam a **composição de hardware** de um sistema e suas **redes de interconexão**
  - **Arquitetura**: descrevem um sistema em termos das **tarefas computacionais e de comunicação**.
  - **Fundamentais**: Analisam aspectos individuais
    - **Modelo de Interação**: estrutura e ordenação dos elementos
    - **Modelo de Falha**: maneiras que o sistema pode deixar de funcionar
    - **Modelo de Segurança**: analisa se o sistema está protegido contra tentativas de interferência em seu funcionamento correto ou de roubo de seus dados.

Agenda

1. Introdução
2. **Modelos Físicos**
3. Modelos de arquitetura para sistemas distribuídos
4. Modelos fundamentais

Modelos Físicos

- É uma representação dos elementos de hardware de um sistema distribuído
  - **Visa abstrair os detalhes específicos do computador e das tecnologias de rede empregadas**.
- Tipos de modelos:
  - **Modelo físico básico** (computadores interconectados / passagem de mensagens)
  - **Sistemas distribuídos primitivos** (surgiu em 1970; Ethernet; Pequena quantidade de serviços)
  - **Sistemas distribuídos adaptados para Internet** (Heterogeneidade)
  - **Sistemas distribuídos contemporâneos** (Computação Móvel, ubíqua e em nuvem)
  - **Sistemas distribuídos de sistemas**

Modelos Físicos

Sistemas distribuídos	*Primitivos*	*Adaptados para Internet*	*Contemporâneos*
Escala	Pequenos	Grandes	Ultragrandes
Heterogeneidade	Limitada (normalmente, configurações relativamente homogêneas)	Significativa em termos de plataformas, linguagens e *middleware*	Maiores dimensões introduzidas, incluindo estilos de arquitetura radicalmente diferentes
Sistemas abertos	Não é prioridade	Prioridade significativa, com introdução de diversos padrões	Grande desafio para a pesquisa, com os padrões existentes ainda incapazes de abranger sistemas complexos
Qualidade de serviço	Em seu início	Prioridade significativa, com introdução de vários serviços	Grande desafio para a pesquisa, com os serviços existentes ainda incapazes de abranger sistemas complexos

Agenda

1. Introdução
2. Modelos Físicos
3. **Modelos de arquitetura para sistemas distribuídos**
4. Modelos fundamentais

- Estrutura em termos de **componentes especificados separadamente e suas inter-relações**.
- O objetivo global é garantir que a **estrutura atenda às demandas atuais** e, provavelmente, às futuras demandas impostas sobre ela.
    - As maiores preocupações são tornar o **sistema confiável, gerenciável, adaptável e rentável**
- Serão abordado o estudo em três áreas:
    - Elementos arquitetônicos
    - Padrões
    - Plataformas de middleware

- **Elementos arquitetônicos**:
  - Para o entendimento desses elementos, é necessário responder quatro perguntas:
    - Quais são as **entidades que estão se comunicando** no sistema distribuído?
    - **Como elas se comunicam** ou, mais especificamente, qual é o **paradigma de comunicação** utilizado?
    - Quais **funções e responsabilidades** (possivelmente variáveis) estão relacionadas a eles na arquitetura global?
    - Como eles são mapeados na infraestrutura distribuída física **(qual é sua localização)**?

Quais entidades se comunicam nos sistemas distribuídos?

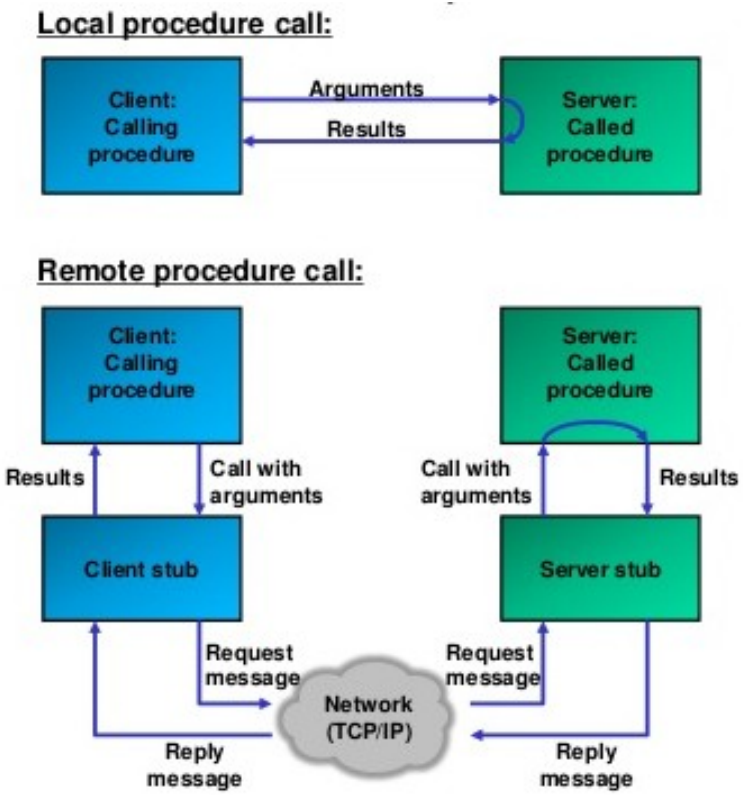
- **Elementos arquitetônicos: Entidades em comunicação**
  - Do **ponto de vista do sistema**, as entidades que se comunicam em um sistema distribuído normalmente são **processos**. Ressalvas:
    - **Ambiente primitivos** (ex: redes de sensores)
    - Complementação dos processos através de **threads**.

- **Elementos arquitetônicos: Entidades em comunicação**
  - Do **ponto de vista da programação**, tem-se as seguintes abstrações:
    - **Objetos**: uso de estratégias orientadas a objeto nos sistemas distribuídos
      - uma computação consiste em **vários objetos interagindo**
      - Os objetos são **acessados por meio de interfaces**, com uma **linguagem de definição de interface**
    - **Componentes**: oferecem abstrações voltadas ao problema e são acessados por meio de interfaces.
      - Diferença para os objetos: componentes especificam
        - **Interfaces (fornecidas)**
        - **Suposições** que fazem em termos de outros componentes/interfaces (dependências externas) necessárias ao funcionamento

- **Elementos arquitetônicos: Entidades em comunicação**
  - Do **ponto de vista da programação**, tem-se as seguintes abstrações:
    - **Serviços Web**:
      - Definição do W3C
        - Aplicativo de software **identificado por um URI**, cujas interfaces e vínculos podem ser definidos, descritos e descobertos como **artefatos da XML**.
        - Um serviço Web suporta interações diretas com outros agentes de software, usando **trocas de mensagens baseadas em XML por meio de protocolos Internet**.
      - Considerados serviços completos, em comparação com os objetos e componentes (internos da organização)

- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Principais paradigmas:
    - **Comunicação entre processos**: se refere ao suporte de nível relativamente baixo para comunicação entre processos nos sistemas distribuídos
    - **Invocação remota**: baseada na troca bilateral entre as entidades que se comunicam em um sistema distribuído e **resultando na chamada de uma operação, um procedimento ou método remoto**

- **Elementos arquitetônicos: Paradigmas de comunicação**



- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Principais paradigmas:
    - **Protocolos de requisição-resposta**: os protocolos de requisição-resposta são efetivamente um padrão imposto em um serviço de passagem de mensagens para suportar computação cliente-servidor

- **Elementos arquitetônicos: Paradigmas de comunicação**
  - As operações de chamada de procedimento remoto e invocação de método remoto **são suportadas pelas trocas requisição-resposta**:
    - **Chamada de procedimento remoto (RPC)**: procedimentos nos processos de computadores remotos podem ser chamados como se fossem **procedimentos no espaço de endereçamento local**.
      - O sistema de RPC **oculta aspectos importantes da distribuição**, incluindo a codificação e a decodificação de parâmetros e resultados, a passagem de mensagens e a preservação da semântica exigida para a chamada de procedimento

Sistemas Distribuídos: RPC Básico

Procedimento **PrepararPizzaPortuguesa**{
ingredientes
2 xícaras (chá) de farinha de trigo
1 colher (sopa) de fermento em pó
….
MododePreparo{
1. Faça uma cova no meio da farinha depois de peneirada com o fermento e o sal
2. Acrescente a água e o óleo
3. Amasse bem e abra com o rolo, formando uma massa lisa
…
}
}

Fonte: http://www.tudogostoso.com.br/receita/2807-pizza-a-portuguesa.html

Procedimento **PrepararSucoAbacaxiHortela**{
ingredientes
1 abacaxi maduro
10 folhas de hortelã
MododePreparo{
1. Descasque o abacaxi e bata no liquidificador com as folhas de hortelã
2. Coloque a água, o açúcar e 5 pedras de gelo e bata por 3 minutos
3. Coloque o restante do gelo na jarra de suco ou nos copos a serem servidos
4. Rende 5 copos de 200 ml
}
}

Fonte: http://www.tudogostoso.com.br/receita/112517-suco-de-abacaxi-com-hortela.html

Sistemas Distribuídos: RPC Básico

Quanto **Tempo** você levaria **para preparar** isto**?**

Sistemas Distribuídos: RPC Básico

Quanto **Tempo** você levaria **para preparar** isto**?**

Sistemas Distribuídos: RPC Básico

Vamos analisar os resultados?

1 Limitação de Recursos
2. Tempo maior de Processamento
3. No caso de falhas do recurso, o resultado desejado é afetado diretamente

Sistemas Distribuídos: RPC Básico

Lanchonete: Real Sucos

4. SolicitarSucoAbacaxiHortela(1 pessoa){ }

5. **PrepararSucoAbacaxiHortela**(1 pessoa){ }

Restaurante: Pizza Hut

1. QueroPizzaPortuguesa{}
2. QueroSucoAbacaxiHortela{}

3. SolicitarSucoAbacaxiHortela(1 pessoa){ }

2. SolicitarPizzaPortuguesa(pequena){ }

5. **PrepararPizzaPortuguesa**(pequena){ }

Sistemas Distribuídos: RPC Básico

Vamos analisar os resultados?

1 Recursos compartilhados
2. Tempo menor de Processamento
3. Possibilidade de Processamentos Paralelos
4. Abstração do método de preparo
5. Abstração da localização de execução dos métodos
6. Não imune a falhas

DEFINIÇÃO

Remote Procedure Call, originalmente definido como Open Network Computing Remote Call (ONC RPC), surgiu na década de 70 desenvolvido pela Sun Microsystems. A chamada de procedimento remoto (RPC) é uma técnica que originalmente seguia um modelo cliente / servidor e permitia que chamadas locais fossem executadas de forma transparente em recursos remotos ocultando entrada/saídas de mensagens.

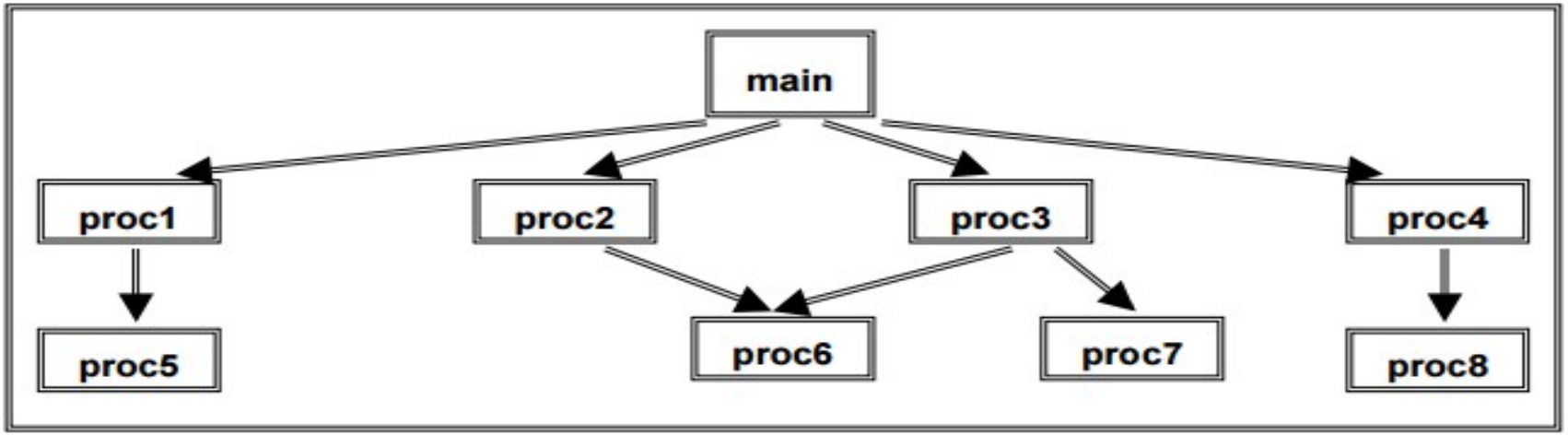
(COULOURIS; KINDBERD;DOLLIMORE, 2013, p146)

Sistemas Distribuídos: RPC Intermediário

Como seria o funcionamento do RPC?

Sistemas Distribuídos: RPC Intermediário

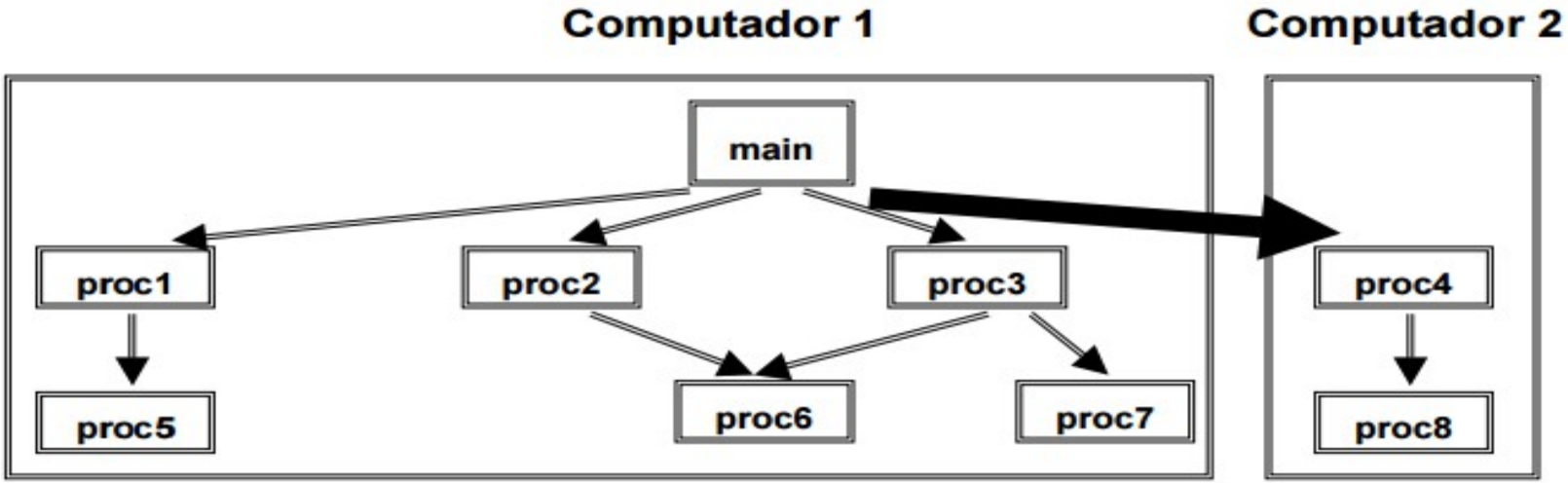
- Chamada de procedimento local:



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

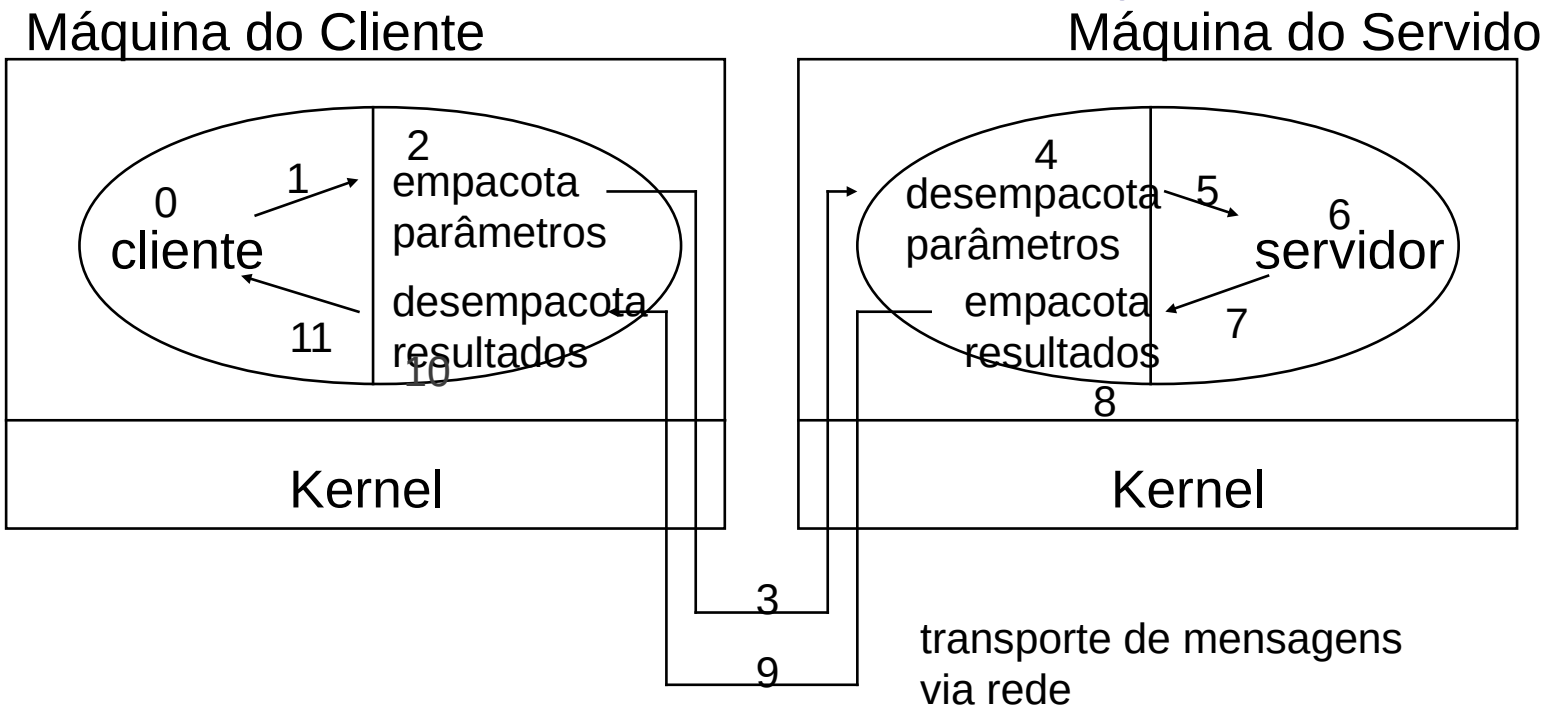
- Chamada de procedimento remoto:
  - Necessidade de um protocolo de comunicação para a implementação da chamada



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

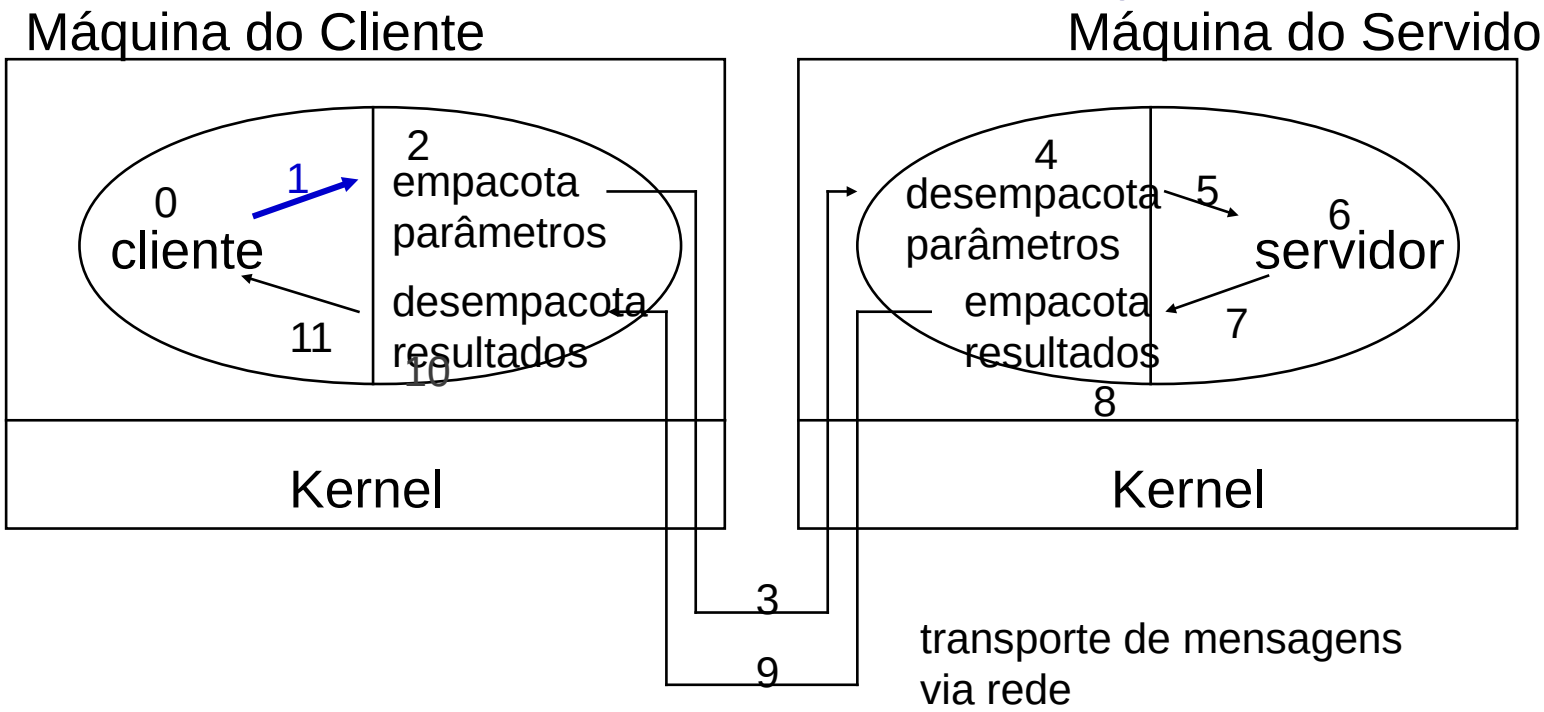
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

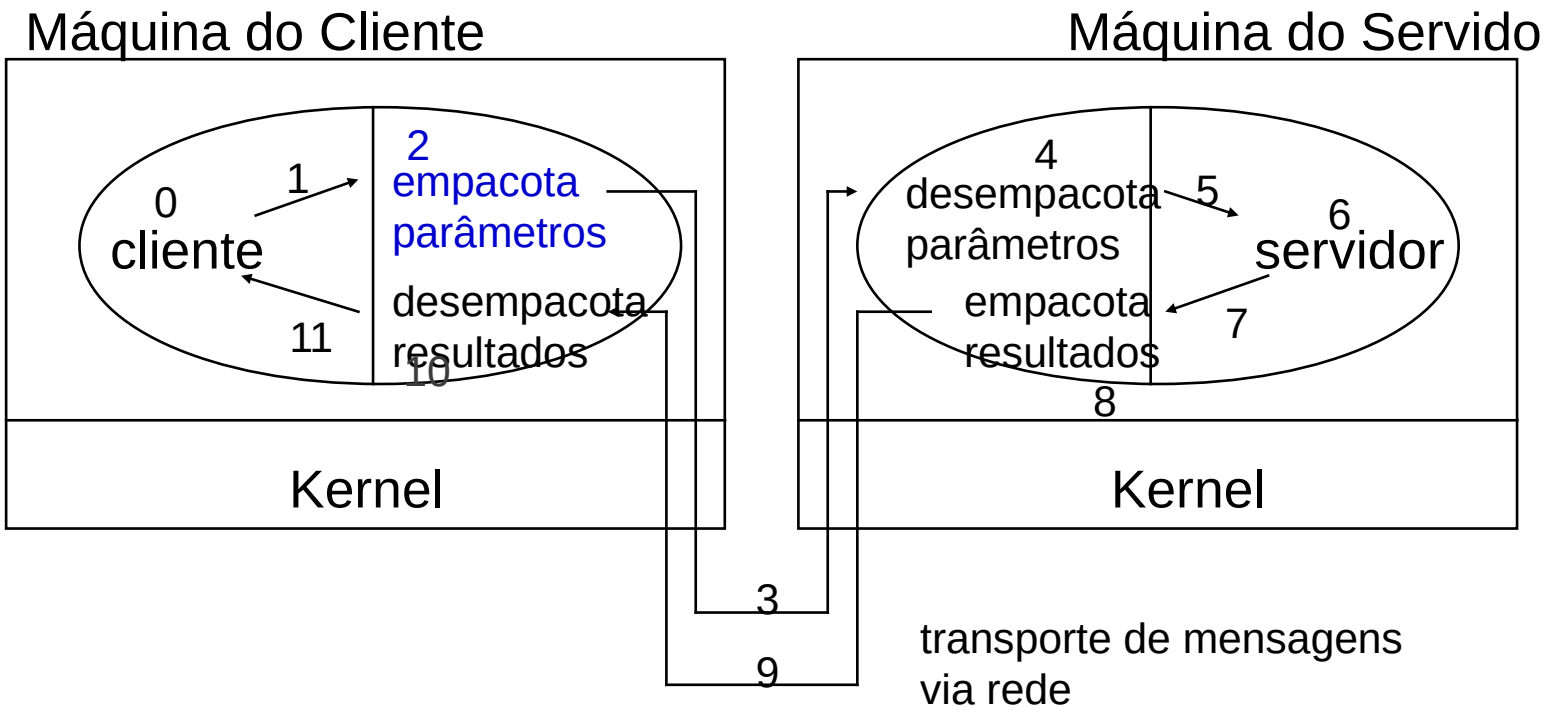
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

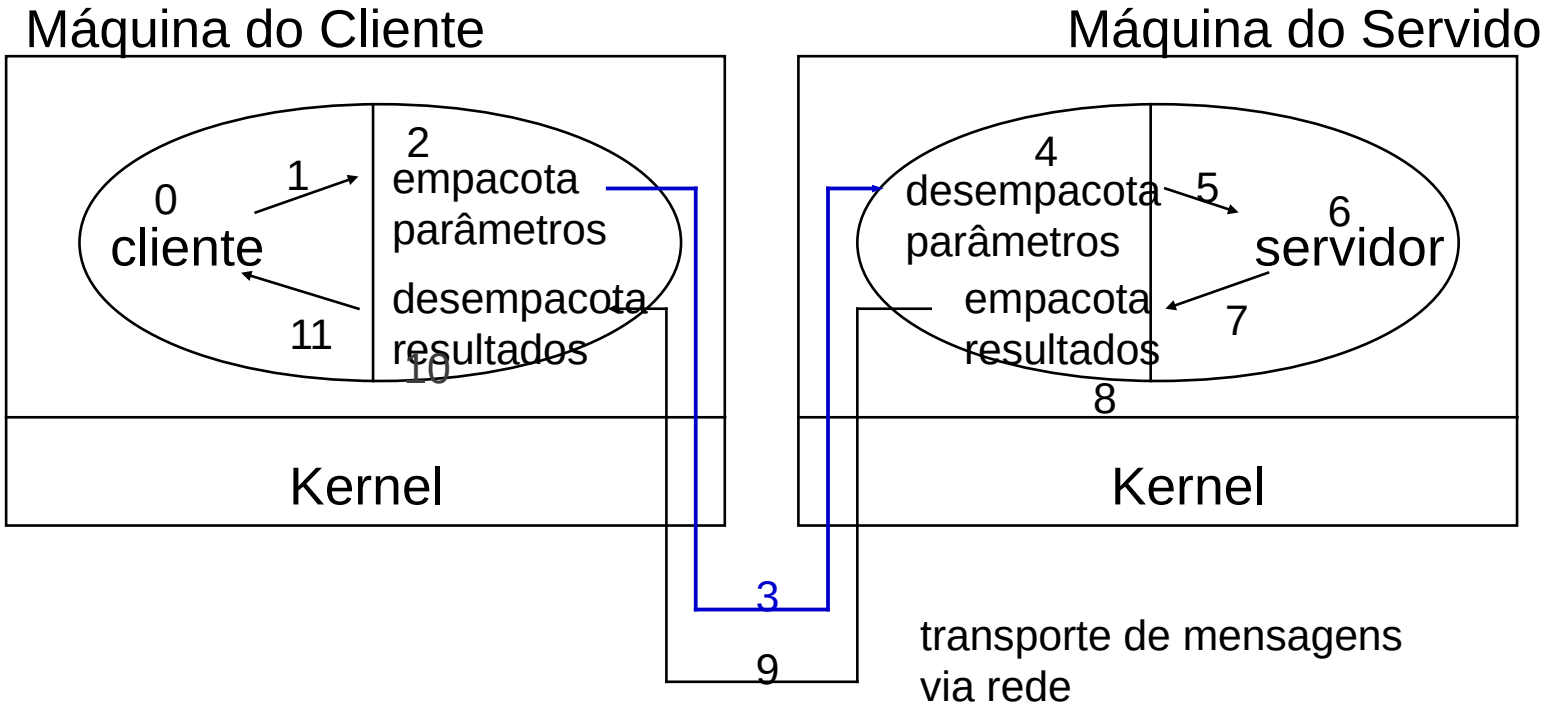
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

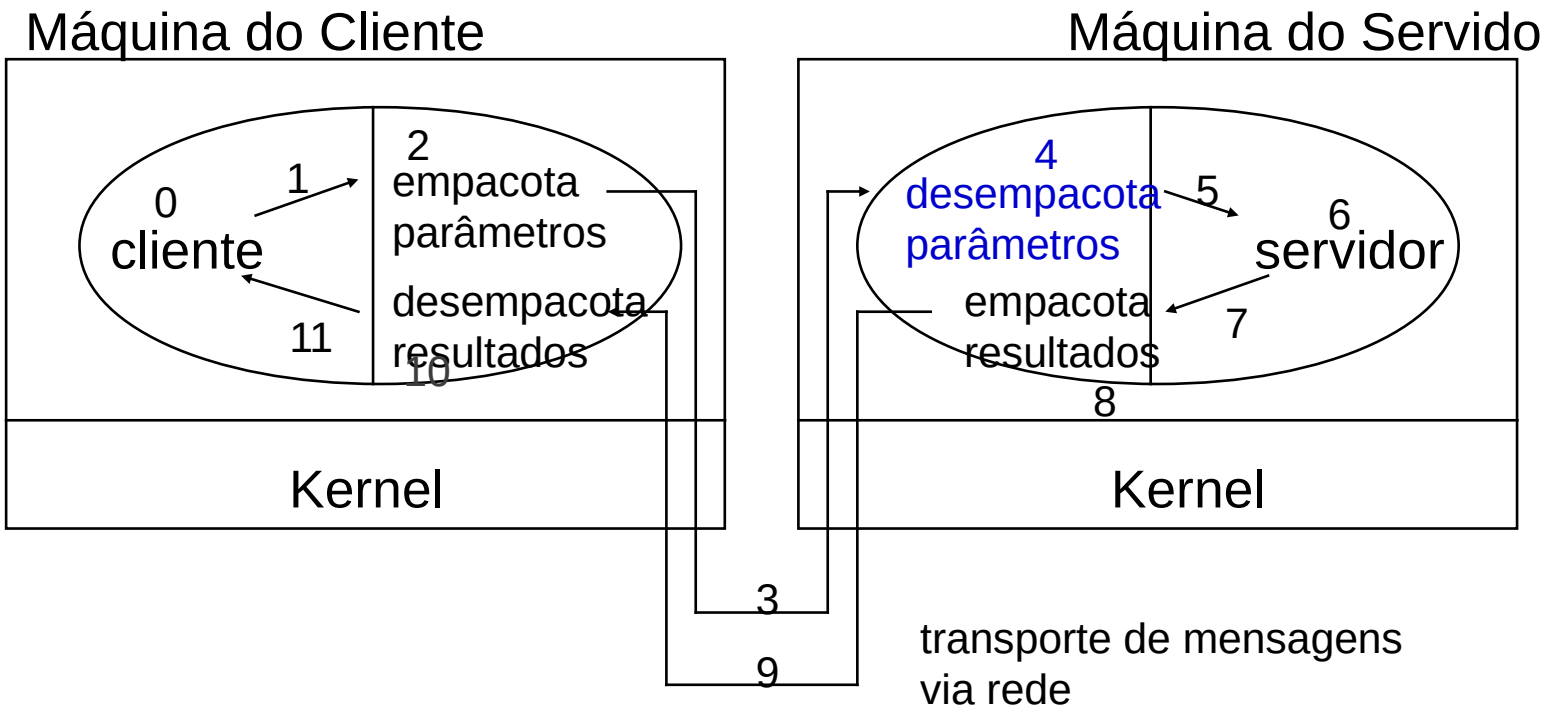
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

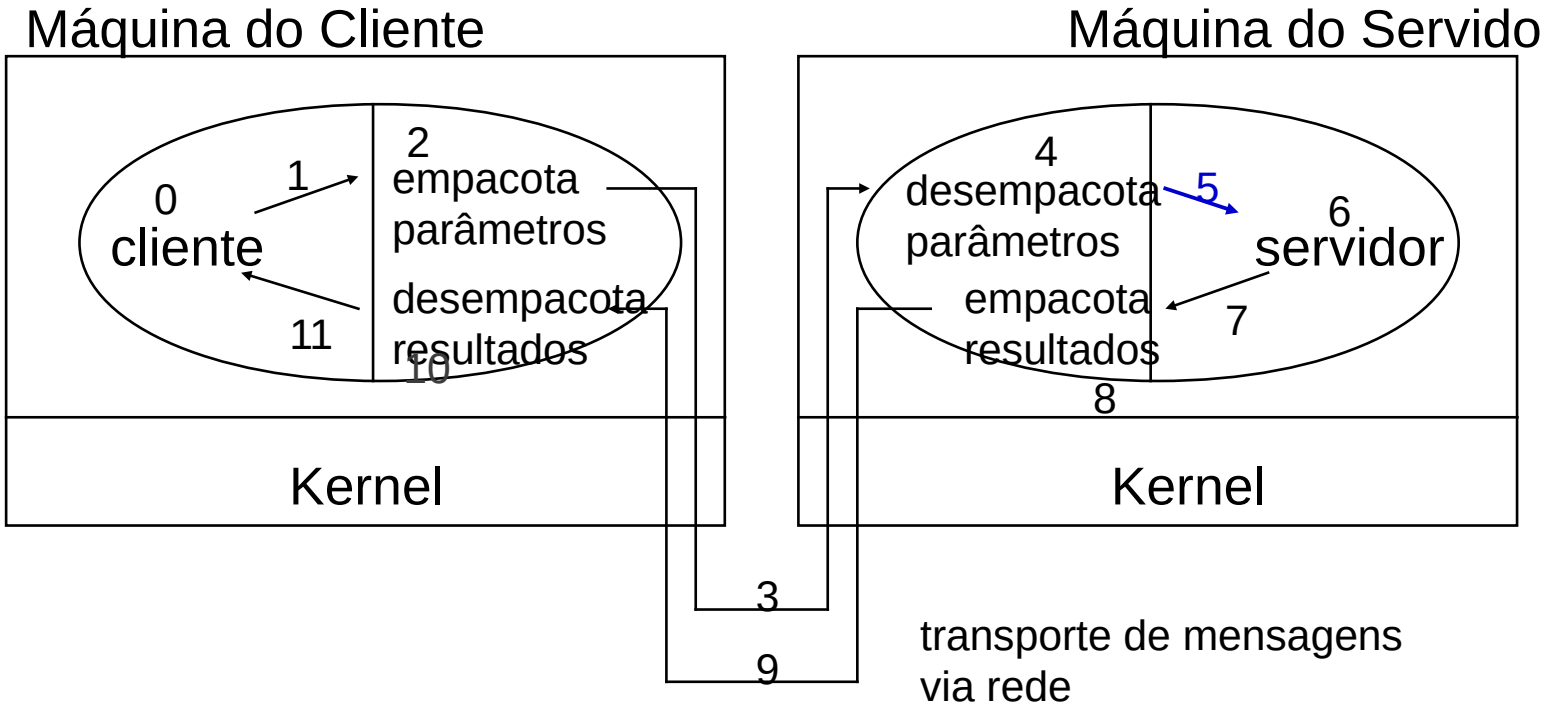
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

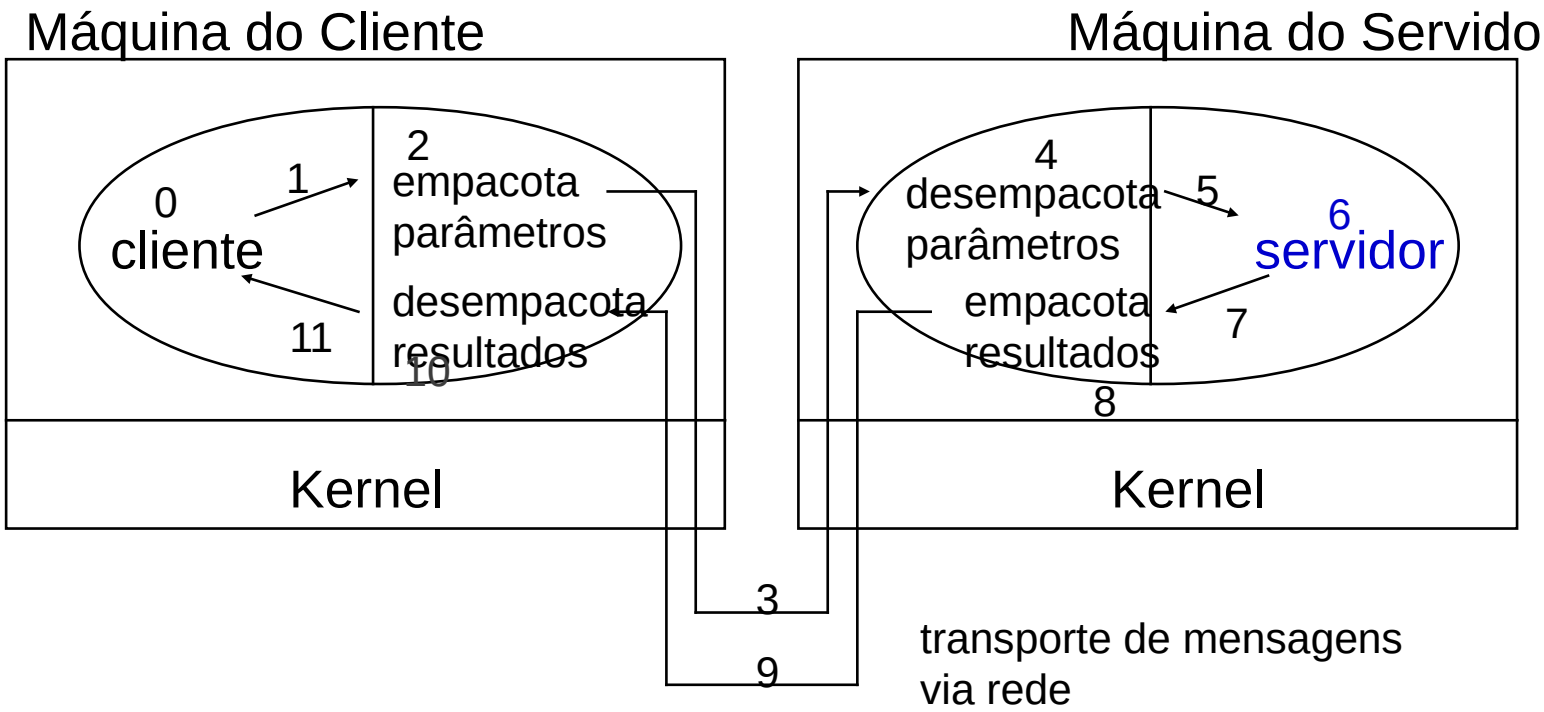
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

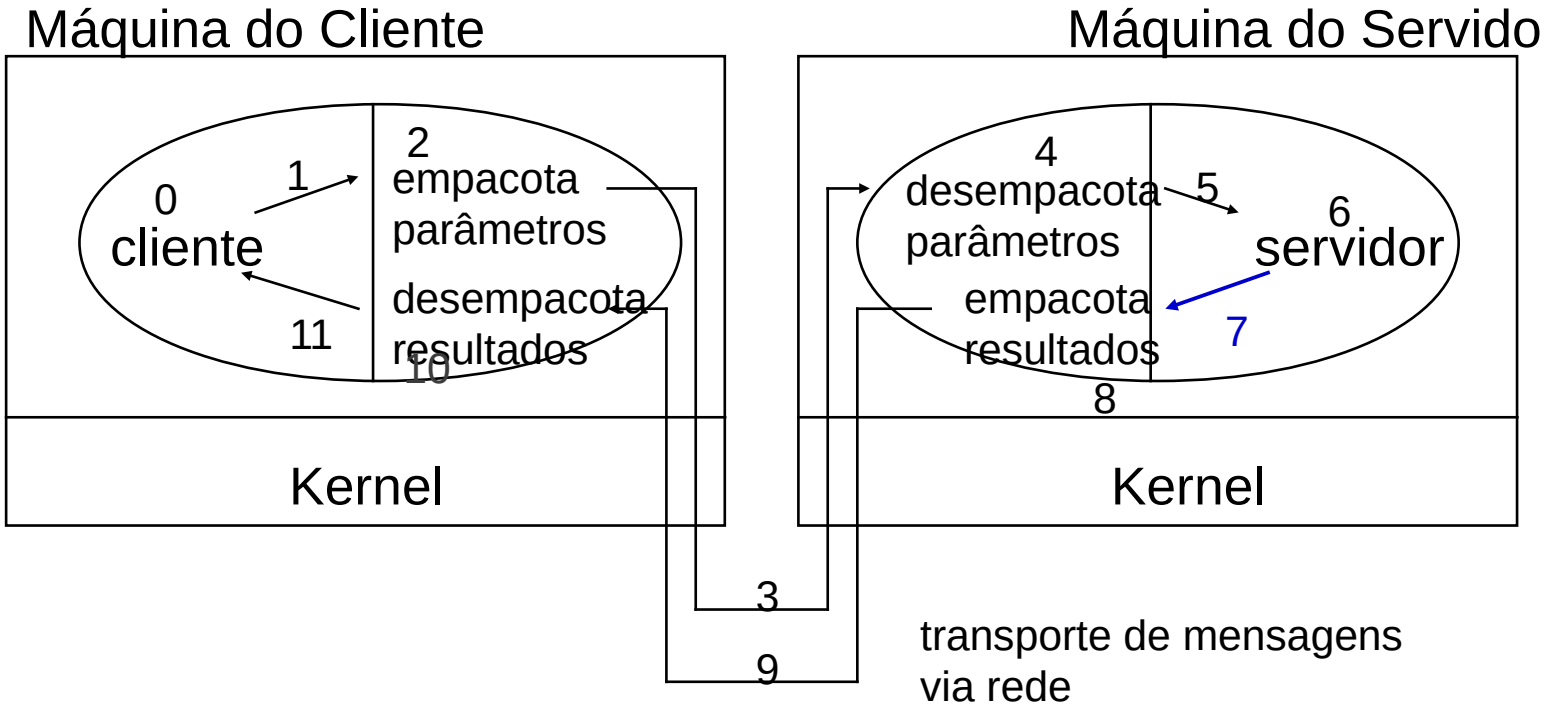
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

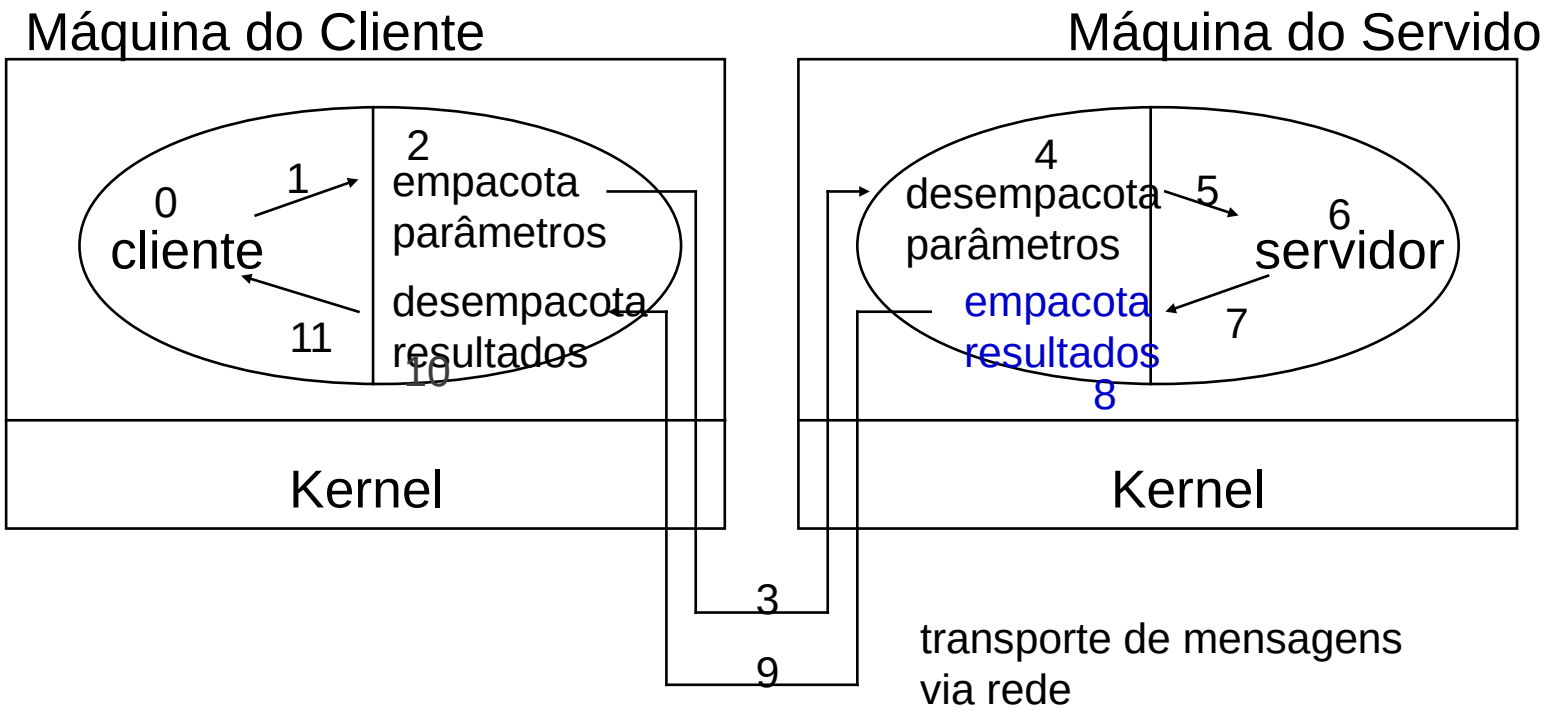
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

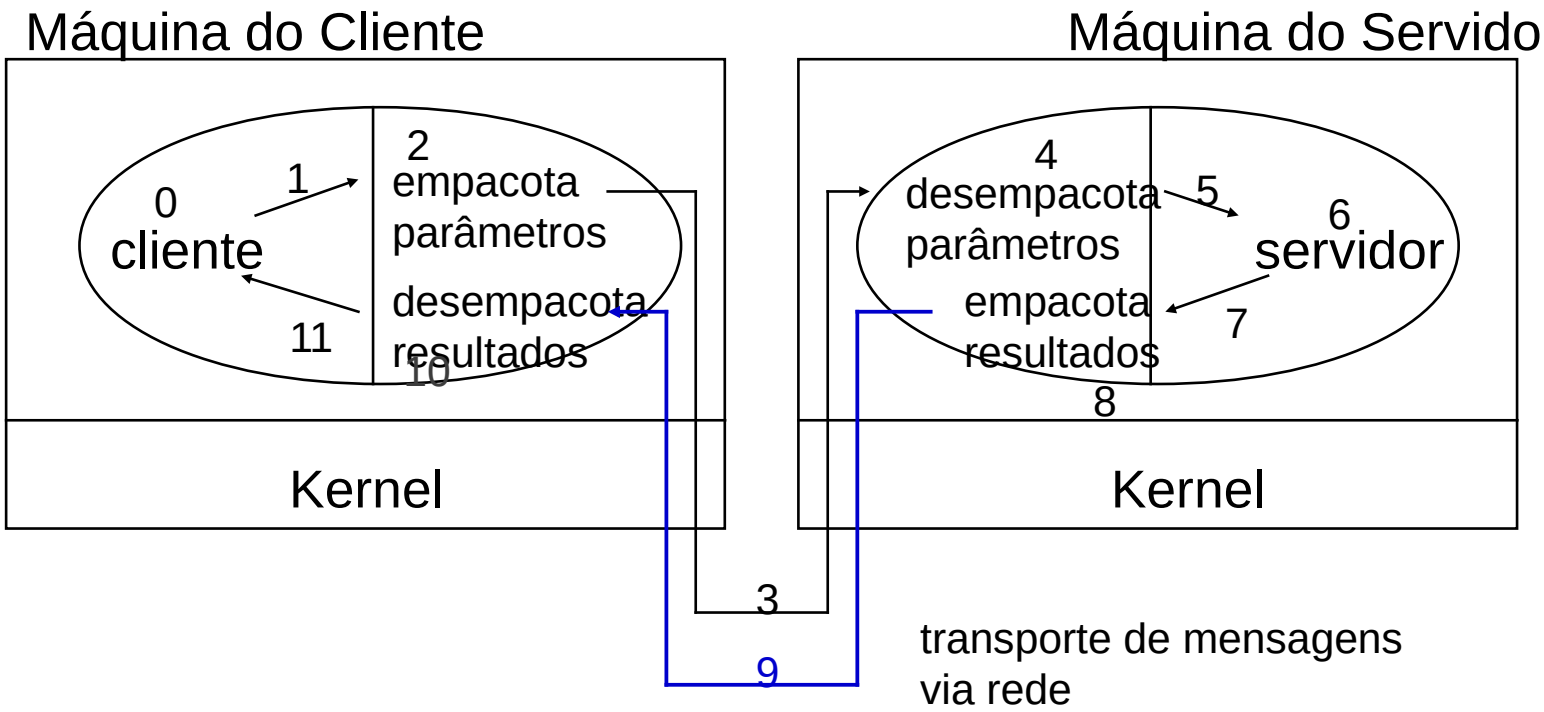
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

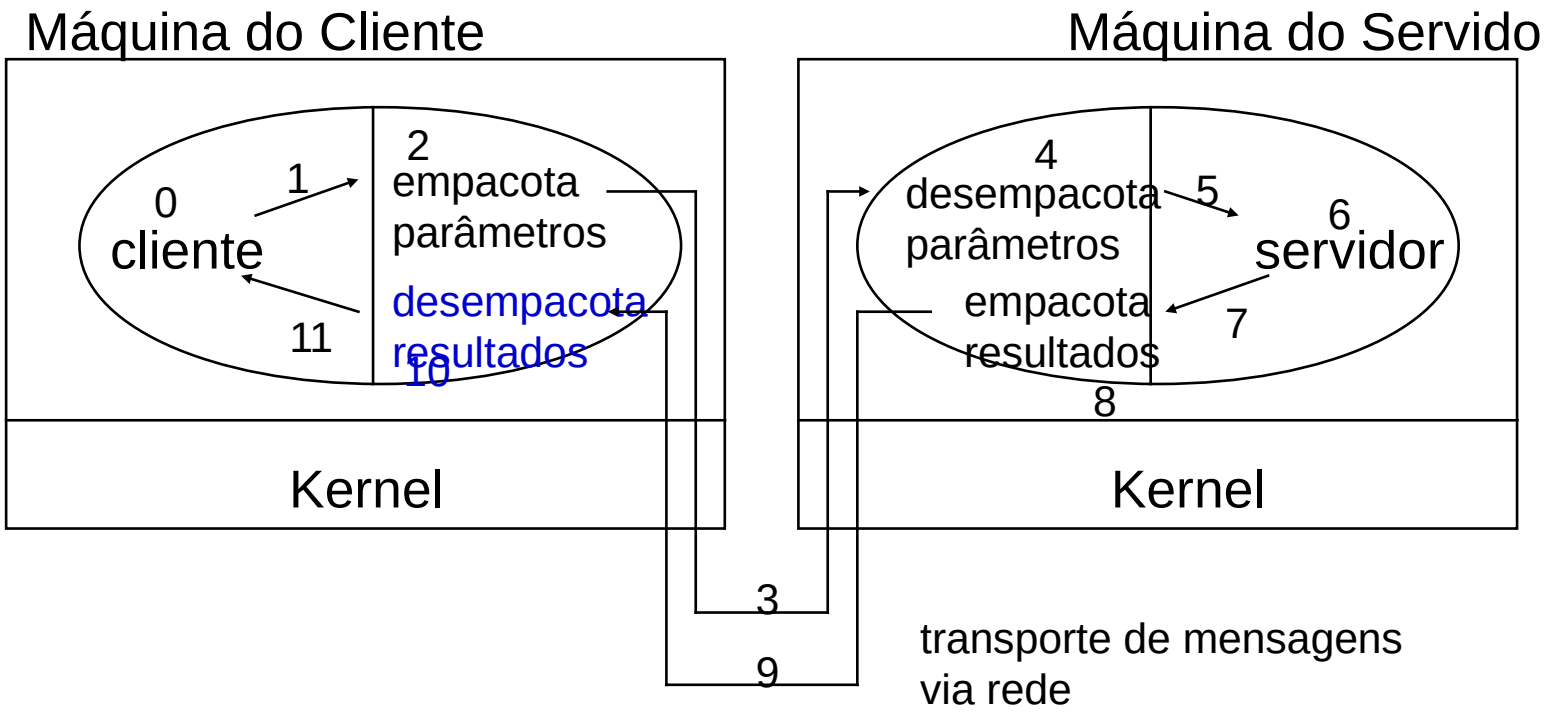
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

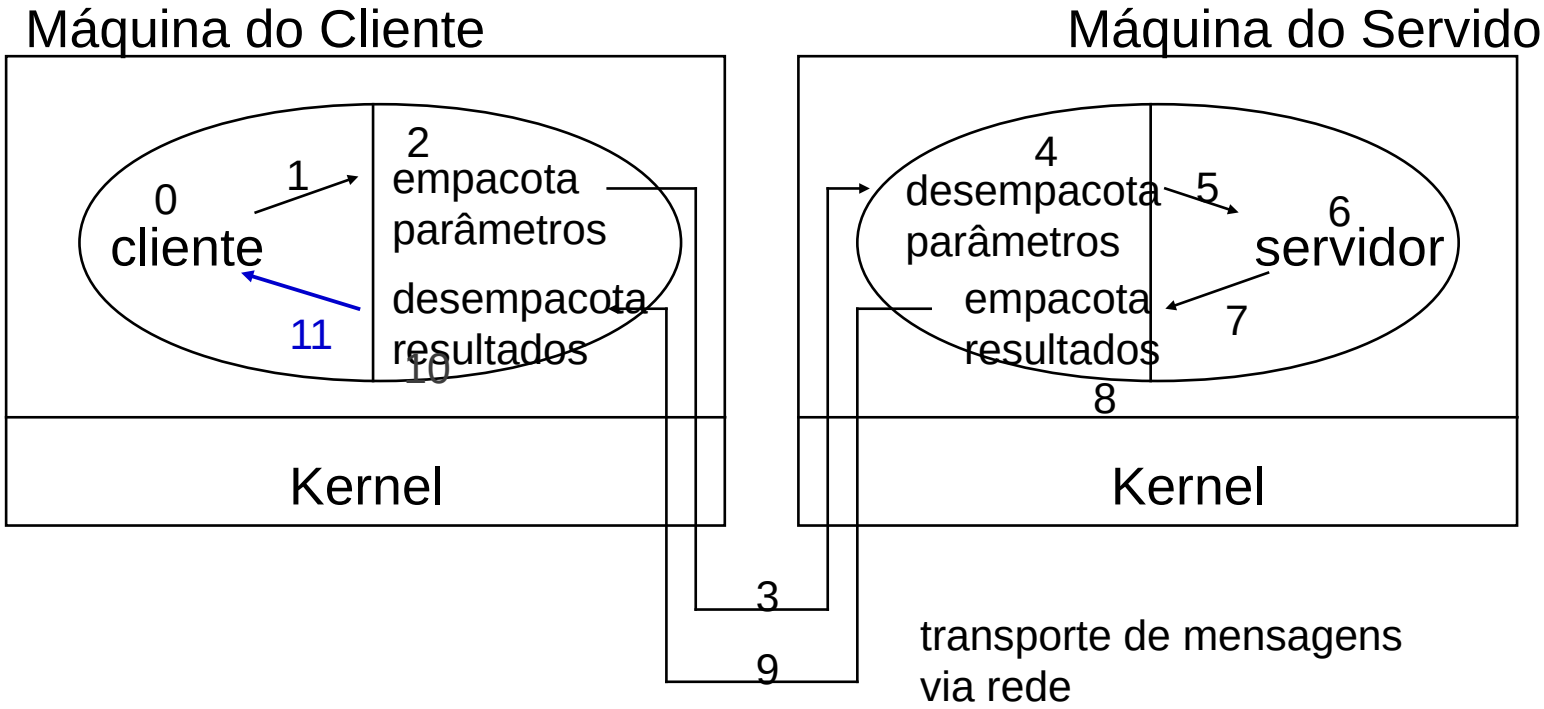
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Intermediário

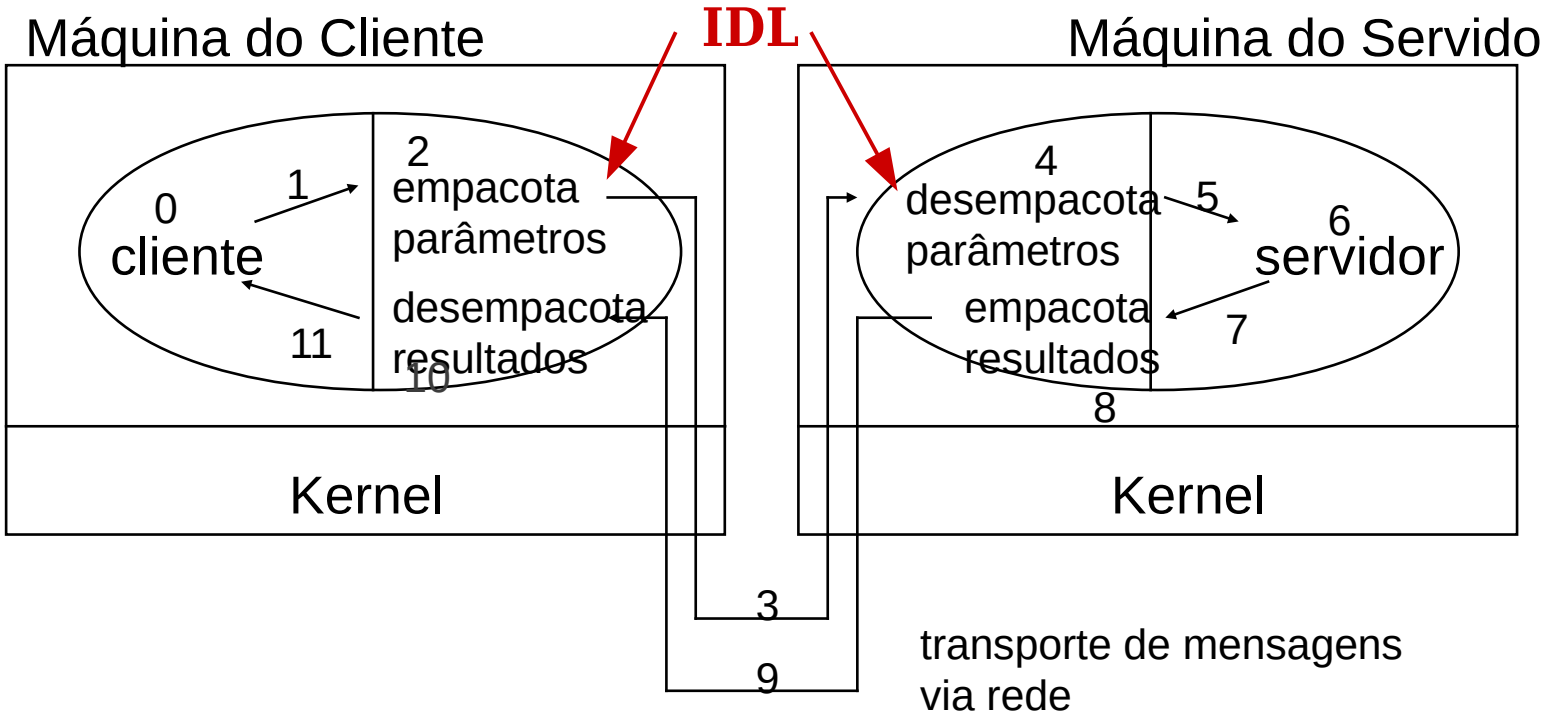
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Sistemas Distribuídos: RPC Avançado

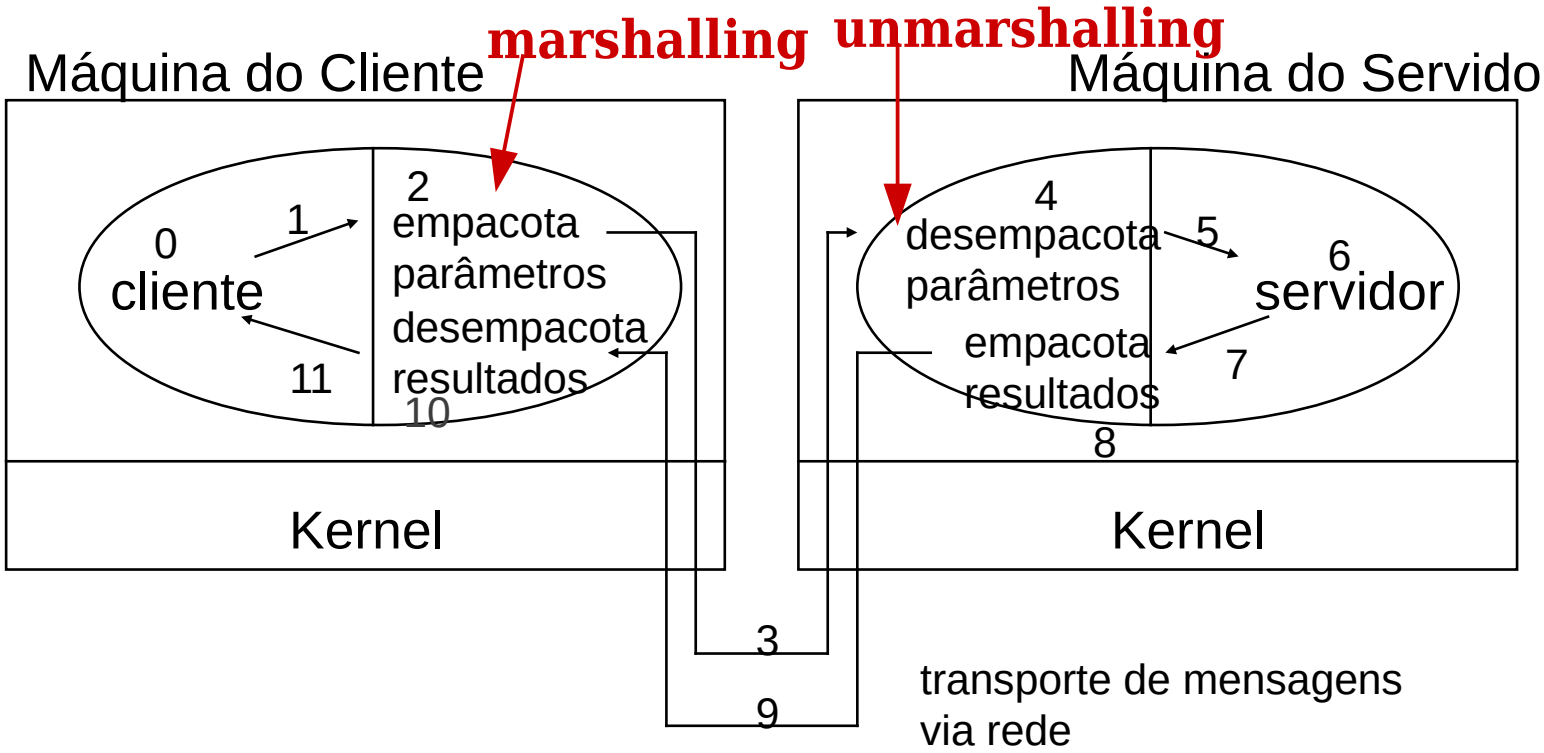
Chamadas e mensagens em RPC



Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

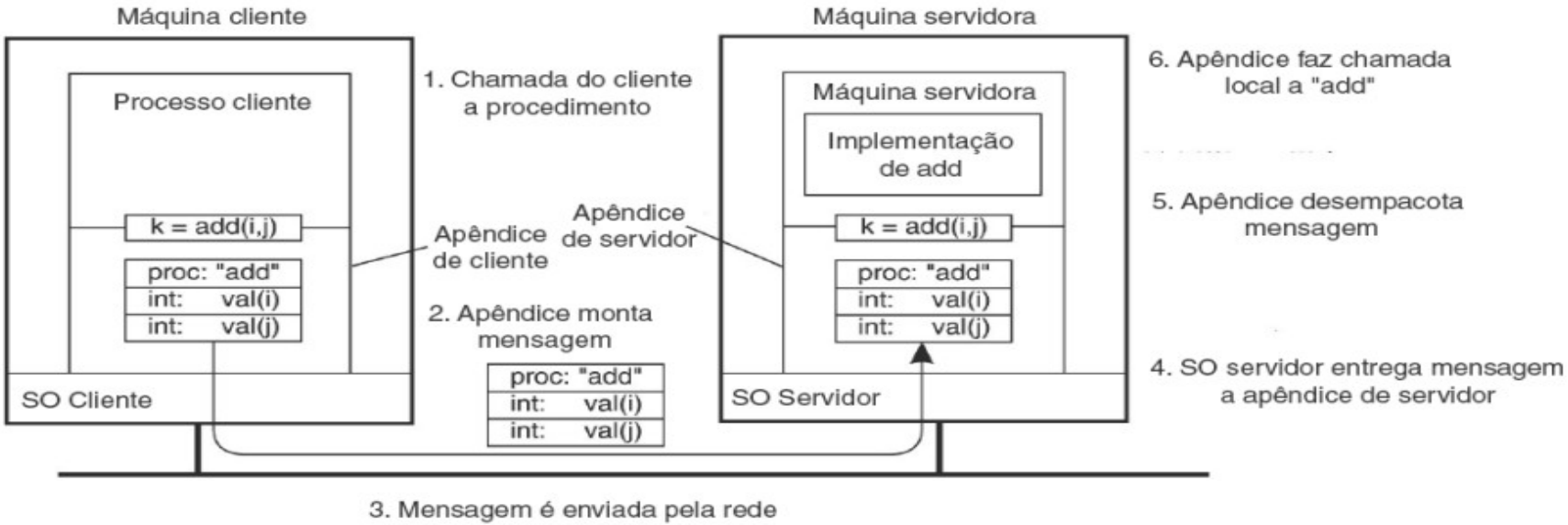
Sistemas Distribuídos: RPC Avançado

Chamadas e mensagens em RPC



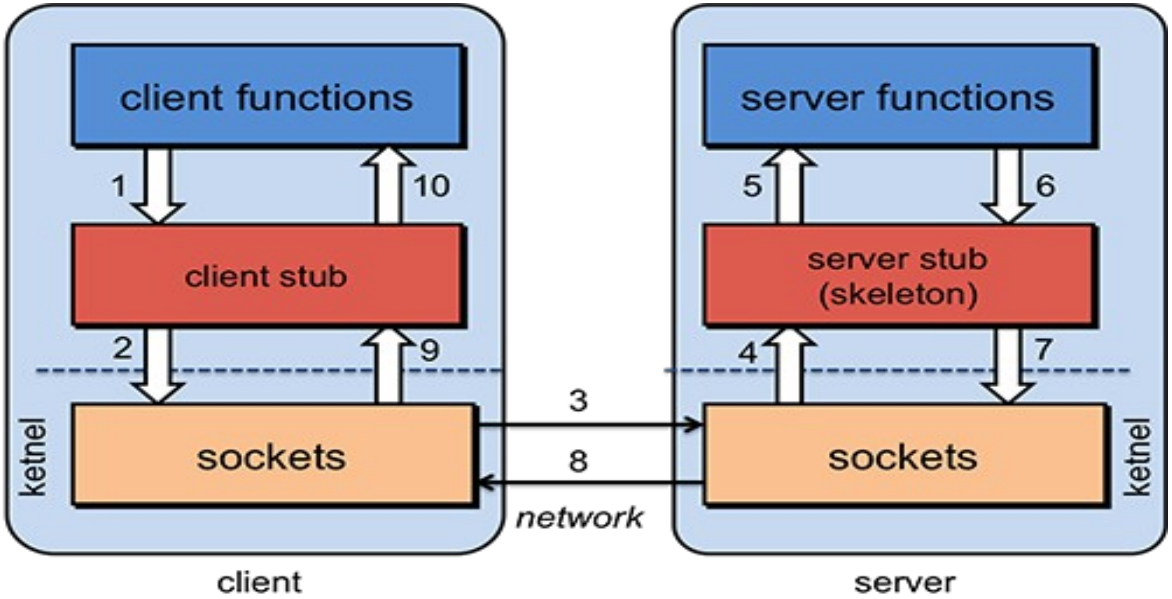
Fonte: Chamadas Remotas de Procedimentos, 2017. Disponível em: https://dimap.ufrn.br/~thais/SD20071/RPC-RMI.pdf Acesso em: 01 de Agosto 2018

Chamadas e mensagens em RPC



Fonte: Andrew S. Tanembaum, Sistemas Distribuídos, Princípios e Paradigmas 2ª edição, p. 78

Chamadas e mensagens em RPC



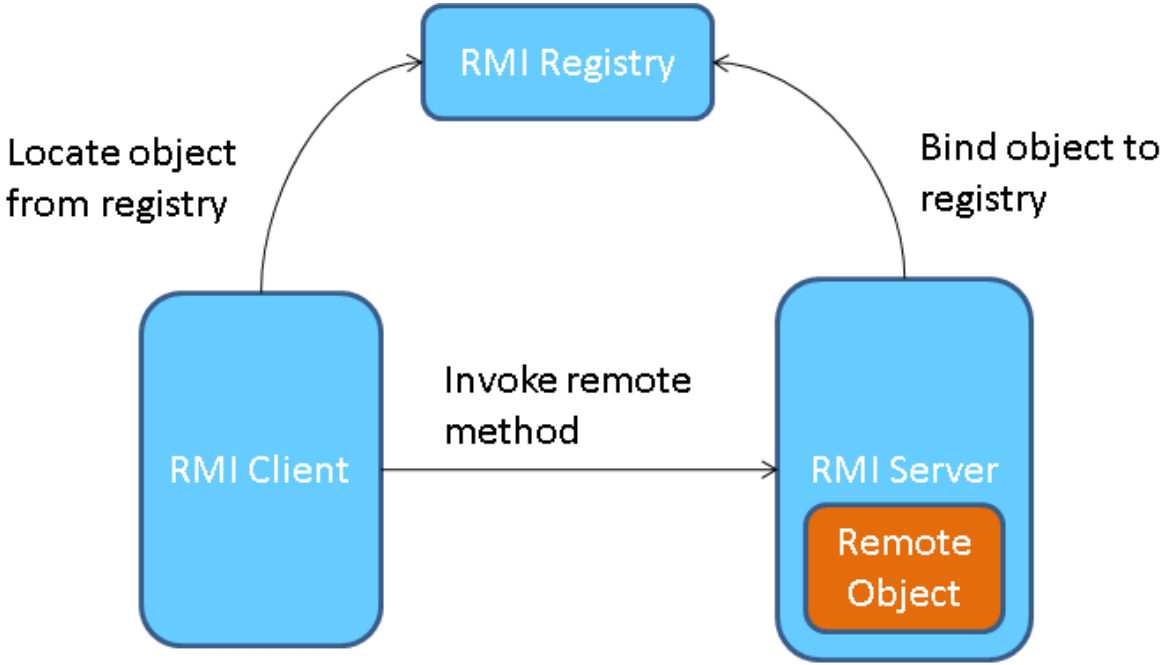
Fonte: Remote Procedure Calls, 2012. Disponível em: https://www.cs.rutgers.edu/~pxk/417/notes/08-rpc.html Acesso em: 01 de Agosto 2018

Funções dos Stubs

- Client stub
  - intercepta a chamada
  - empacota os parâmetros (marshalling)
  - envia mensagem de request ao servidor (através do núcleo)
- Server stub
  - recebe a mensagem de request (através do núcleo)
  - desempacota os parâmetros (unmarshalling)
  - chama o procedimento, passando os parâmetros
  - empacota o resultado
  - envia mensagem de reply ao cliente (através do núcleo)
- Client stub
  - recebe a mensagem de reply (através do núcleo)
  - desempacota o resultado
  - passa o resultado para o cliente

- **Elementos arquitetônicos: Paradigmas de comunicação**
  - **Invocação de método remoto (RMI, Remote Method Invocation):** similar ao RPC, mas voltado para trabalhar com objetos distribuídos.
    - Os detalhes subjacentes ficam ocultos do usuário
    - Possibilita:
      - verificar a identidade de objetos
      - passagem dos identificadores de objeto como parâmetros em chamadas remotas

- **Elementos arquitetônicos: Paradigmas de comunicação**
  - **Invocação de método remoto (RMI, Remote Method Invocation):**

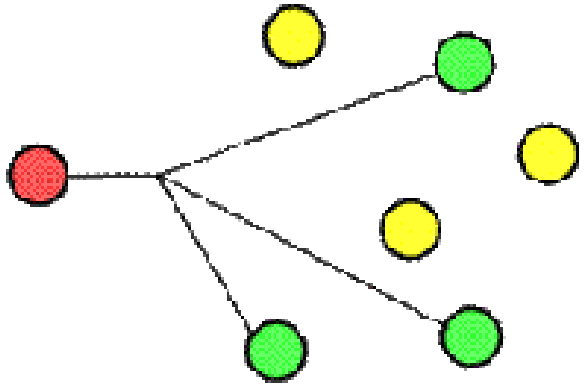


Introdução

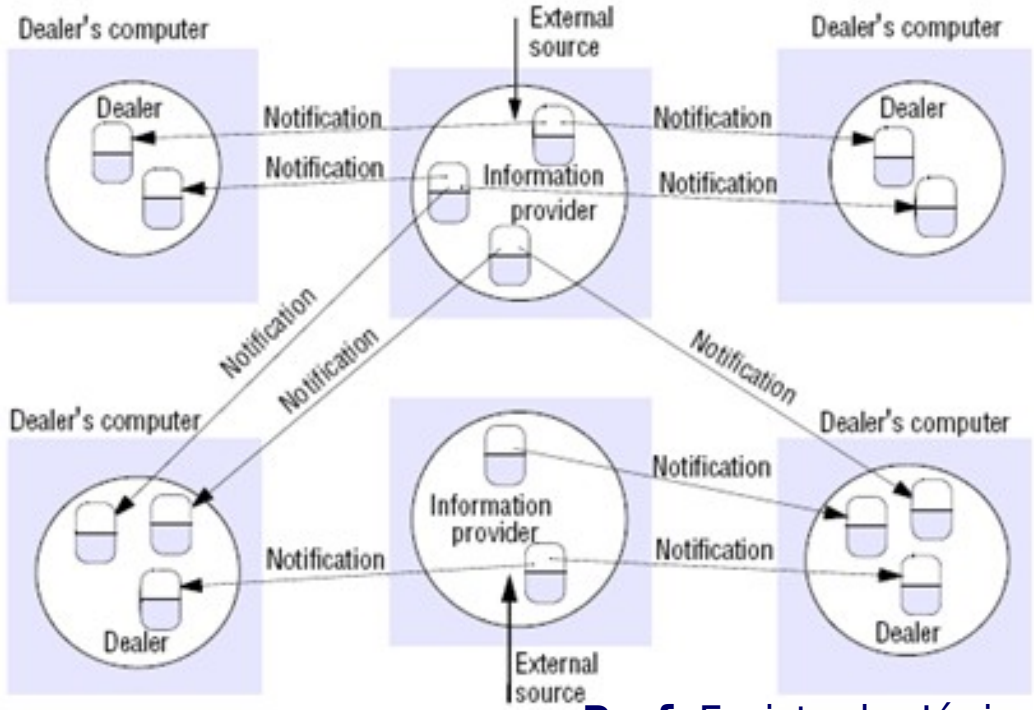
Qual a característica comum entre os paradigmas vistos?

- **Elementos arquitetônicos: Paradigmas de comunicação**
  - **Característica comum** entre os envolvidos:
    - Relação bilateral entre um remetente e um destinatário (**troca de mensagens direta**)
  - Técnicas com abordagens diferentes (comunicação indireta):
    - Os remetentes não precisam saber para quem estão enviando **(desacoplamento espacial)**.
    - Os remetentes e os destinatários não precisam existir ao mesmo tempo **(desacoplamento temporal)**.

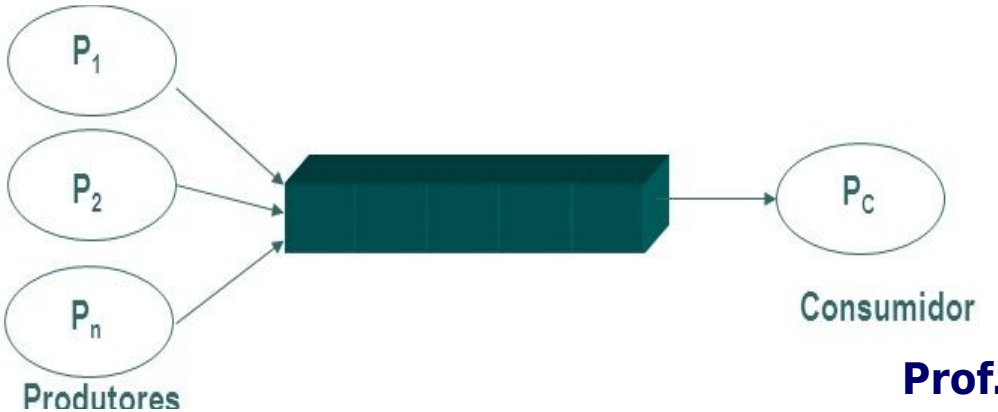
- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Principais técnicas de comunicação indireta:
    - **Comunicação em grupo:** a comunicação em grupo está relacionada à entrega de mensagens para um conjunto de destinatários (Paradigma um para muitos)



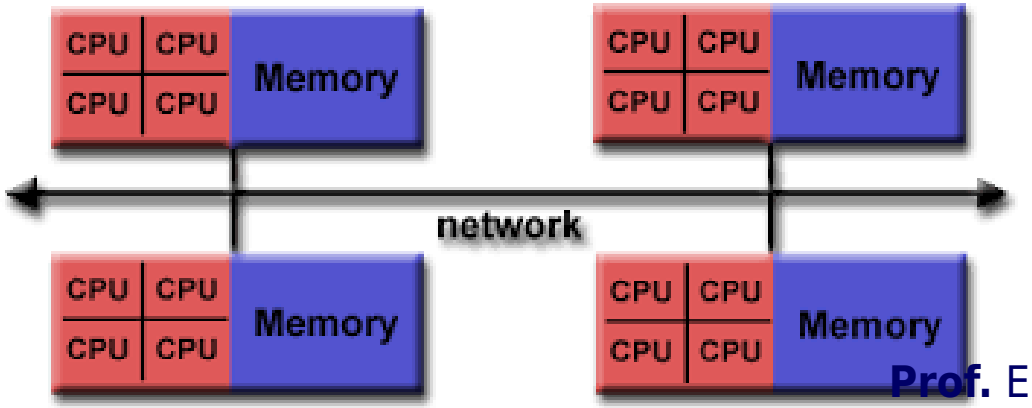
- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Principais técnicas de comunicação indireta:
    - **Sistemas publicar-assinar**:
      - Disseminação de informações
      - produtores (ou publicadores) x consumidores (ou assinantes



- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Principais técnicas de comunicação indireta:
    - **Filas de mensagem**:
      - Oferecem um serviço ponto a ponto
      - Processos produtores enviam mensagens para uma fila especificada
      - Processos consumidores recebem mensagens da fila ou são notificados da chegada de novas mensagens na fila



- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Principais técnicas de comunicação indireta:
    - **Memória compartilhada distribuída:**
      - os sistemas de memória compartilhada distribuída (**DSM, Distributed Shared Memory**) fornecem uma abstração para compartilhamento de dados entre processos que não compartilham a memória física.

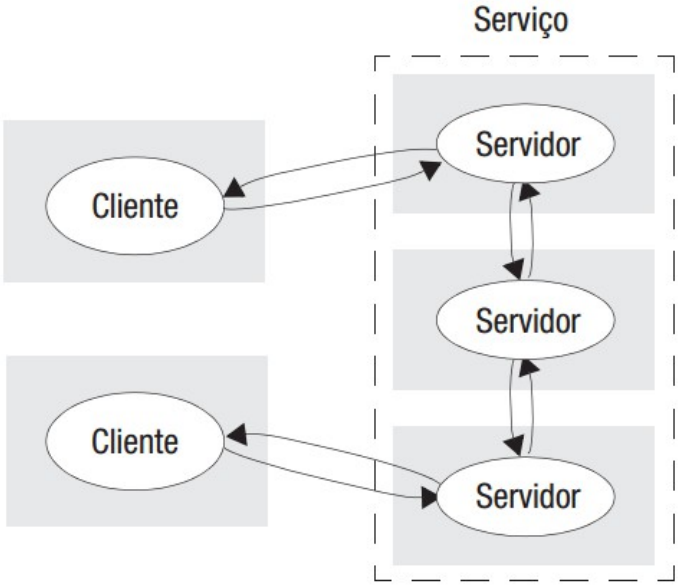


- **Elementos arquitetônicos: Paradigmas de comunicação**
  - Resumindo...

Entidades em comunicação (o que se comunica)		Paradigmas de comunicação (como se comunicam)		
Orientados a sistemas	Orientados a problemas	Entre processos	Invocação remota	Comunicação indireta
Nós	Objetos	Passagem de mensagem	Requisição-resposta	Comunicação em grupo
Processos	Componentes	Soquetes	RPC	Publicar-assinar
	Serviços Web	Multicast	RMI	Fila de mensagem
				DSM

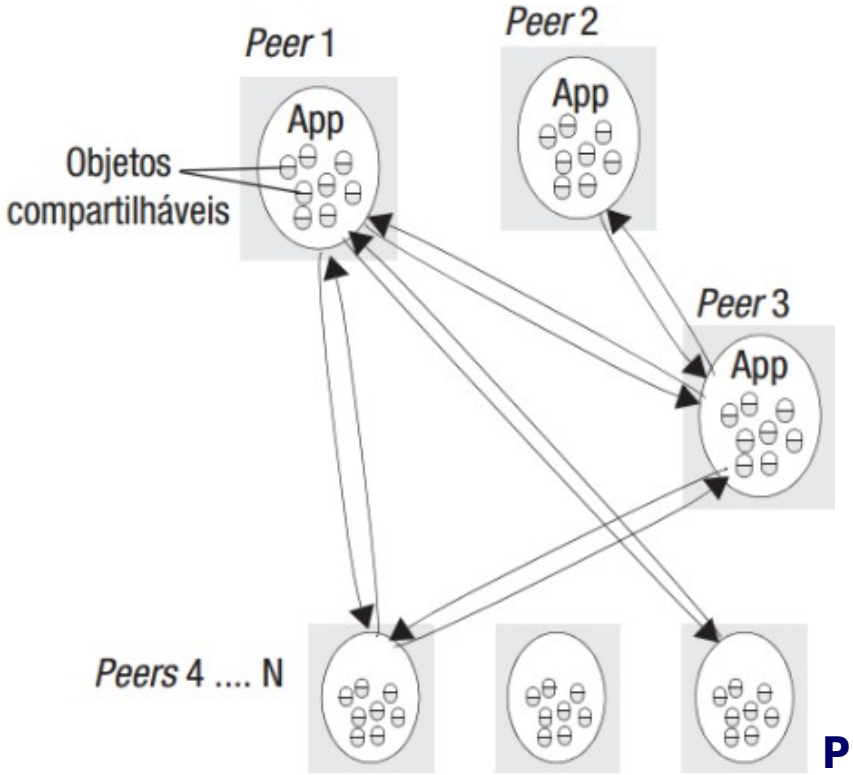
- **Elementos arquitetônicos: Funções e Responsabilidades**
  - Os processos nos sistemas assumem **determinadas responsabilidades relacionadas ao sistema onde estão inseridos**, proporcionando a especificação de uma arquitetura global a ser adotada.
  - Estilos de arquitetura básicos:
    - **Cliente servidor**
    - **Peer-to-peer**
  - Como funcionam essas arquiteturas básicas?

- **Elementos arquitetônicos: Funções e Responsabilidades**
  - **Cliente-servidor**:
    - os processos clientes interagem com **processos servidores**, localizados possivelmente em distintos computadores hospedeiros, para **acessar os recursos compartilhados que estes gerenciam**



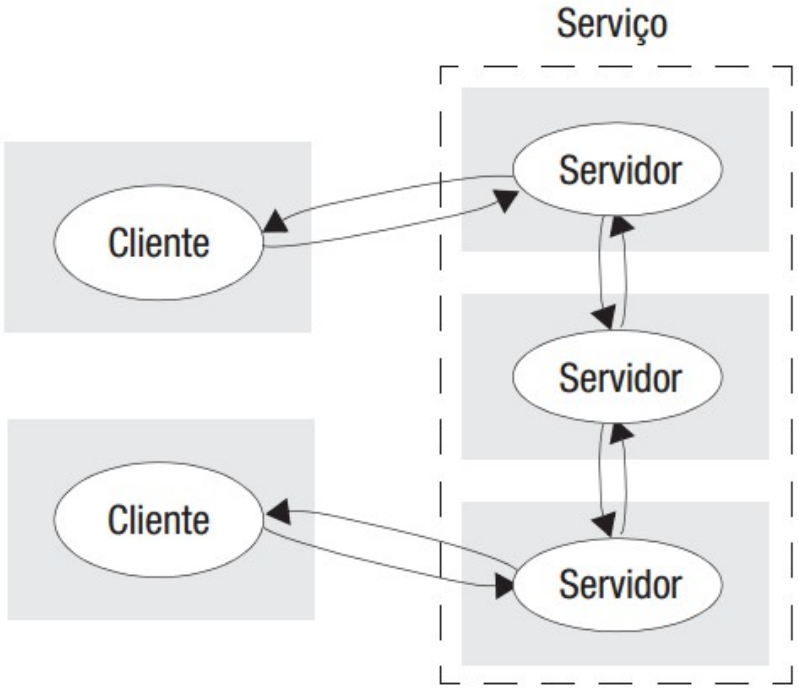
- **Elementos arquitetônicos: Funções e Responsabilidades**
  - **Peer-to-peer**: **todos os processos envolvidos** em uma tarefa ou atividade **desempenham funções semelhantes**, interagindo cooperativamente como pares
    - Ideia principal: a **rede e os recursos computacionais** pertencentes aos usuários de um serviço também poderiam ser utilizados para suportar esse serviço
    - **Objetivo**: é explorar os recursos (tanto dados como de hardware) de um grande número de computadores para o cumprimento de uma dada tarefa ou atividade

- **Elementos arquitetônicos: Funções e Responsabilidades**
  - **Peer-to-peer**:

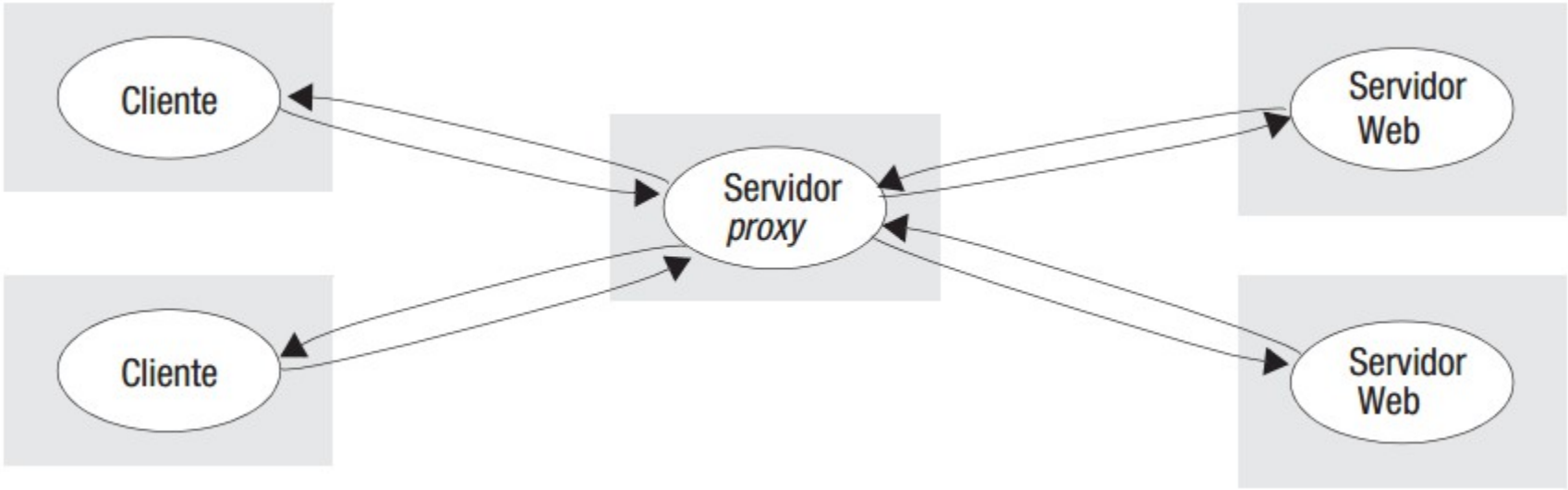


- **Elementos arquitetônicos: Posicionamento**
  - Define como objetos ou serviços são **mapeadas na infraestrutura física distribuída subjacente**
  - O posicionamento precisa levar em conta os **padrões de comunicação entre as entidades, a confiabilidade de determinadas máquinas e sua carga atual**, a qualidade da comunicação entre as diferentes máquinas, etc
  - Estratégias:
    - mapeamento de serviços em vários servidores;
    - uso de cache;
    - código móvel;
    - agentes móveis

- **Elementos arquitetônicos: Posicionamento**
  - **Mapeamento de serviços em vários servidores**:

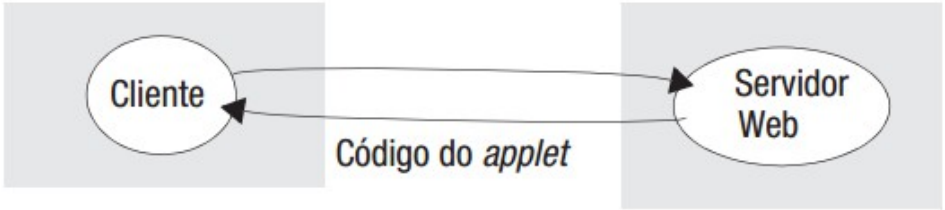


- **Elementos arquitetônicos: Posicionamento**
  - **Uso de Cache**:



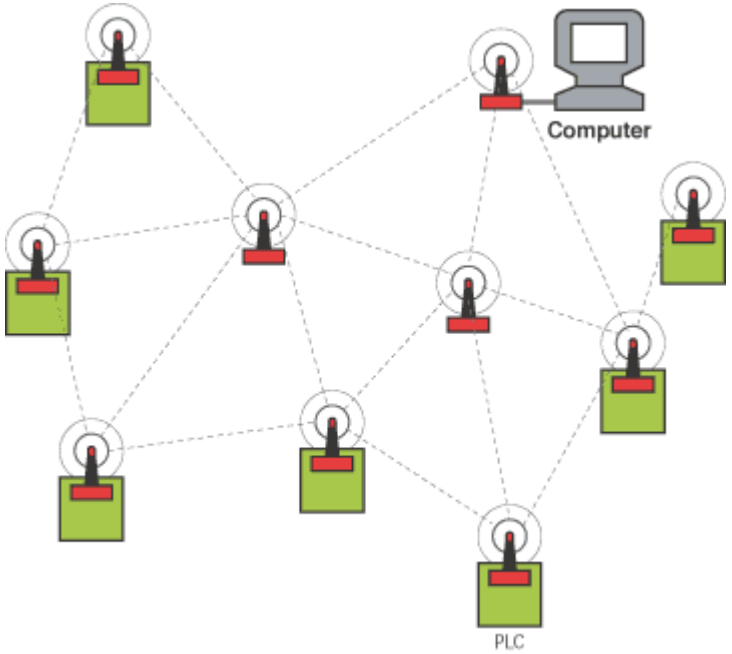
- **Elementos arquitetônicos: Posicionamento**
  - **Código Móvel:**
    - Uma vantagem de executar um código localmente é que ele pode dar uma **boa resposta interativa**

a) Requisição do cliente resulta no download do código de um *applet*



b) O cliente interage com o *applet*

- **Elementos arquitetônicos: Posicionamento**
  - **Agente Móvel:**
    - Programa em execução (inclui código e dados) que **passa de um computador para outro em um ambiente de rede**, realizando uma **tarefa em nome de alguém**, como uma coleta de informações, e finalmente retornando com os resultados obtidos a esse alguém



Exercícios

1. Dê três exemplos específicos e contrastantes dos níveis de heterogeneidade cada vez maiores experimentados nos sistemas distribuídos atuais, conforme definido na Seção 2.2
2. Descreva e ilustre a arquitetura cliente-servidor de um ou mais aplicativos de Internet importantes (por exemplo, Web, correio eletrônico ou news).
3. Para os aplicativos discutidos no primeiro exercício, quais estratégias de posicionamento são empregadas na implementação dos serviços associados?

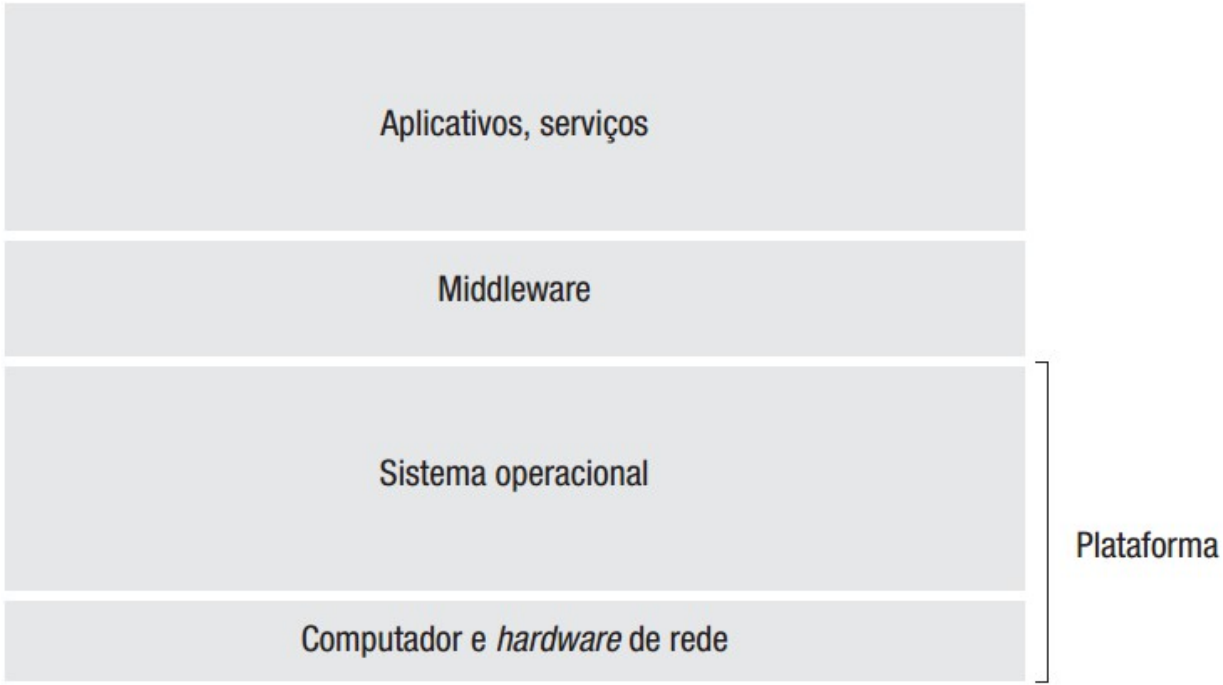
O que seriam os padrões arquitetônicos?

- Os padrões arquitetônicos baseiam-se nos **elementos de arquitetura mais primitivos** e fornecem estruturas recorrentes compostas que mostraram bom funcionamento em determinadas circunstâncias.
- Os seguintes padrões arquitetônicos serão vistos:
  - Arquitetura de camadas lógicas
  - Arquitetura de camadas físicas
  - Clientes “magros”(thin)

Qual seria um exemplo clássico de arquitetura de camadas lógicas?

- **Arquitetura de Camadas Lógicas**:
  - Sistema complexo é **particionado em várias camadas**, com cada uma utilizando os serviços oferecidos pela camada lógica inferior.
  - Nesse padrão, dois termos são importantes: **plataforma e middleware**. Quais são as suas definições?

- **Arquitetura de Camadas Lógicas**:



- **Arquitetura de Camadas Lógicas**:
  - **Plataforma**:
    - Uma plataforma para sistemas e aplicativos distribuídos consiste nas **camadas lógicas de hardware e software de nível mais baixo.**
    - Essas camadas lógicas de baixo nível **fornecem serviços para as camadas que estão acima delas, as quais são implementadas independentemente** em cada computador.
    - Ex: Intel x86/Windows

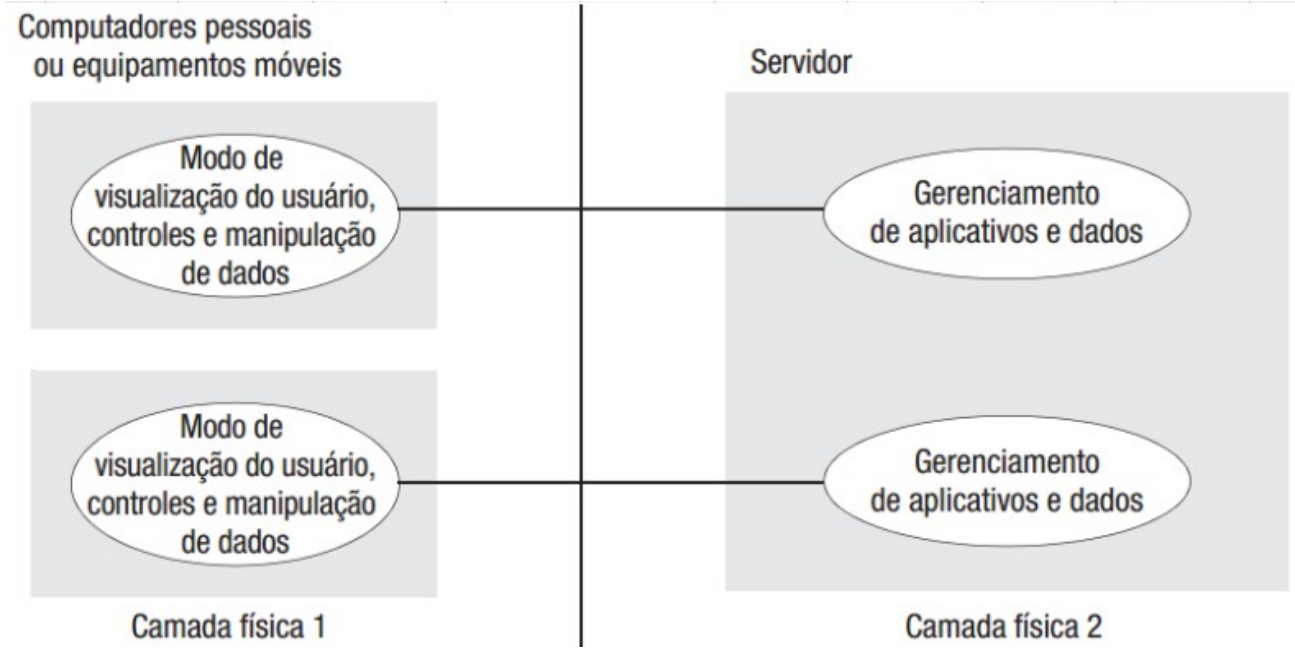
Relembrando, o que seria um middleware?

- **Arquitetura de Camadas Lógicas**:
  - **Middleware**:
    - Camada de software cujo objetivo é **mascarar a heterogeneidade e fornecer um modelo de programação** conveniente para os programadores de aplicativo.
    - É **representado por processos ou objetos** em um conjunto de computadores que **interagem entre si** para implementar o suporte para **comunicação e compartilhamento de recursos** para sistemas distribuídos

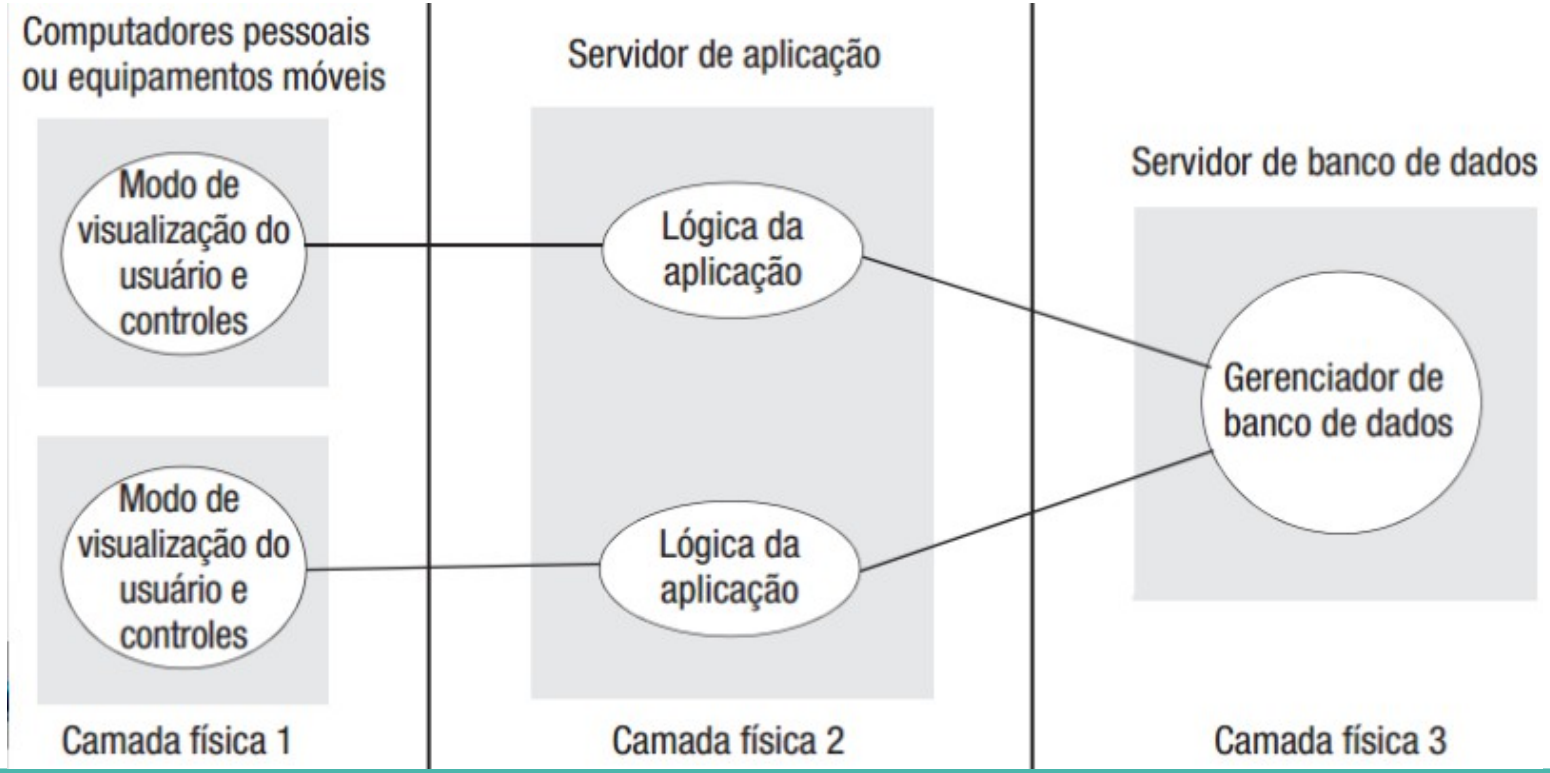
- **Arquitetura de Camadas Lógicas**:
  - **Middleware**:
    - Limitação:
      - A confiabilidade dos sistemas exige suporte em nível de aplicação
        - o problema de um usuário que tenta transferir um arquivo muito grande por meio de uma rede potencialmente não confiável. O protocolo TCP fornece certa capacidade de detecção e correção de erros, mas não consegue se recuperar de interrupções mais sérias na rede

Como seria a arquitetura em camadas físicas?

- **Arquitetura de Camadas Físicas**:
  - Representam uma técnica para **organizar a funcionalidade de determinada camada lógica e colocar essa funcionalidade nos servidores apropriados**



- **Arquitetura de Camadas Físicas**:



Quais as vantagens e desvantagens do processo ser particionado no cliente e no servidor?

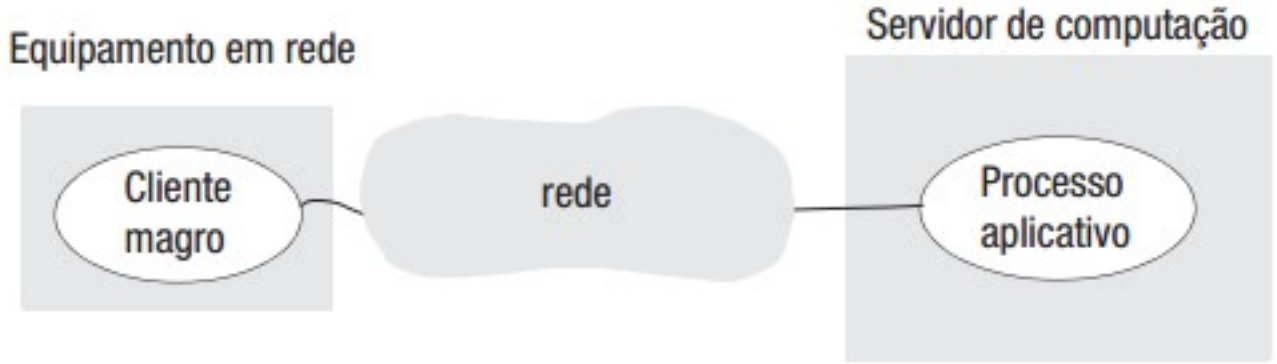
- **Arquitetura de Camadas Físicas**:
  - **Vantagem**:
    - **Baixas latências** em termos de interação, com apenas uma troca de mensagens para ativar uma operação.
  - **Desvantagem**
    - Divisão da lógica da aplicação entre limites de processo, com a consequente **restrição sobre quais partes podem ser chamadas diretamente de quais outras partes**

Quais as restrições para o desenvolvimento de aplicativos com base no padrão de interação padrão da Internet?

- **Arquitetura de Camadas Físicas**:
  - Restrições do padrão de interação básico no desenvolvimento de aplicativos Web:
    - Uma vez que o navegador tenha feito um **pedido HTTP** para uma nova página Web, o **usuário não pode interagir com ela** até que o novo conteúdo HTML seja recebido e apresentado pelo navegador.
    - Para **atualizar mesmo uma pequena parte da página atual com dados adicionais do servidor**, uma nova página inteira precisa ser solicitada e exibida.
    - O conteúdo de uma página exibida em um **cliente não pode ser atualizado em resposta a alterações feitas nos dados do aplicativo mantidos no servidor**.

O que você entende por “clientes magros”?

- **Arquitetura de Camadas Físicas**:
  - **Clientes magros**:
    - **Tendência**: retirar a complexidade do equipamento do usuário final e passá-la para os serviços da Internet
    - **Cliente magro(thin)**: acesso a sofisticados serviços interligados em rede, fornecidos, por exemplo, por uma solução em nuvem, com poucas suposições ou exigências para o equipamento cliente.



Quais as vantagens e desvantagens em se utilizar “clientes magros”?

- **Arquitetura de Camadas Físicas**:
  - **Clientes magros**:
    - **Vantagem:**
      - Um equipamento local potencialmente simples (incluindo, por exemplo, smartphones e outros equipamentos com poucos recursos) pode ser melhorado significativamente com diversos serviços e recursos interligados em rede.
    - **Desvantagem**:
      - O principal inconveniente da arquitetura de cliente magro aparece em **atividades gráficas altamente interativas, como CAD e processamento de imagens**

- **Arquitetura de Camadas Físicas**:
  - **Outros padrões**:
    - **Padrão proxy**:
      - Recorrente em sistemas distribuídos projetados especificamente para suportar **transparência de localização em chamadas de procedimento remoto ou invocação de método remoto**

```java
import java.util.ArrayList;
import java.util.List;

/**
 * Imagine que esta classe faz acessos ao banco de dados
 */
class PessoaDAO {
	public static Pessoa getPessoaByID(String id){
		System.out.println("select * from PESSOA where id="+id);
		return new PessoaImpl(id,"Pessoa " + id);
	}
}

/**
 * Interface comum entre o objeto real e o Proxy
 */
interface Pessoa {
	public String getNome();
	public String getId();
}

/**
 * Objeto real
 */
```

```
25 class PessoaImpl implements Pessoa {
26       private String nome;
27       private String id;
28 
29       public PessoaImpl(String id,String nome) {
30             this.id       = id;
31             this.nome = nome;
32             // apenas para simular algo...
33             System.out.println("Retornou a pessoa do banco de dados ->  " + nome);
34       }
35 
36       public String getNome() {
37             return nome;
38       }
39       public String getId() {
40             return this.id;
41       }
42 }
```

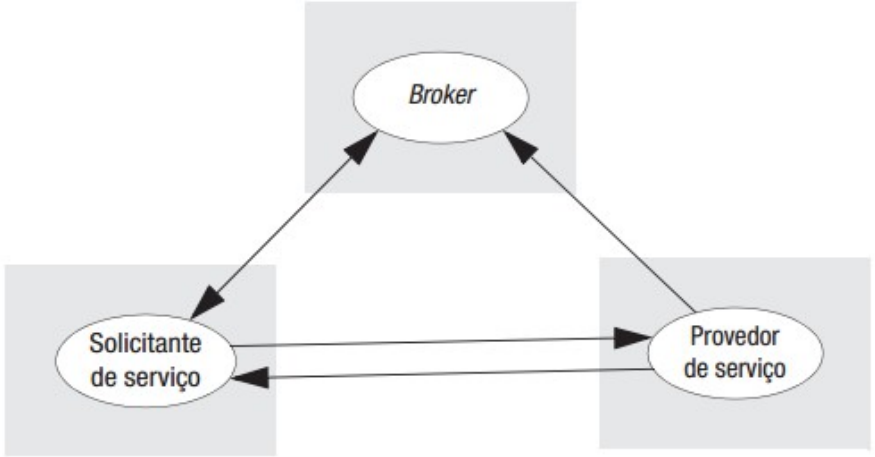
```
44 class ProxyPessoa implements Pessoa {
45       private String id;
46
47       private Pessoa pessoa;//mesma interface
48
49       public ProxyPessoa(String nome) {
50             this.id = nome;
51       }
52
53       /**
54        * Método comum da interface
55        *
56        * @see Pessoa#getNome()
57        */
58       public String getNome() {
59             if (pessoa == null) {
60                   //Apenas cria o objeto real quando chamar este método
61                   pessoa = PessoaDAO.getPessoaByID(this.id);
62             }
63             /** Delega para o objeto real **/
64             return pessoa.getNome();
65       }
66
67       public String getId() {
68             return this.id;
69       }
70 }
```

```java
public class ProxyExample {
      public static void main(String[] args) {
            List<Pessoa> pessoas = new ArrayList<Pessoa>();

      //Cria cada Proxy para encapsular o acesso a classe "PessoaImpl"
            pessoas.add(new ProxyPessoa("A"));
            pessoas.add(new ProxyPessoa("B"));
            pessoas.add(new ProxyPessoa("C"));

            System.out.println("Nome: " + pessoas.get(0).getNome());
            // busca do banco de dados
            System.out.println("Nome: " + pessoas.get(1).getNome());
            // busca do banco de dados
            System.out.println("Nome: " + pessoas.get(0).getNome());
            // já buscou esta pessoa... apenas retorna do cache...

            // A terceira pessoa nunca será consultada no banco de dados
            (melhor performance - lazy loading)
            System.out.println("Id da 3ª - " + pessoas.get(2).getId());
            //pode imprimir o ID do objeto, e o proxy nao será inicializado.
      }
}
```

Saída do Programa

```
select * from PESSOA where id=A
Retornou a pessoa do banco de dados ->  Pessoa A
Nome: Pessoa A
select * from PESSOA where id=B
Retornou a pessoa do banco de dados ->  Pessoa B
Nome: Pessoa B
Nome: Pessoa A
Id da 3ª – C
```

- **Arquitetura de Camadas Físicas**:
  - **Outros padrões**:
    - **Brokerage em serviços Web**:
      - Padrão arquitetônico que **suporta interoperabilidade em infraestrutura distribuída** potencialmente complexa.
      - Consiste no trio: provedor de serviço, solicitante de serviço e “corretor” de serviço



Agenda

1. Introdução
2. Modelos Físicos
3. Modelos de arquitetura para sistemas distribuídos
4. **Modelos fundamentais**

- Modelo fundamental:
  - deve conter apenas os conteúdos essenciais que precisamos considerar para **entender e raciocinar a respeito de certos aspectos do comportamento de um sistema**
- Objetivos:
  - **Tornar explícitas todas as suposições** relevantes sobre os sistemas que estamos modelando
  - **Fazer generalizações** a respeito do que é possível ou impossível, dadas essas suposições

- Aspectos que serão abordados:
  - **Interação**:
    - Computação é feita por processos;
    - Interagem passando mensagens, resultando na comunicação (fluxo de informações) e na coordenação (sincronização e ordenação das atividades) entre eles
  - **Falha**:
    - Ameaça ao correto funcionamento de um sistema distribuído devido a falha em qualquer um dos computadores em que ele é executado (incluindo falhas de software) ou na rede que os interliga

- Aspectos que serão abordados:
  - **Segurança**:
    - a natureza modular dos sistemas distribuídos expõem-nos a ataques de agentes externos e internos

Relembrando, qual a definição de sistemas distribuídos?

- Um sistema distribuído é constituído por múltiplos processos trocando informações entre si
- Baseado nisso, qual a definição de um **algoritmo distribuído**?

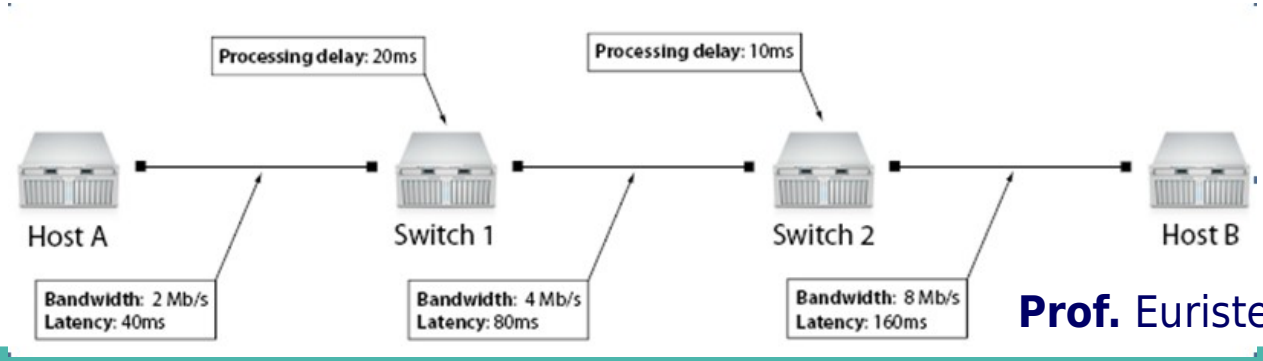
- Um sistema distribuído é constituído por múltiplos processos trocando informações entre si
- Baseado nisso, qual a definição de um **algoritmo distribuído**?
  - **Definição dos passos a serem executados** por cada um dos **processos** que compõem o sistema, incluindo a **transmissão de mensagens entre eles**.
  - As mensagens são enviadas para **transferir informações entre processos e para coordenar suas atividades**

Quais as consequências em se ter um algoritmo distribuído?

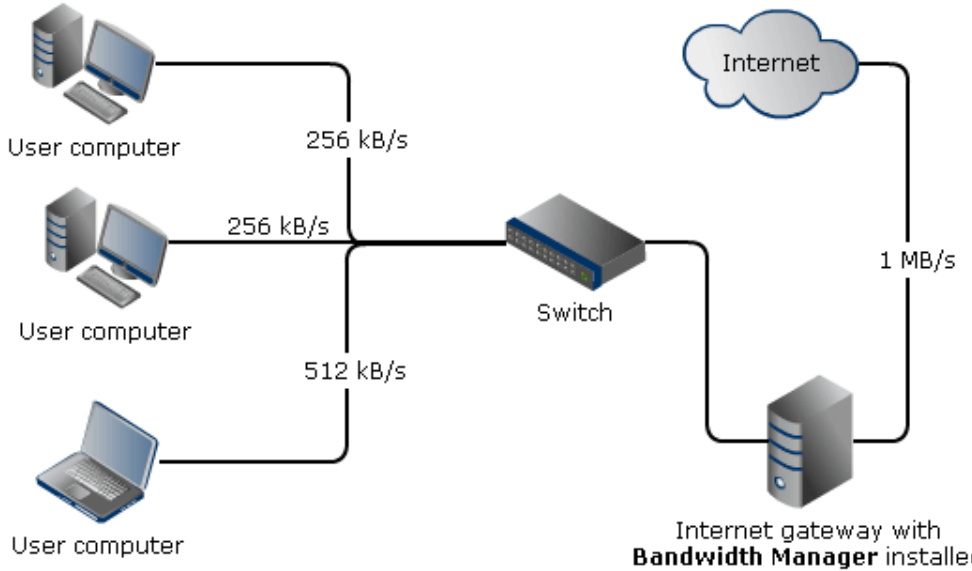
- Consequências:
  - não é possível prever a **velocidade com que cada processo é executado** e a sincronização da troca das mensagens entre eles.
  - difícil descrever **todos os estados** de um algoritmo distribuído (ocorrência de falhas)
  - O **estado pertencente a cada processo é privativo**, isto é, ele não pode ser acessado, nem atualizado, por nenhum outro processo
- Serão analisados dois fatores que impactam significativamente a interação entre processos:
  - Desempenho da comunicação
  - Impossibilidade de manter uma noção global de tempo única

Quais as características de desempenho relacionadas a redes de computadores?

- **Desempenho da comunicação:**
  - **Latência**:
    - É o **atraso** decorrido entre o início da transmissão de uma mensagem em um processo remetente e o início da recepção pelo processo destinatário. A latência inclui:
      - O tempo que o primeiro bit de um conjunto de bits transmitido em uma rede leva para chegar ao seu destino
      - O **atraso no acesso à rede**
      - **Tempo de processamento gasto pelos serviços de comunicação**

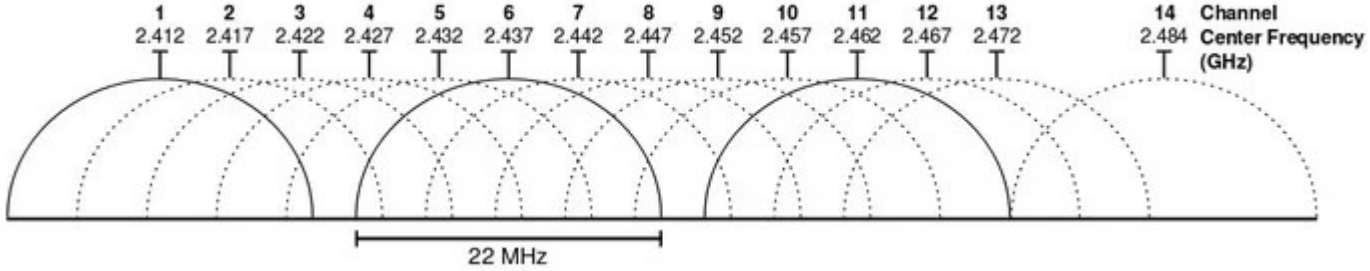


- **Desempenho da comunicação:**
  - **Largura de Banda**:
    - Volume total de informações que pode ser transmitido em determinado momento

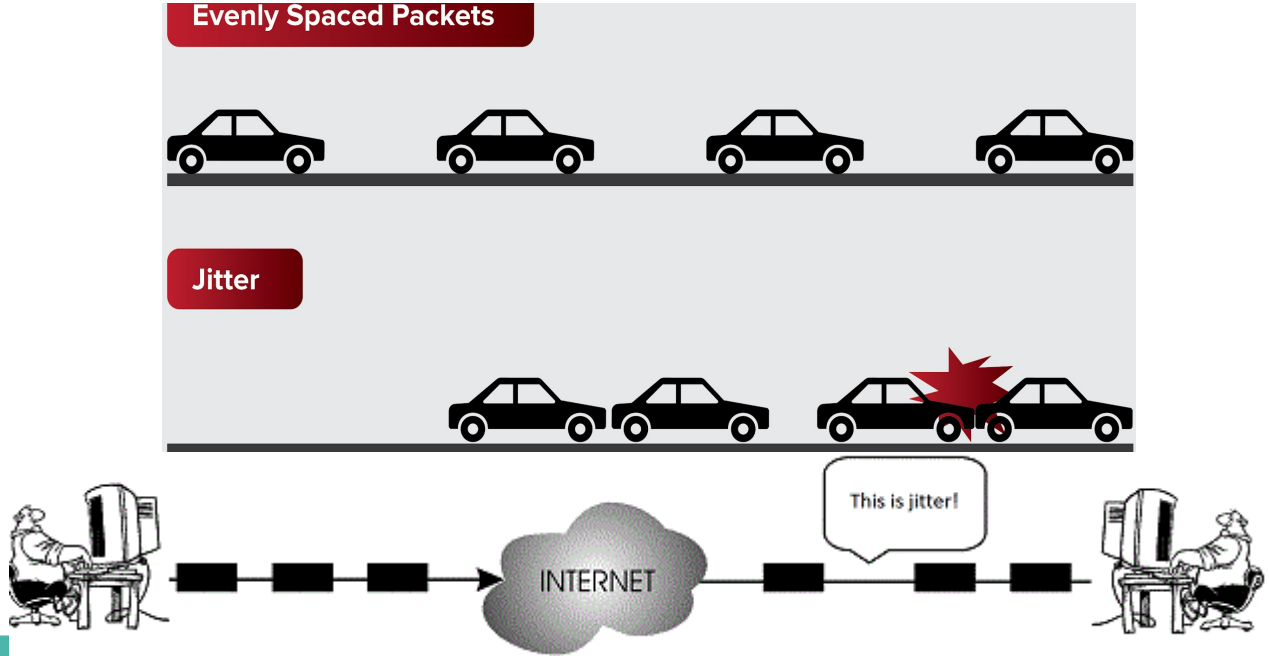


- **Desempenho da comunicação:**
  - **Largura de Banda**:
    - Uma frequência é dividida em canais. A banda de 2,4 GHz, por exemplo, possui 14 canais, um a cada 5 MHz, começando do 2412 MHz. Cada um desses canais tem 22 MHz de largura, valor que determina a capacidade de transferência de dados. Isto é, um canal com largura de 22 MHz tem disponibilidade de enviar informações dentro desse espectro.

      Como é possível ver no esquema abaixo, os canais se sobrepõem em certas faixas de frequência. Tomando como exemplo o canal 6, pode-se perceber que abrange frequências usadas também pelos canais 2, 3, 4, 5, 7, 8, 9 e 10.



- **Desempenho da comunicação:**
  - **Jitter**:
    - É a variação no tempo para distribuir uma série de mensagens. O jitter é crucial para dados multimídia



Qual o impacto de não haver uma noção global de tempo única?

- **Impossibilidade de manter uma noção global de tempo única**:
  - Cada computador possuir seu relógio local
  - Mesmo que dois processos leiam seus relógios locais ao mesmo tempo, esses **podem fornecer valores diferentes**.
  - **Motivo**: relógios de computador possuem uma taxa de desvio:
    - O termo **taxa de desvio do relógio** (drift) se refere à **quantidade relativa pela qual um relógio de computador difere de um relógio de referência perfeito**

- Existem duas variantes do modelo de Interação:
  - **Sistemas distribuídos síncronos**:
    - Definição de Hadzilacos e Toueg [1994]:
      - o **tempo para executar cada etapa de um processo tem limites inferior e superior conhecidos**;
      - cada mensagem transmitida em um canal é recebida dentro de um **tempo limitado, conhecido**;
      - cada processo tem um relógio local cuja **taxa de desvio do tempo real tem um valor máximo conhecido**.

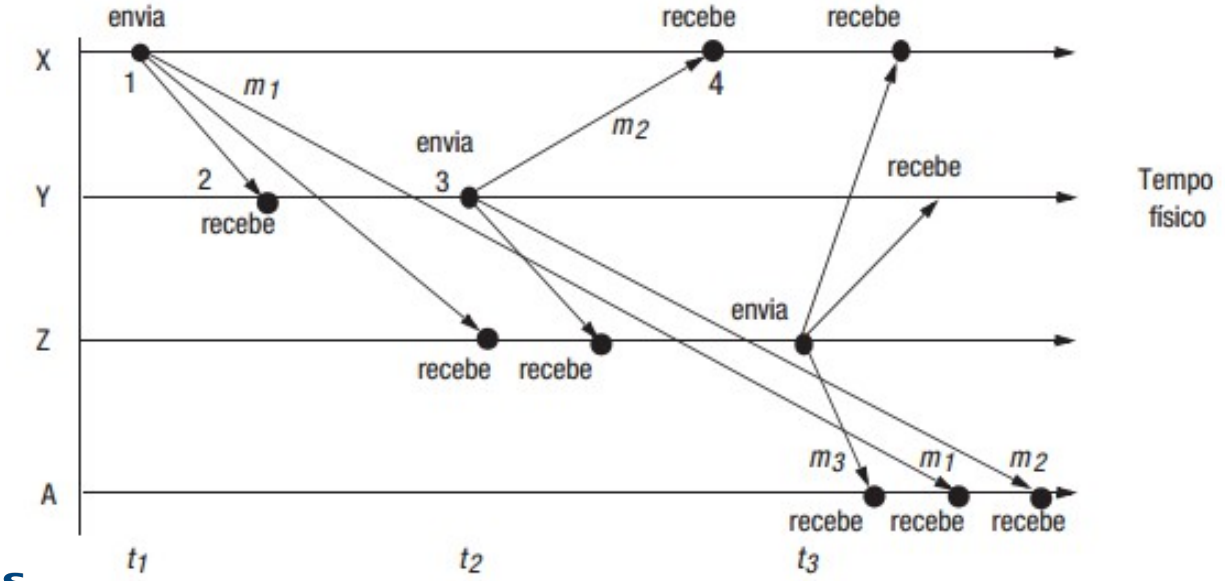
- Existem duas variantes do modelo de Interação:
  - **Sistemas distribuídos assíncronos**:
    - É um sistema que **não existem considerações sobre**:
      - As velocidades de execução de processos
      - Os atrasos na transmissão das mensagens
      - As taxas de desvio do relógio
    - O modelo assíncrono **não faz nenhuma consideração sobre os intervalos de tempo envolvidos** em qualquer tipo de execução.

Então, como as mensagens são ordenadas?

- **Ordenação de eventos**:
  - Muitas vezes, é necessário saber se um evento (envio ou recepção de uma mensagem) **ocorreu em um processo antes, depois ou simultaneamente** com outro evento em outro processo.
    - Ex: envio de email
  - Em sistemas distribuídos os relógios não podem ser perfeitamente sincronizados
  - **Lamport [1978]** propôs um modelo de **relógio lógico** para proporcionar uma ordenação de eventos ocorridos em processos **executados em diferentes computadores**.

- **Ordenação de eventos**:
  - O relógio lógico permite **deduzir a ordem em que as mensagens devem ser apresentadas**, sem apelar para os relógios físicos de cada máquina.
  - O relógio lógico **atribui a cada evento um número correspondente à sua ordem lógica**, de modo que os eventos posteriores tenham números mais altos do que os anteriores.

- **Ordenação de eventos**:
  - Como podemos resolver o problema nas mensagens que chegam no computador A?



Como você definiria uma falha em sistema distribuído?

- Em um sistema distribuído, **tanto os processos como os canais de comunicação podem falhar** – isto é, podem divergir do que é considerado um comportamento correto ou desejável.
- O modelo de falhas define **como uma falha pode se manifestar em um sistema**, de forma a proporcionar um entendimento dos **seus efeitos e consequências**

- **Modelo de falhas de Hadzilacos e Toueg [1994]**:
  - **Falhas por omissão**: As falhas classificadas como falhas por omissão se referem aos casos em que **um processo ou canal de comunicação deixa de executar as ações que deveria**.
    - **Falhas por omissão de processo**: a principal falha por omissão de um processo é quando ele entra em colapso, parando e não executando outro passo de seu programa
      - esse método de detecção de falhas é baseado no uso de timeouts

- **Modelo de falhas de Hadzilacos e Toueg [1994]**:
  - **Falhas por omissão**:
    - **Falhas por omissão na comunicação:** O canal de comunicação produz uma falha por omissão **quando não concretiza a transferência de uma mensagem** m do buffer de envio de p para o buffer de recepção de q.

Baseado nesse cenário, como podemos dividir a falha por omissão na comunicação?

- **Modelo de falhas de Hadzilacos e Toueg [1994]**:
  - **Falhas por omissão**:
    - **Falhas por omissão na comunicação:**
      - Categorização:
        - falhas por **omissão de envio**
        - falhas por **omissão de recepção**
        - falhas por **omissão de canal**

- **Modelo de falhas de Hadzilacos e Toueg [1994]**:
  - **Falhas Arbitrárias:**
    - O termo falha arbitrária, ou bizantina, é usado para **descrever a pior semântica de falha possível** na qual qualquer tipo de erro pode ocorrer.
      - Uma falha arbitrária de um processo é aquela em que ele **omite** arbitrariamente passos desejados do processamento ou **efetua processamento indesejado**.
    - Portanto, as falhas arbitrárias não podem ser detectadas verificando-se se o processo responde às invocações, pois ele poderia omitir arbitrariamente a resposta.

- **Modelo de falhas de Hadzilacos e Toueg [1994]**:
  - Resumindo...

Classe da falha	Afeta	Descrição
Parada por falha	Processo	O processo pára e permanece parado. Outros processos podem detectar esse estado.
Colapso	Processo	O processo pára e permanece parado. Outros processos podem não detectar esse estado.
Omissão	Canal	Uma mensagem inserida em um *buffer* de envio nunca chega no *buffer* de recepção do destinatário.
Omissão de envio	Processo	Um processo conclui um envio, mas a mensagem não é colocada em seu *buffer* de envio.
Omissão de recepção	Processo	Uma mensagem é colocada no *buffer* de recepção de um processo, mas esse processo não a recebe efetivamente.
Arbitrária (bizantina)	Processo ou canal	O processo/canal exibe comportamento arbitrário: ele pode enviar/transmitir mensagens arbitrárias em qualquer momento, cometer omissões; um processo pode parar ou realizar uma ação incorreta.

- **Modelo de falhas de Hadzilacos e Toueg [1994]**:
  - **Falhas de temporização**: aplicáveis aos **sistemas distribuídos síncronos** em que limites são estabelecidos para o tempo de execução do processo, para o tempo de entrega de mensagens e para a taxa de desvio do relógio
    - A temporização é particularmente relevante **para aplicações multimídia**, com canais de áudio e vídeo, e para sistemas de tempo real.

Modelos de Fundamentais - Modelo de Falha

O que seria o mascaramento de falhas?

- **Mascaramento de Falhas:**
  - Cada componente em um sistema distribuído geralmente é construído a partir de um conjunto de outros componentes.
  - É possível construir serviços confiáveis a partir de componentes que exibem falhas.
  - O conhecimento das características da falha de um componente pode permitir que **um novo serviço seja projetado de forma a mascarar a falha dos componentes dos quais ele depende**.
    - Qual seria um exemplo disso?

- **Mascaramento de Falhas:**
  - Um serviço mascara uma falha ocultando-a completamente ou convertendo-a em um tipo de falha mais aceitável.
    - Ex: somas de verificação
      - O que elas fazem?

- **Mascaramento de Falhas:**
  - Um serviço mascara uma falha ocultando-a completamente ou convertendo-a em um tipo de falha mais aceitável.
    - Ex: somas de verificação
      - O que elas fazem?
        - são usadas para mascarar mensagens corrompidas – convertendo uma falha arbitrária em falha por omissão.

O que é necessário para dizer que tem-se uma comunicação confiável?

- **Confiabilidade da comunicação:**
  - O termo comunicação confiável é **definido em termos de validade e integridade**, como segue:
    - **Validade**: qualquer mensagem do buffer de envio é entregue ao buffer de recepção de seu destino, independentemente do tempo necessário para tal;
    - **Integridade**: a mensagem recebida é idêntica à enviada e nenhuma mensagem é entregue duas vezes.

Modelos de Fundamentais - Modelo de Falha

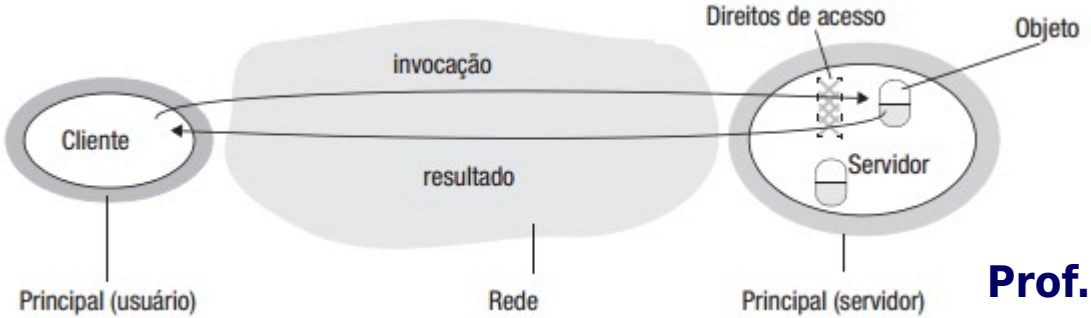
Como a integridade pode ser ameaçada?

- **Confiabilidade da comunicação:**
  - As ameaças à integridade vêm de duas fontes independentes:
    - Qualquer **protocolo que retransmite mensagens, mas não rejeite uma mensagem que foi entregue duas vezes**.
      - Os protocolos podem incluir números de sequência nas mensagens para detectar aquelas que são entregues duplicadas.
    - Usuários mal-intencionados que podem **injetar mensagens espúrias, reproduzir mensagens antigas ou falsificar mensagens**.
      - Medidas de segurança podem ser tomadas para manter a propriedade da integridade diante de tais ataques.

Quais elementos de um sistema distribuído devem ser protegidos para se ter um ambiente seguro?

- Princípio de funcionamento:
  - a segurança de um sistema distribuído pode ser obtida **tornando seguros os processos e os canais usados** por suas interações e **protegendo contra acesso não autorizado** os objetos que encapsulam.

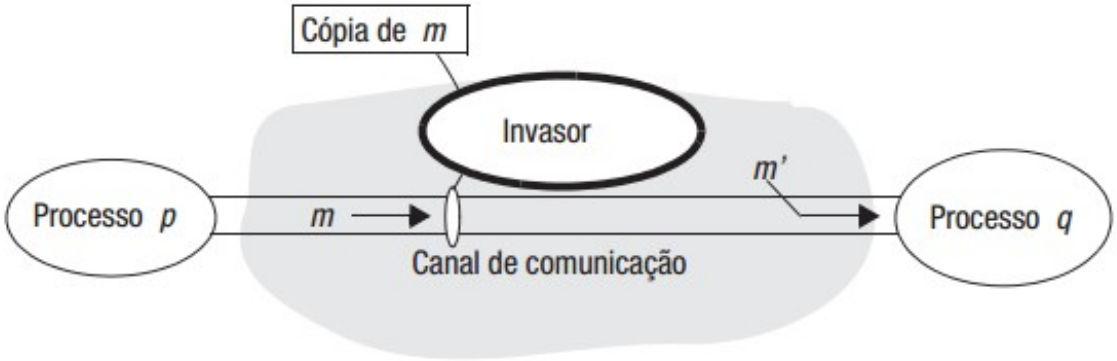
- **Proteção de objetos**:
  - **Direitos de acesso** - especificam quem pode executar determinadas operações sobre um objeto – por exemplo, quem pode ler ou escrever seu estado
    - O servidor é responsável por **verificar a identidade do usuário** que está por trás de cada invocação e conferir se ele **tem direitos de acesso suficientes** para efetuar a operação solicitada em determinado objeto, rejeitando as que ele não pode efetuar.



Quais elementos de um sistema distribuído estão expostos a ataques?

- As mensagens ficam expostas a ataques, porque **o acesso à rede e ao serviço de comunicação é livre** para permitir que quaisquer dois processos interajam.
- Modelo para a análise de ameaças:
  - Invasor
  - Ameaças aos processos
  - Ameaças aos canais de comunicação
  - Métodos preventivos

- **Invasor**:
  - É capaz de enviar qualquer mensagem para qualquer processo e ler ou copiar qualquer mensagem entre dois processo



- **Ameaças ao processo**:
  - Na arquitetura atual das redes, **não há um reconhecimento confiável da origem de uma mensagem**, tornando-se uma ameaça tanto para servidores como para clientes:
    - **Servidores**: Mesmo que um servidor exija a inclusão da identidade do principal em cada invocação, um atacante poderia gerá-la com uma identidade falsa.
      - Sem o reconhecimento garantido da identidade do remetente, um servidor não pode saber se deve executar a operação ou rejeitá-la
    - **Clientes**: quando um cliente recebe o resultado de uma invocação feita a um servidor, **ele não consegue identificar se a origem da mensagem** com o resultado é proveniente do servidor desejado ou de um invasor, talvez fazendo **spoofing** desse servidor.

Como anular essas ameaças?

- **Anulando as ameaças:**
  - **Criptografia e chaves compartilhadas**: dois processos (por exemplo, um cliente e um servidor) compartilhem um segredo
    - Criptografia é a ciência de manter as mensagens seguras, e **cifragem é o processo de embaralhar uma mensagem de maneira a ocultar seu conteúdo**.
  - **Autenticação**: o uso de chaves compartilhadas e da criptografia fornece a base para a autenticação de mensagens – provar as identidades de seus remetentes.
    - A técnica de autenticação básica é incluir em uma mensagem uma parte cifrada que possua conteúdo suficiente para garantir sua autenticidade.

- **Anulando as ameaças:**
  - **Utilização de canais seguros:**
    - Propriedades:
      - Cada um dos processos **conhece com certeza a identidade do principal** em nome de quem o outro processo está executando
      - Um canal seguro garante a **privacidade e a integridade** (proteção contra falsificação) dos dados transmitidos por ele
      - Cada mensagem **inclui uma indicação de relógio lógico ou físico** para impedir que as mensagens sejam reproduzidas ou reordenadas.

Obrigado!!!

Dúvidas?

Exercícios

1. Um mecanismo de busca é um servidor Web que responde aos pedidos do cliente para pesquisar em seus índices armazenados e (concomitantemente) executa várias tarefas de Web crawling para construir e atualizar esses índices. Quais são os requisitos de sincronização entre essas atividades concomitantes?
2. Frequentemente, os computadores usados nos sistemas peer-to-peer são computadores desktop dos escritórios ou das casas dos usuários. Quais são as implicações disso na disponibilidade e na segurança dos objetos de dados compartilhados que eles contêm e até que ponto qualquer vulnerabilidade pode ser superada por meio da replicação?

Bibliografia

- Sistemas Distribuídos - Conceitos e Projeto - 5ª Ed. 2013 - George Coulouris, Tim Kindberg, Jean Dollimore
- Andrew S. Tanembaum, Sistemas Distribuídos,. Princípios e Paradigmas 2ª edição