

# Universidade Federal do Sul e Sudeste do Pará

---

## Sistemas Distribuídos

*Prof.: Warley Junior*  
wmvj@unifesspa.edu.br

# PLANO DE DISCIPLINA

---

- ❑ Carga horária:

- ❑ 48 horas teórica / 20 horas prática

- ❑ Objetivo Geral:

- ❑ Prover o conhecimento dos conceitos básicos de sistemas distribuídos, bem como de técnicas e critérios de projeto e implementação.

# PLANO DE DISCIPLINA

---

## ❑ Objetivos específicos:

- ❑ Compreender o que são e para quê existem
- ❑ Saber quando e como usar
- ❑ Entender o arcabouço de componentes
- ❑ Aprender **técnicas** para sua construção
  - **Práticas**: paradigmas, arquiteturas, tecnologias
  - **Teóricas**: modelos, algoritmos

# METODOLOGIA

---

- ❑ 50% aulas síncronas (ao vivo)
- ❑ 50% aulas assíncronas

## Ferramentas

- ❑ **Google Meet** para aulas síncronas
- ❑ **SIGAA** para aulas assíncronas, materiais e atividades práticas
- ❑ **Discord** para chat, dúvidas e atividades em grupo

# METODOLOGIA

---

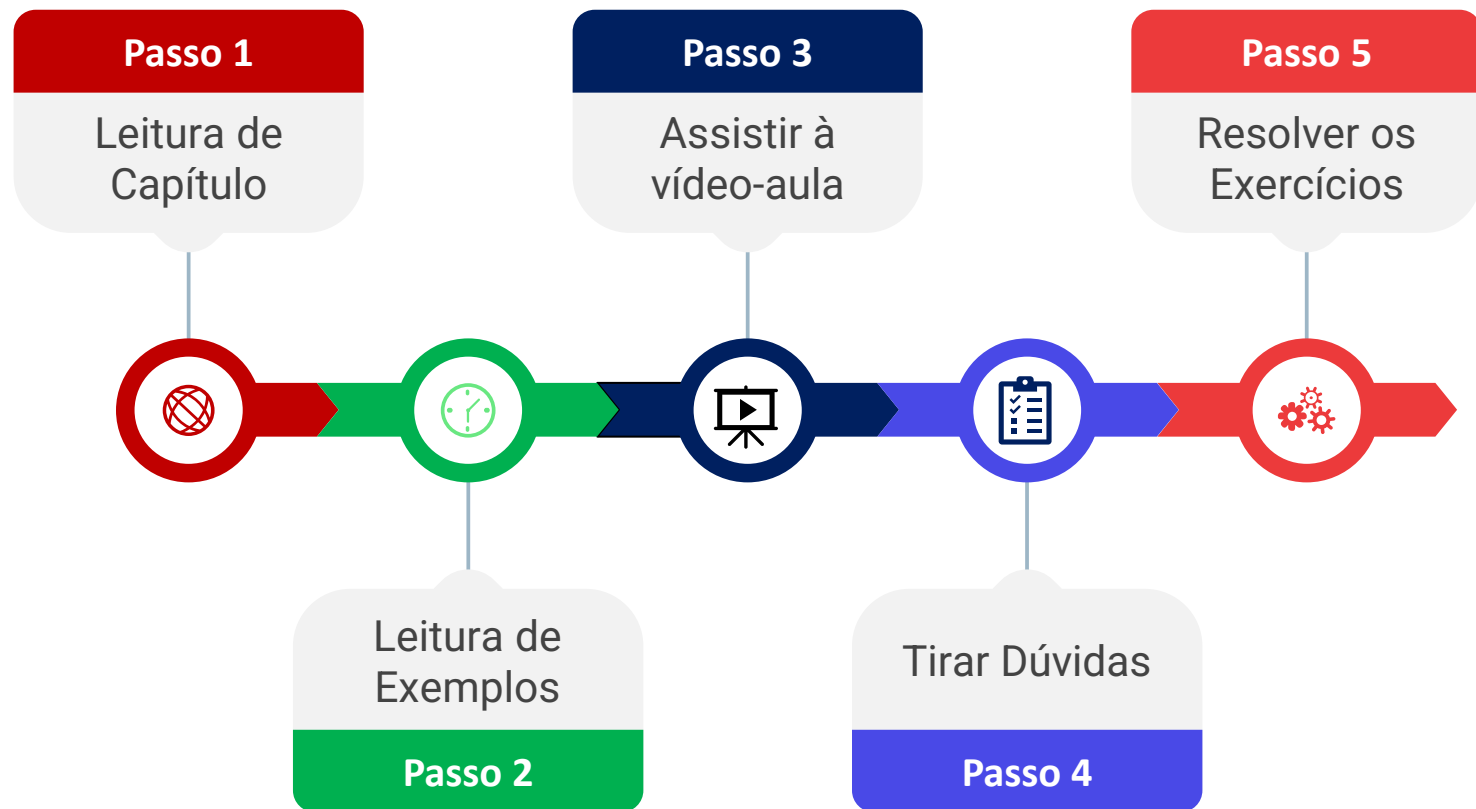
## Aulas assíncronas

- ❑ Vídeo-aulas;
- ❑ Materiais para leitura;
- ❑ Atividades (quizzes, questionários, práticas, ...);
- ❑ Outros materiais (tutoriais, vídeos externos, ...).

Divulgação do material da semana geralmente na quarta-feira

# METODOLOGIA

---



# METODOLOGIA

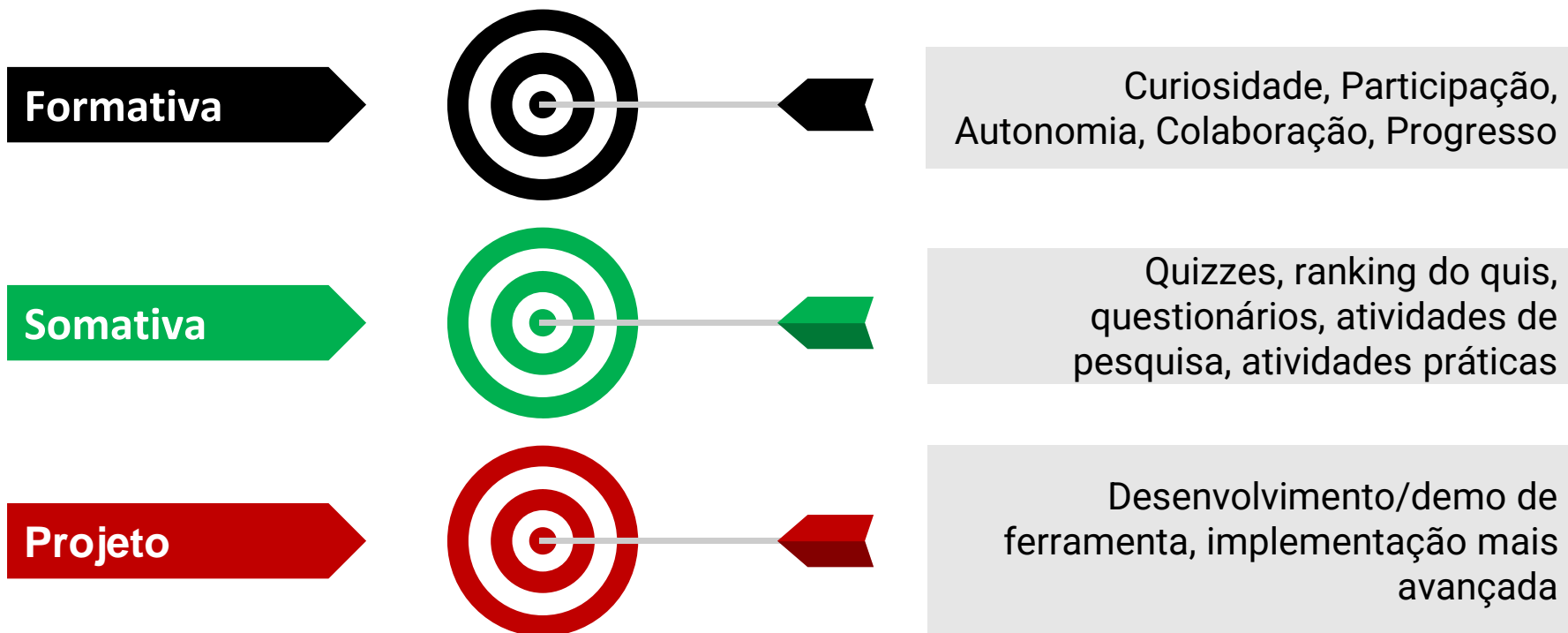
---

Aulas síncronas: segundas 08:20 às 10:10

- ❑ Discussão com alunos sobre o assunto da semana;
- ❑ Aspectos práticos de sistemas distribuídos;
- ❑ Atividade Síncrona;
- ❑ Conversa com profissionais.

# Avaliação

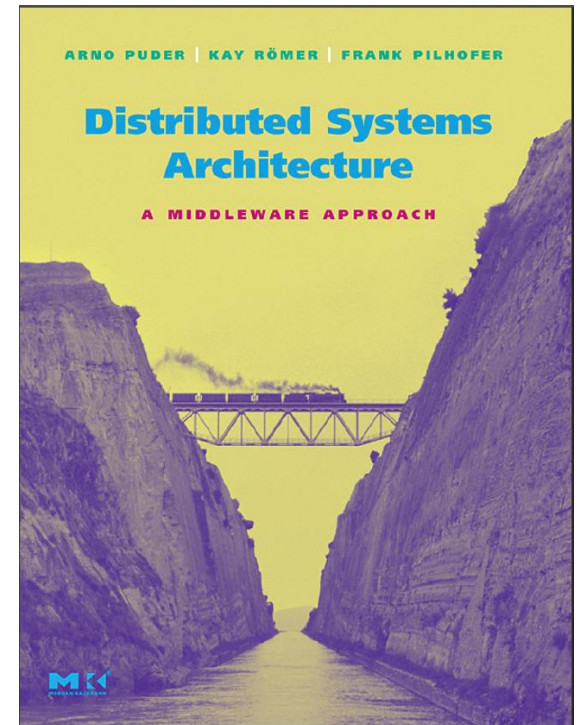
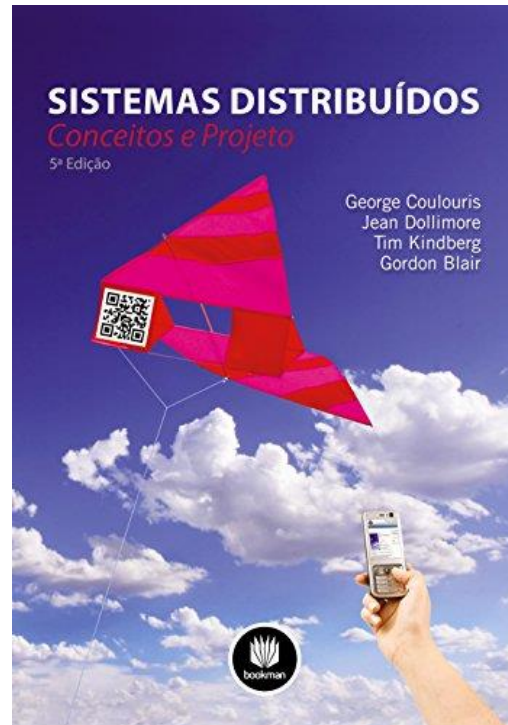
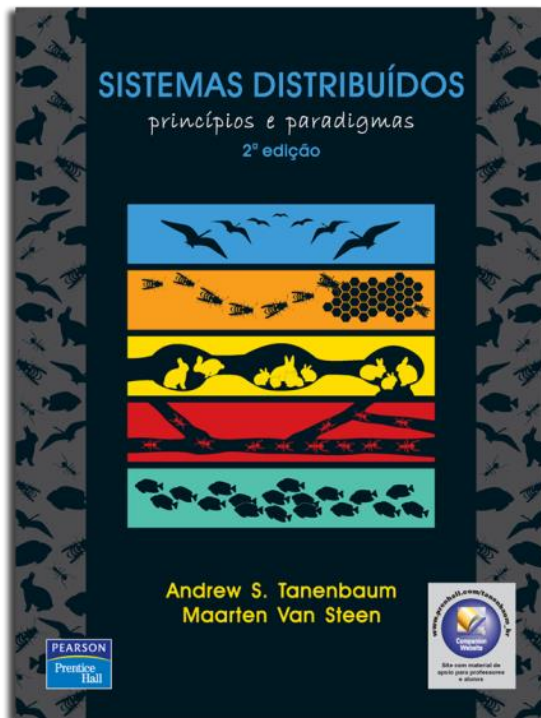
---





# PLANO DE DISCIPLINA

---



# Agenda

---

## ❑ AULA 1:

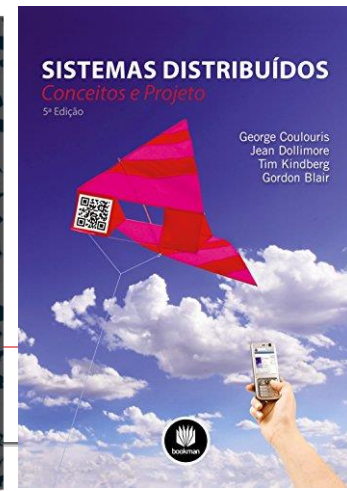
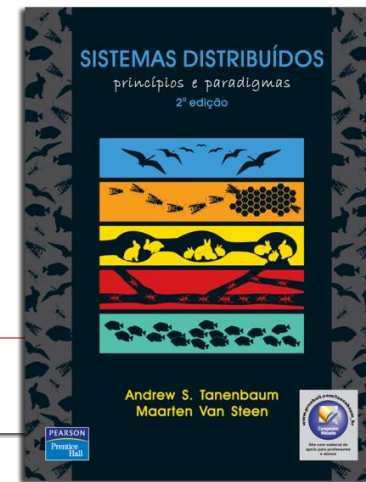
## ❑ Caracterização de Sistemas Distribuídos

- Definição
- Objetivos
- Tipos de SDs
- Desafios

# Leitura Prévia

---

- ❑ COULOURIS, George. Sistemas distribuídos: conceitos e projetos. 5ª ed. Porto Alegre: Bookman, 2013.
  - Capítulo 1.
- ❑ TANENBAUM, Andrew S. Sistemas distribuídos: princípios e paradigmas. 2ª ed. São Paulo: Pearson Prentice Hall, 2007.
  - Capítulo 1.



# Hora da interação!

---

□ <https://www.menti.com/kfhmjuh72i>



**9538 8502**

# O que é um sistema distribuído?

---

“

*Conjunto de **computadores independentes** que se apresenta a seus usuários como um **sistema único e coerente.** (Andrew Tanenbaum)*

---

# O que é um sistema distribuído?

---

“

*Sistema em que componentes de hardware e software localizados em **diferentes computadores interconectados se comunicam e coordenam suas ações trocando mensagens** (George Coulouris)*

---

# O que é um sistema distribuído?

---

“

*Sistema onde você não consegue trabalhar por causa de uma **falha** em um **computador que você nunca viu**”  
(Leslie Lamport)*

---

# O que é um sistema distribuído?

---

COMPUTADORES  
INTERCONECTADOS

COMUNICAÇÃO  
(MENSAGENS)

COORDENAÇÃO DE AÇÕES  
(COLABORAÇÃO)

TRANSPARÊNCIA (SISTEMA ÚNICO)

COERÊNCIA / CONSISTÊNCIA

FALHAS!



# O que é um sistema distribuído?

---

## Outra definição popular\*:

*"É a arte de resolver o **mesmo problema** que você pode resolver em um **único computador** usando **múltiplos computadores**"*

Por que então  
desenvolver um sistema  
distribuído?!



\* Distributed Systems for fun and profit. Mikito Takada.

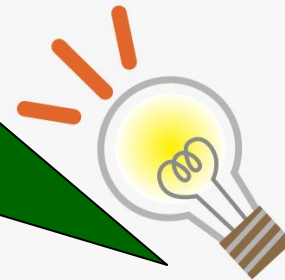
<http://book.mixu.net/distsys>

---

# O que é um sistema distribuído?

---

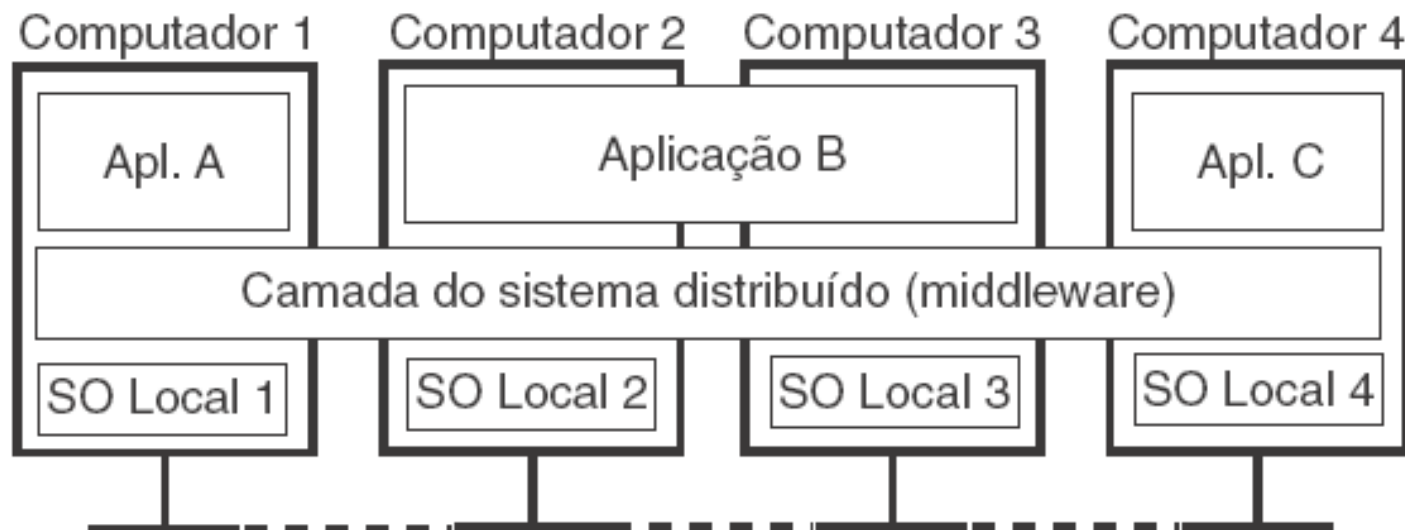
**Por que todo sistema que usa a *Internet* é distribuído. Ou seja, quase tudo desenvolvido atualmente!**



# Definição

---

- ❑ Sistemas distribuídos costumam ser organizados por meio de:
  - Camada de Nível Alto;
  - Camada de Software;
  - Camada Subjacente.



# Definição

---

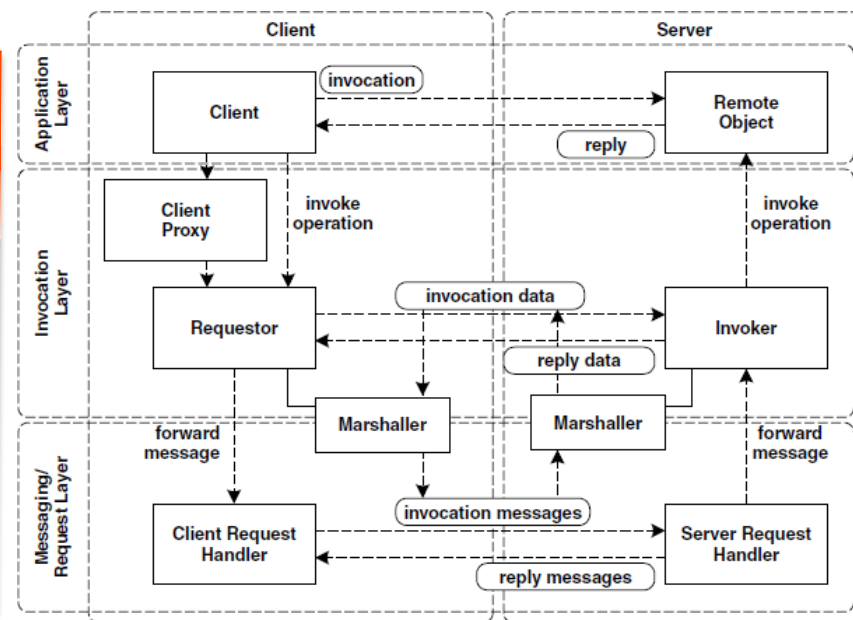
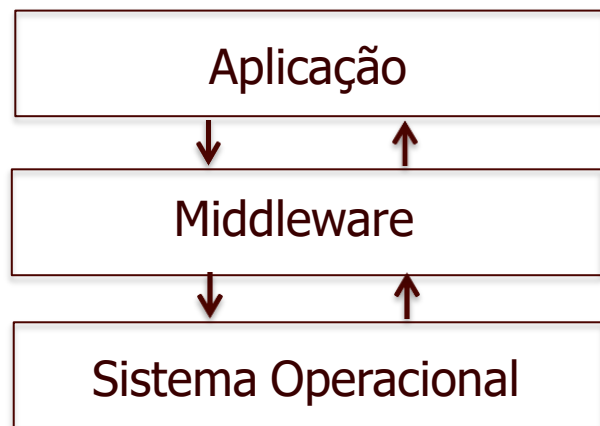
## ☐ MIDDLEWARE

- Camada de software que fornece abstração de programação e pode mascarar heterogeneidade de:

- ☐ Redes de computadores;
- ☐ Máquinas;
- ☐ Sistemas Operacionais;
- ☐ Linguagens de programação.

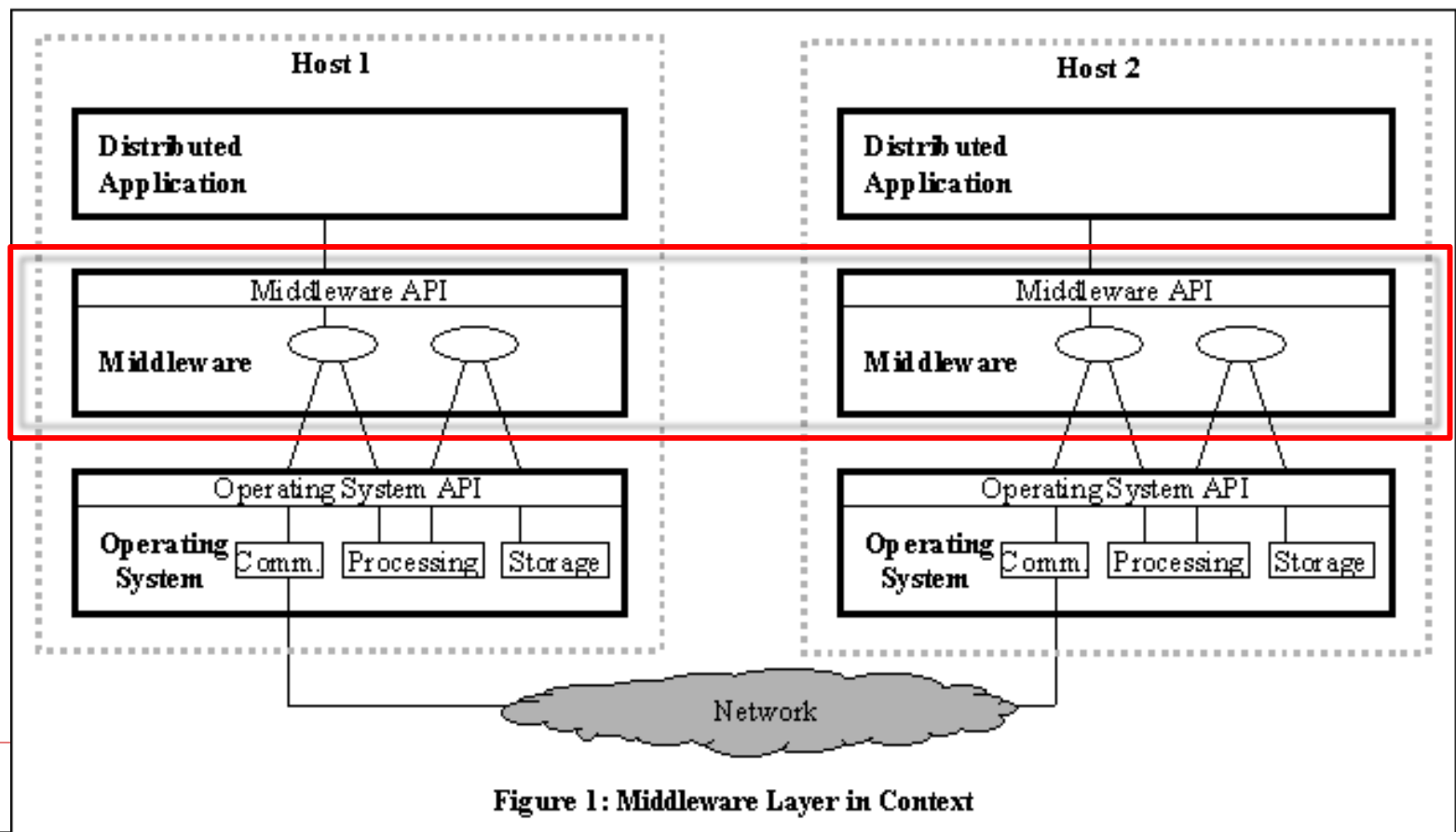
# Definição

## ❑ MIDDLEWARE



# Definição

## ❑ MIDDLEWARE



# Definição

---

## □ MIDDLEWARES

- Java RMI da SUN;
- CORBA da OMG;
- Sun RPC;
- Web Services;
- DCOM (*Distributed Component Object Model*) da Microsoft, etc.

# Por que sistemas distribuídos?

---

## ☐ **Compartilhamento**

- Documentos, dados, software, hardware

## ☐ **Escalabilidade / desempenho**

- Mais carga → Mais recursos → Bom desempenho

## ☐ **Custo x benefício**

- Computador: dinheiro em dobro  $\neq$  desempenho em dobro

## ☐ **Disponibilidade / integridade**

- Em caso de falhas, manter sistema no ar e não perder dados



# Objetivos

---

## Compartilhamento de recursos

- Documentos, dados, hardware, software, serviços, ...
- Por que?
  - Compartilhar é mais barato
  - Facilita colaboração e troca de informações



# Objetivos

---

## Transparência

- ❑ Aparenta ser um sistema único
- ❑ Distribuição de objetos é invisível
  - Esconde forma de acesso, localidade, relocação, migração, replicação, concorrência e falhas
- ❑ Pode ser relaxada em alguns casos
  - Desempenho, entendimento



# Objetivos

---

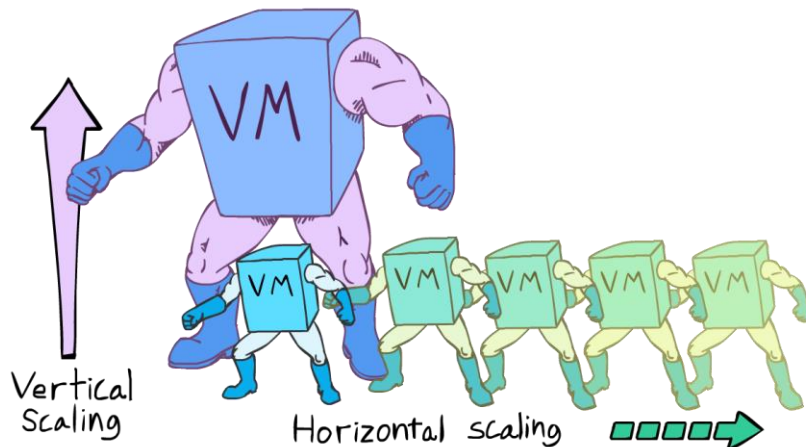


## Aberto

- ☐ Facilmente integrado a outros sistemas
- ☐ Padrões comuns, com interfaces bem definidas
- ☐ Interoperável, portátil e extensível

# Objetivos

---



## Escalabilidade

- Desempenho não deve ser afetado por fatores em dimensões como:
  - **Tamanho:** aumento de usuários e recursos
  - **Geográfica:** usuários e recursos em localizações diferentes
  - **Administrativa:** abrange diferentes organizações administrativas

# Tipos de SDs

## ❑ COMPUTAÇÃO EM CLUSTER

- Característica **Homogênea**

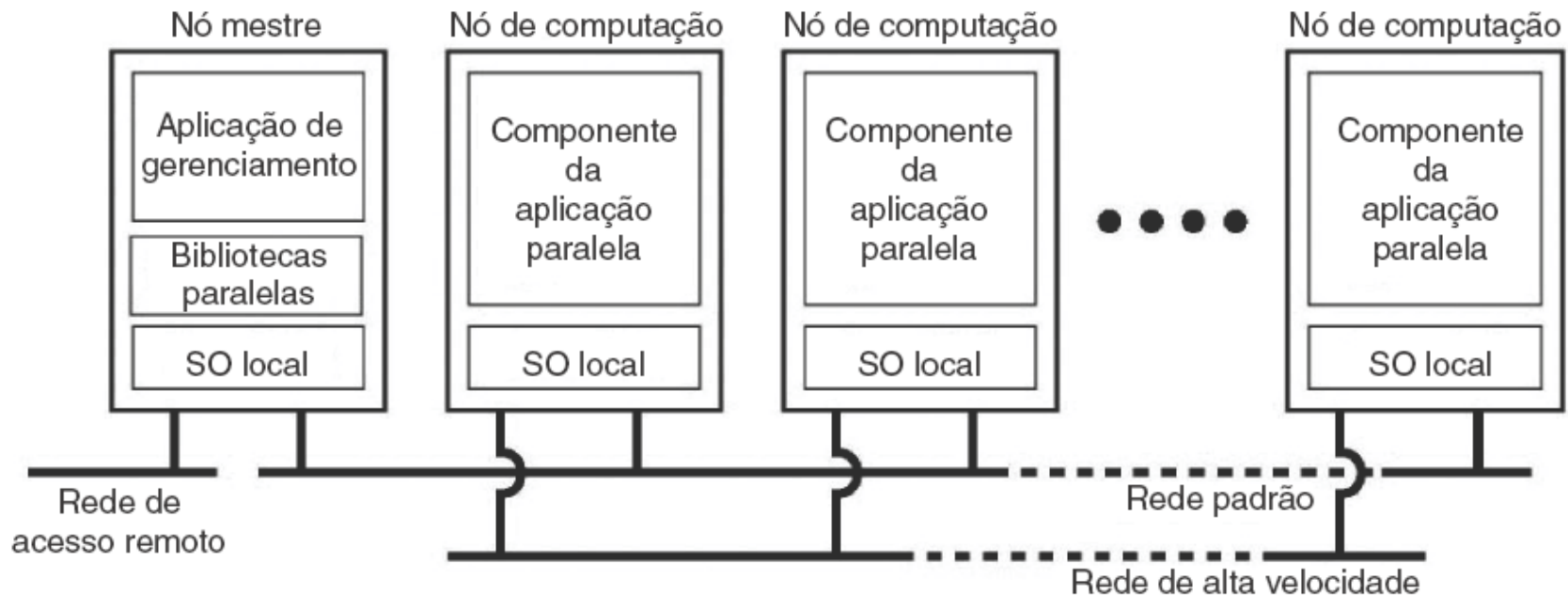


Figura 1.4 Exemplo de um sistema de computação de cluster.

# Tipos de SDs

---

## ❑ COMPUTAÇÃO EM GRADE

- Característica **Heterogeneidade**

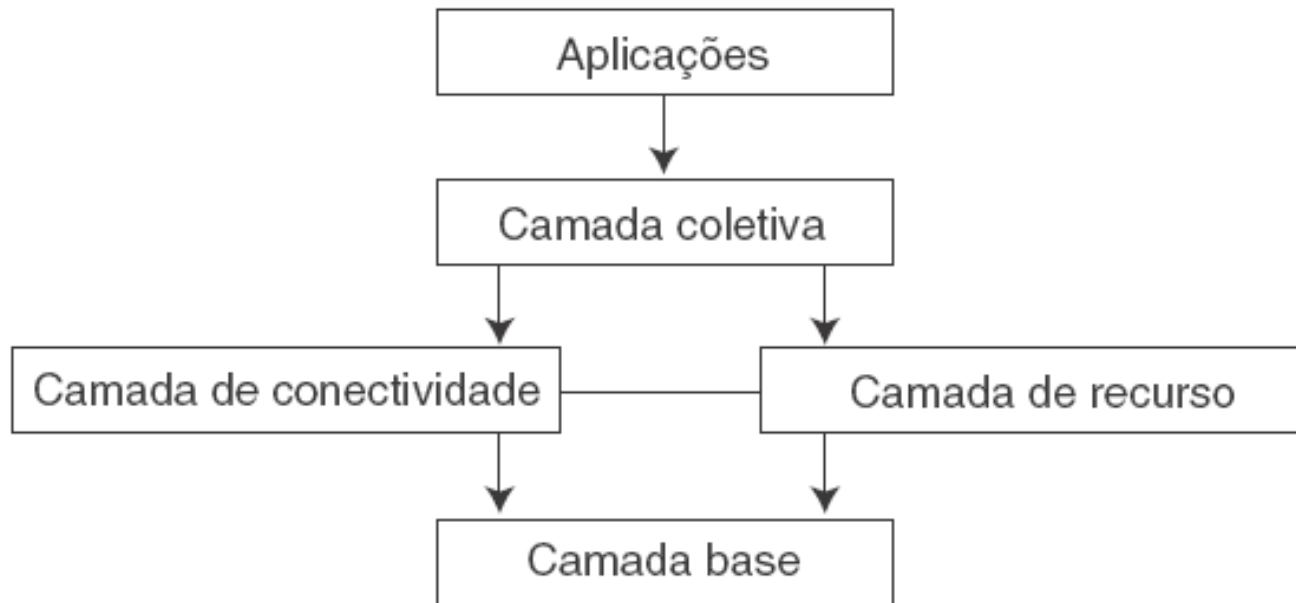


Figura 1.5 Arquitetura em camadas para sistemas de computação em grade. [spsa.edu.br](http://spsa.edu.br)

# Tipos de SDs

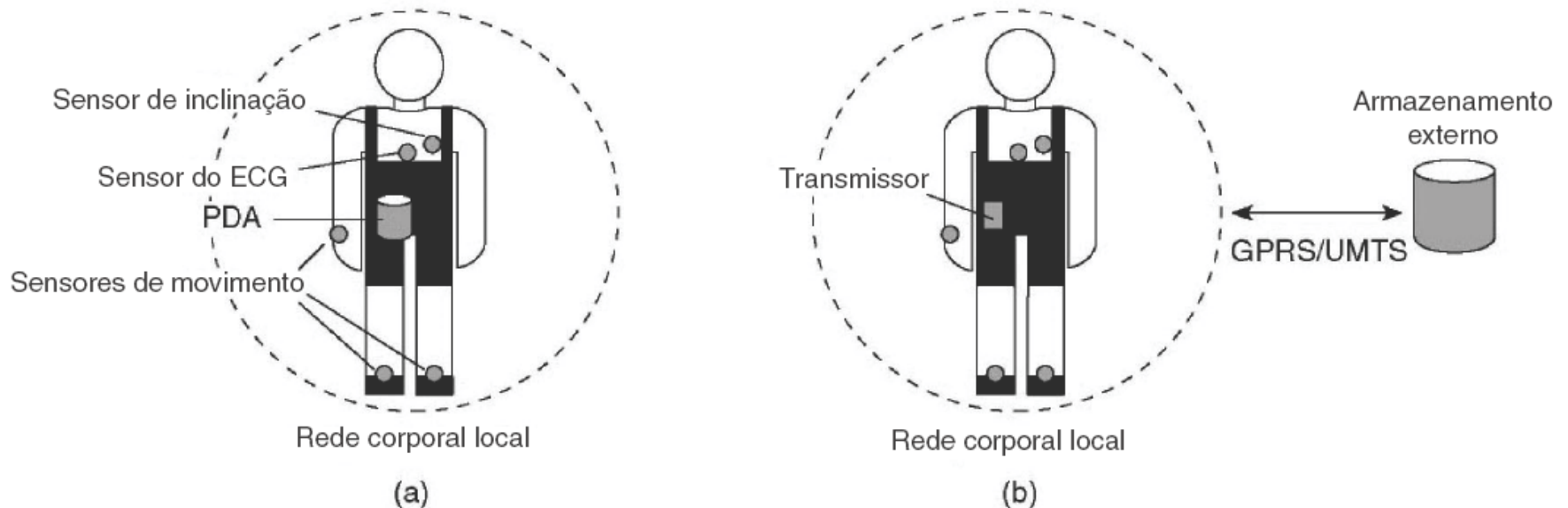
---

## ❑ Computação Móvel e Ubíqua

- ❑ Computação Móvel é a execução de tarefas de computação, enquanto o usuário está se deslocando de um lugar a outro ou visitando lugares diferentes de seu ambiente usual.
- ❑ Computação Ubíqua é a utilização de vários dispositivos computacionais pequenos e baratos, que estão presentes nos ambientes físicos dos usuários.

# Tipos de SDs

## ❑ COMPUTAÇÃO MÓVEL E UBÍQUA



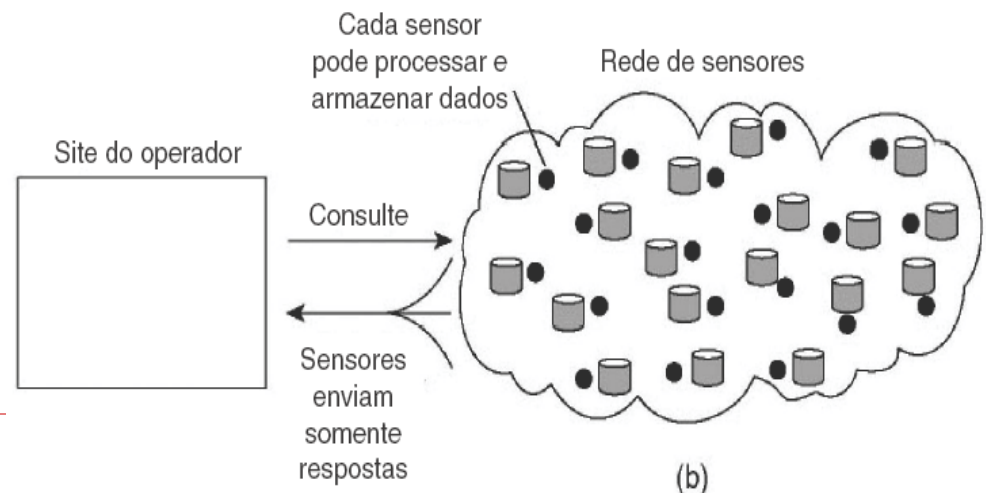
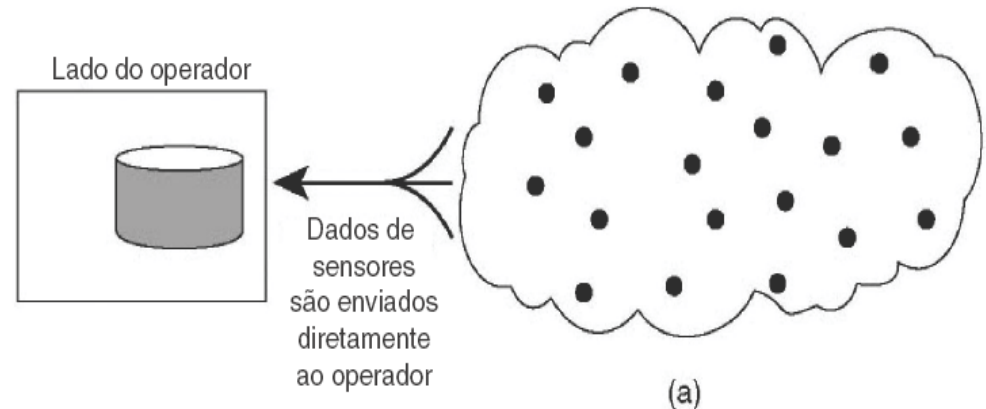
**Figura 1.9** Monitoração de uma pessoa em um sistema eletrônico pervasivo de tratamento de saúde utilizando (a) um hub local ou (b) uma conexão contínua sem fio.



# Tipos de SDs

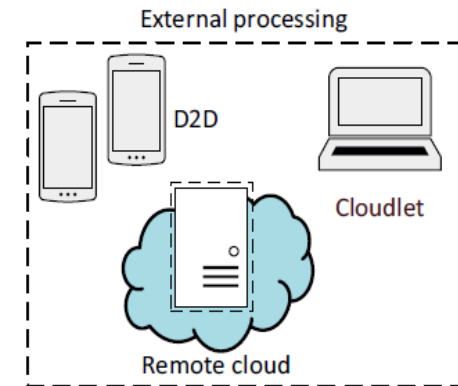
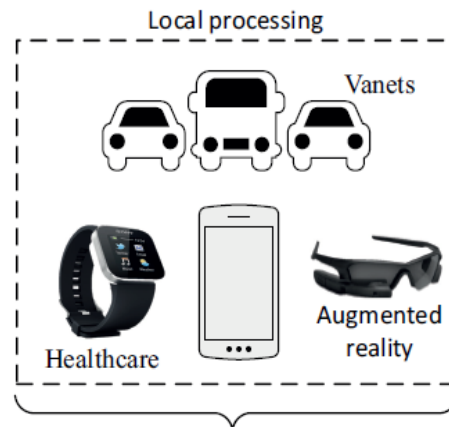
## □ REDE DE SENSORES SEM FIO

- Smartphone sensing
- Crowdsourcing
- Internet das Coisas
- Sistemas Ciberfísicos

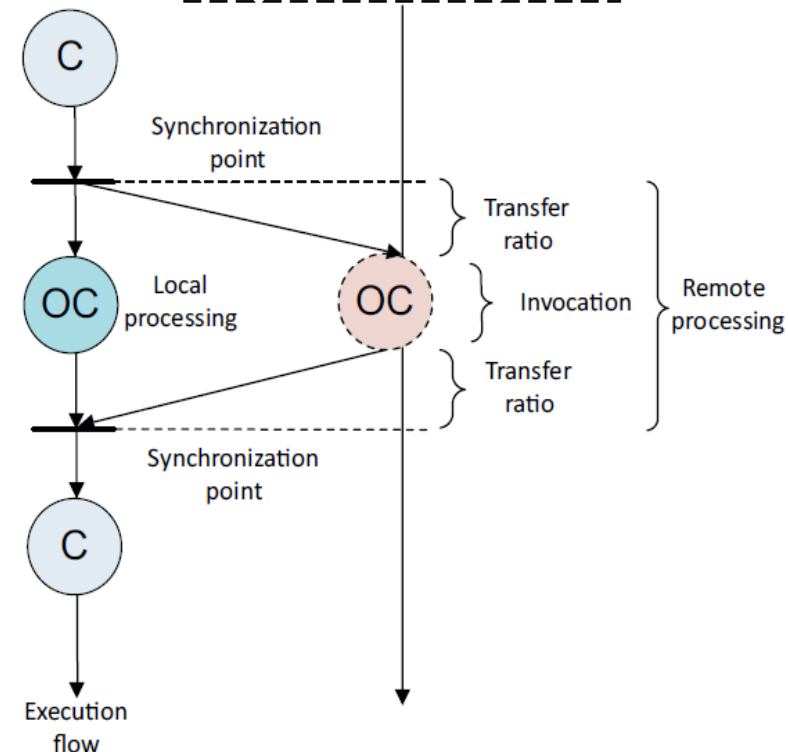


# Tipos de SDs

## □ Computação Móvel na Nuvem

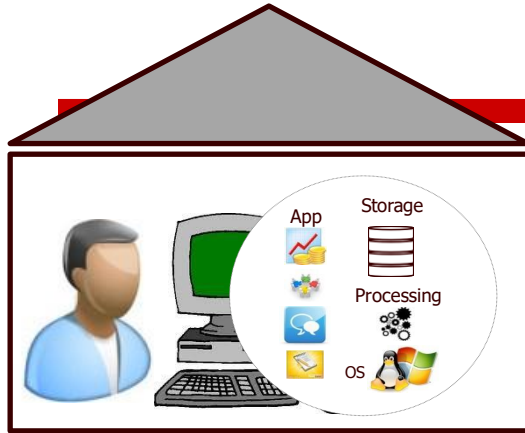


```
public class image {
    void method1(){
        //...
    }
    void method2(){
        // Offloading Candidate
    }
    void method3(){
        //...
    }
    main(){
        method1();
        method2();
        method3();
    }
}
```



# ❑ COMPUTAÇÃO EM NUVEM

## “Traditional” Computing



- ✓ Everything is **“real”** (e.g., you touch your computer)
- ✓ You **buy** everything.
- ✓ You **manage** everything, e.g., buy a new disk storage, pay for software licenses, update software versions, you care about security issues etc.
- ✓ Everything is **only yours**.
- ✓ You **must be at home** to use everything.

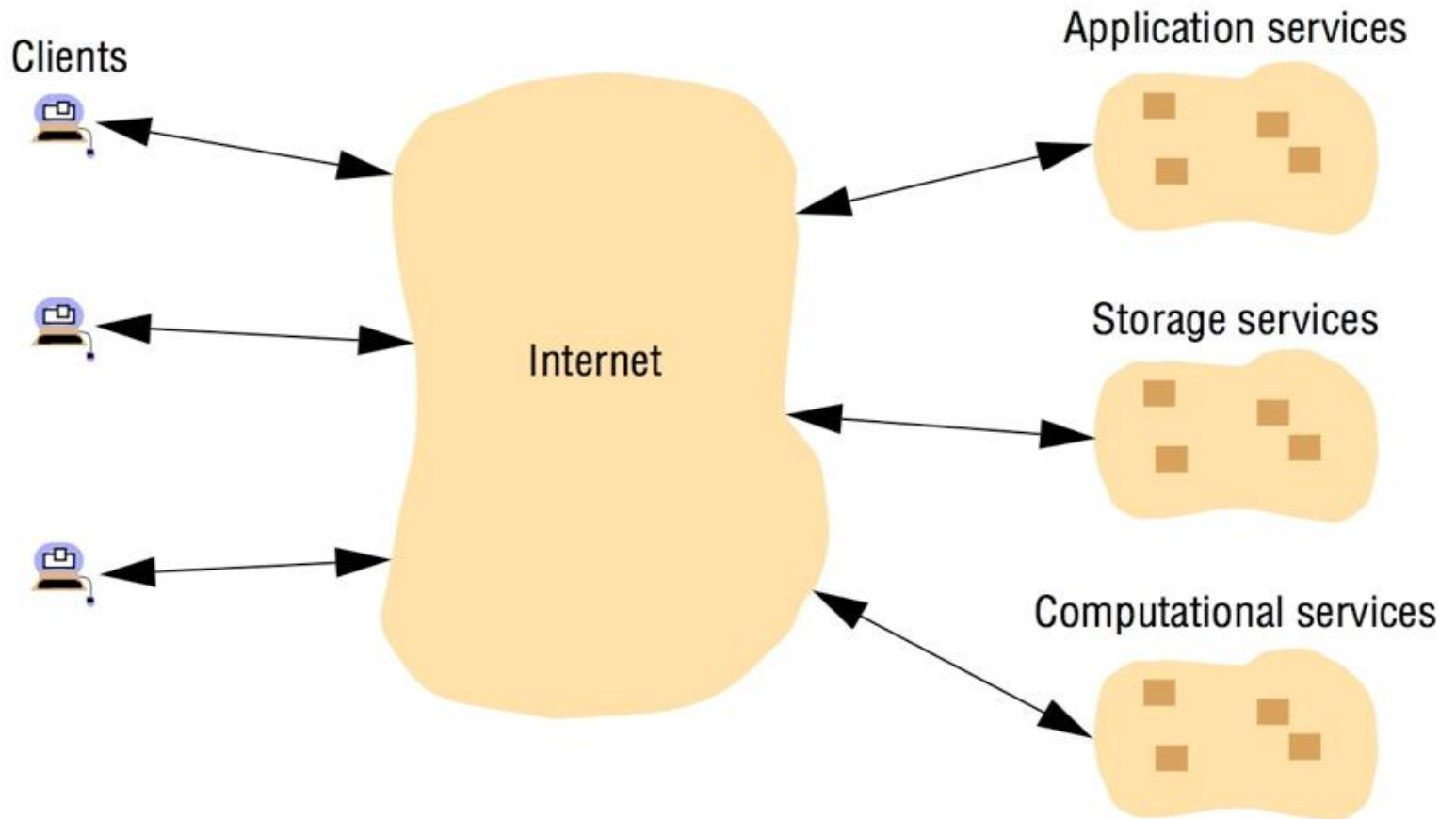
## Cloud computing



- ✓ Everything is **virtual** (you do not touch your computer, it is a virtual machine).
- ✓ **Monthly subscription** (pay-as-you-go).
- ✓ Cloud provider **manages** everything.
- ✓ Almost everything is **shared**.
- ✓ You may be **anywhere** (computer+browser).
- ✓ Everything is a **service**.

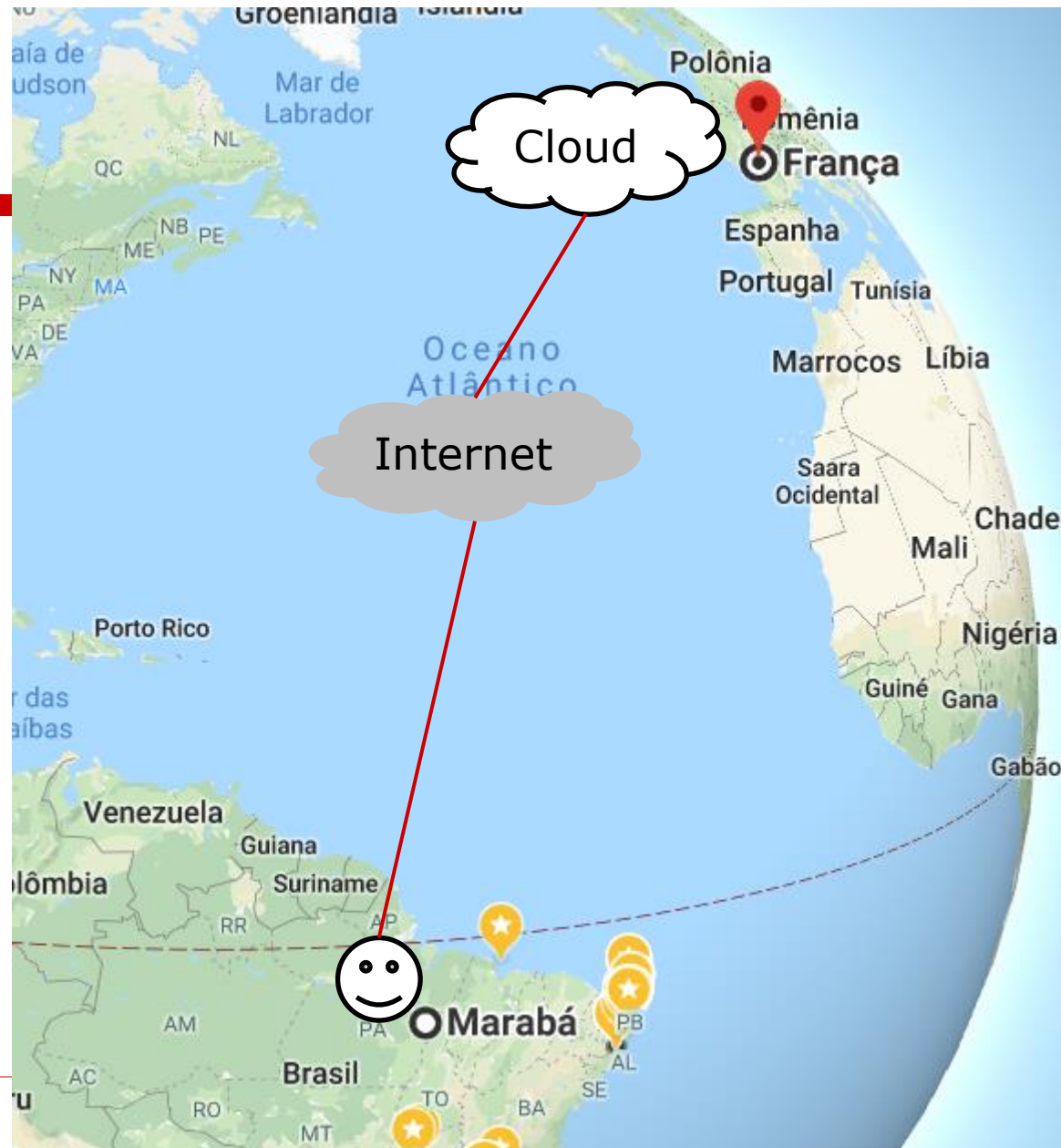
# Tipos de SDs

## ❑ COMPUTAÇÃO EM NUVEM



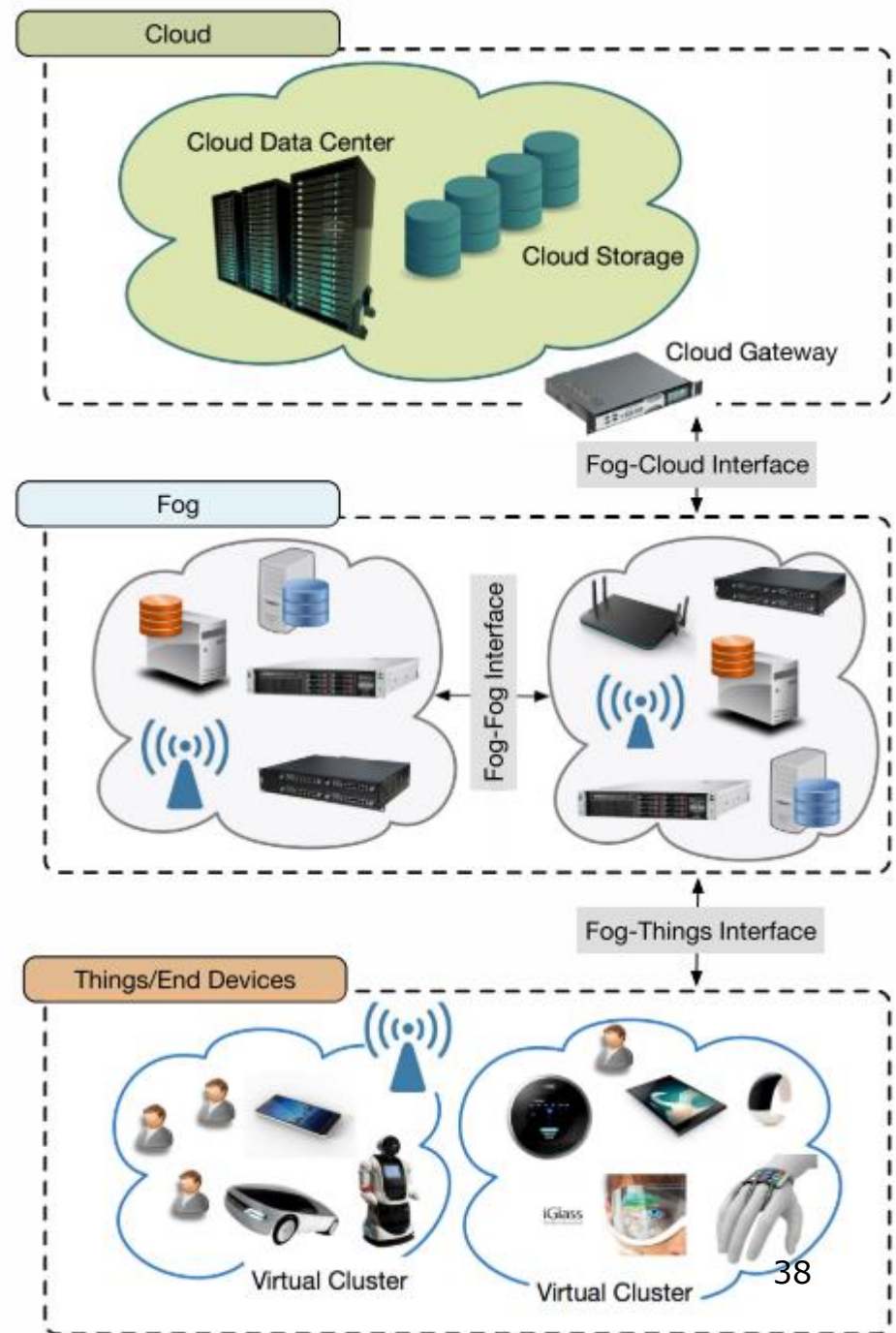
# Tipos de SDs

## ☐ COMPUTAÇÃO EM NUVEM



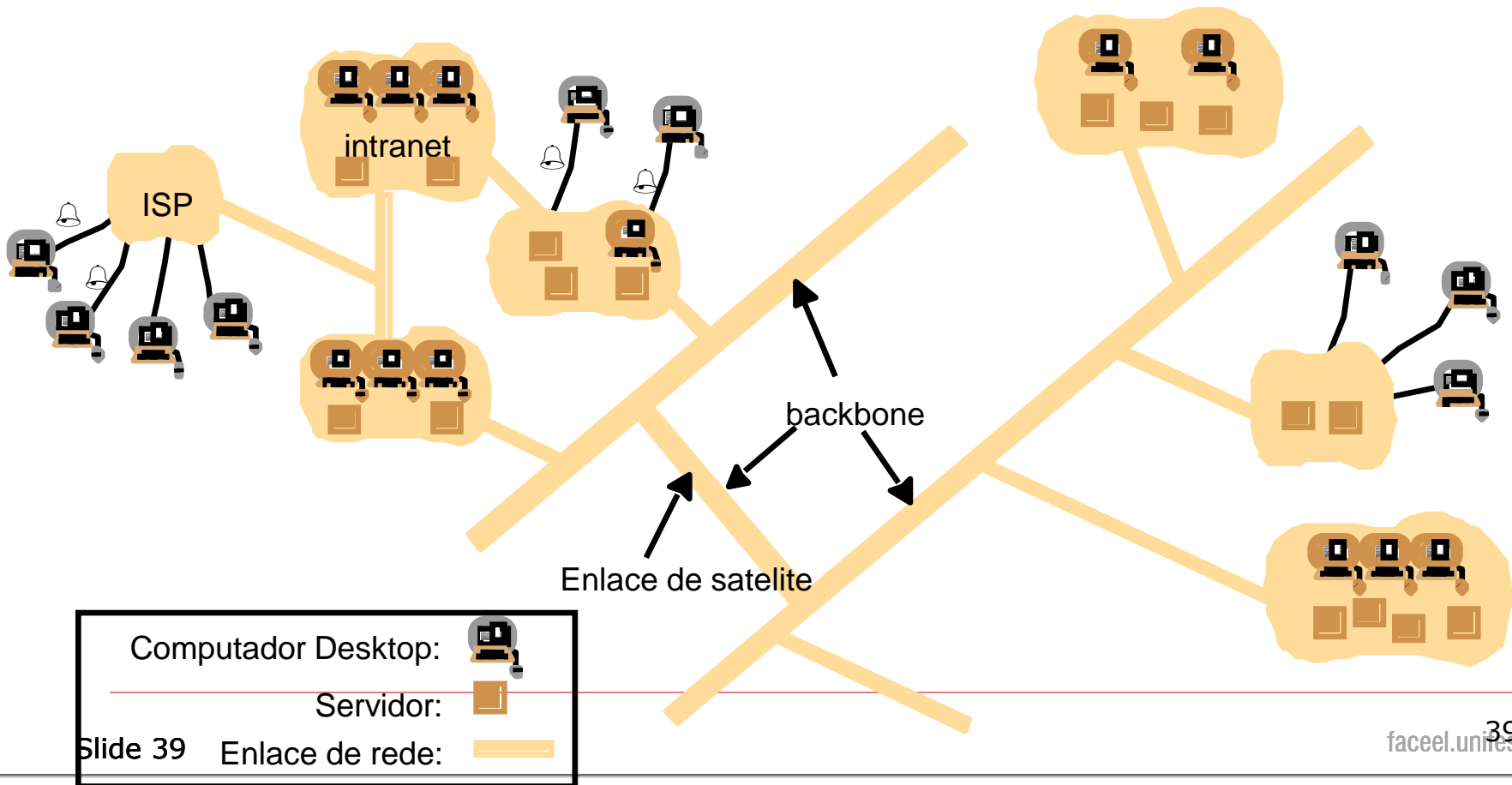
# Tipos de SDs

## ❑ COMPUTAÇÃO EM NÉVOA



# Tipos de SDs

## INTERNET



# Tipos de SDs

---

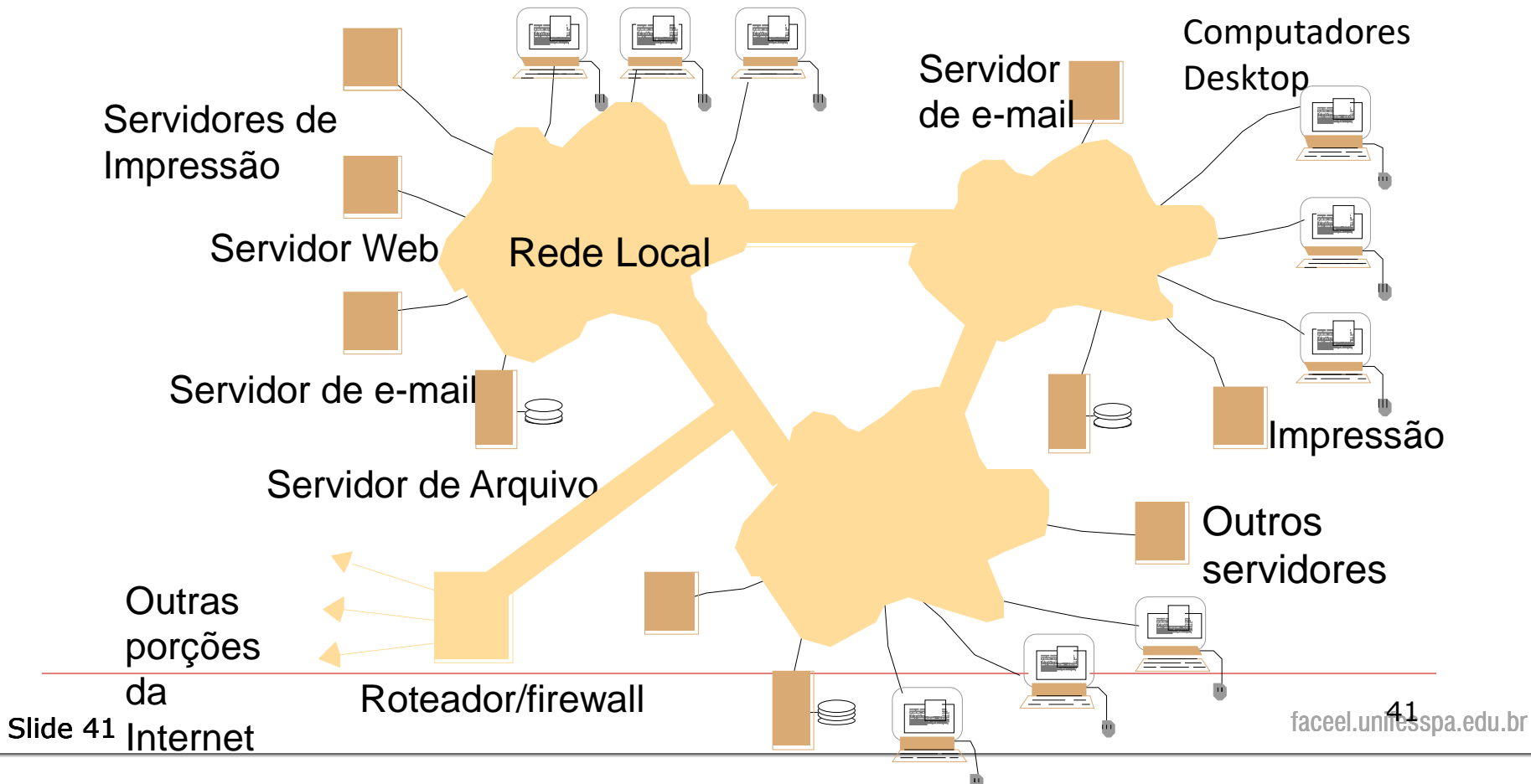
## □ INTERNET

- A Internet é um conjunto de redes de computadores, de muitos tipos diferentes, interligadas.
- A Internet é um sistema distribuído muito grande. Ela permite que os usuários, onde quer que estejam, façam uso de serviços como a *World Wide Web*, e-mail e transferência de arquivos



# Tipos de SDs

## ❑ INTRANET



# Exemplos de Aplicações Distribuídas

---

Google



WhatsApp



YAHOO!



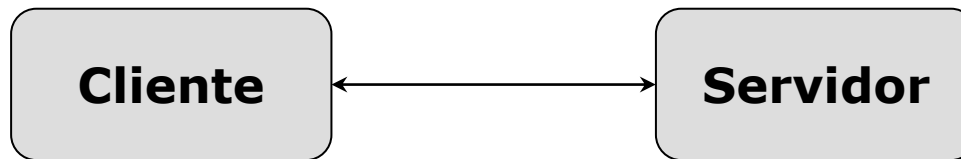
# Desafios - exemplo

---

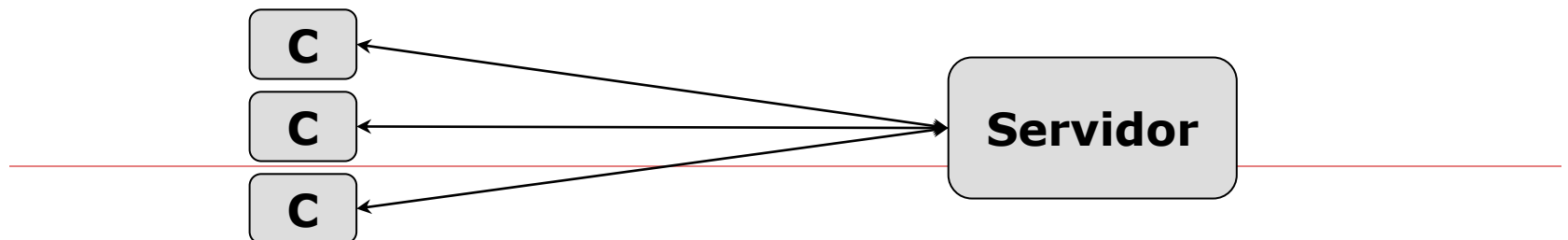
- ❑ Imagine que você criou um sistema de rede social chamado ***Tuits***, para compartilhamento de mensagens curtas na *Internet*
- ❑ Você implantou o sistema em **um servidor** e divulgou para os amigos da universidade usarem



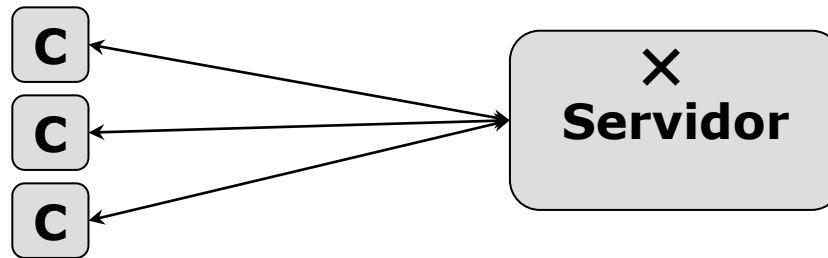
- ❑ **Desafio:** permitir a **comunicação** entre cliente e servidor
  - *Troca de mensagem*, Chamada remota de procedimentos (*RPC*)



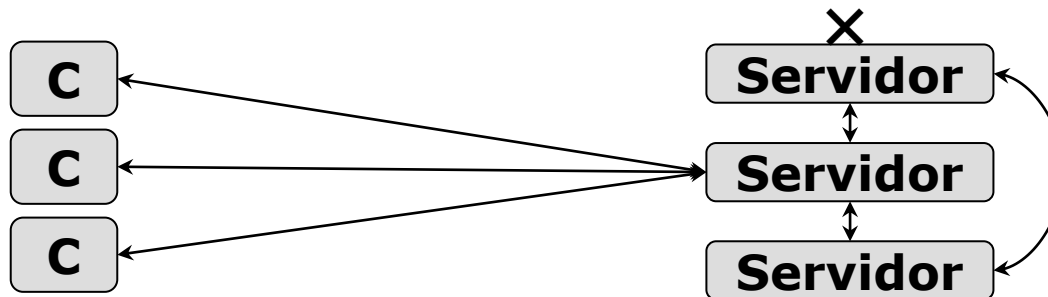
- ❑ **Desafio:** permitir o acesso de **múltiplos usuários**
  - *Multithreading*, programação assíncrona



- ❑ **Desafio:** se o servidor falhar, o sistema fica **indisponível**
  - Se o disco falhar permanentemente, todos os **dados serão perdidos**

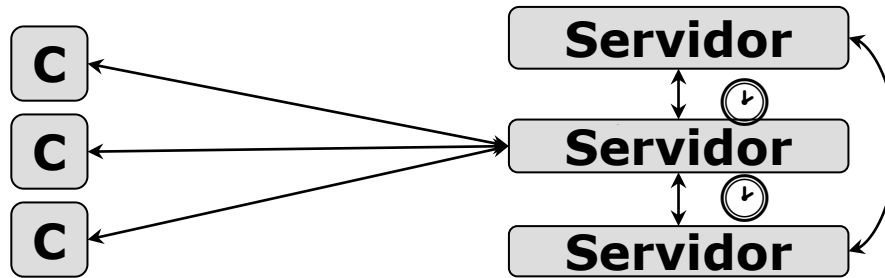


- ❑ **Solução:** **replicação** (*réplicas secundárias ou backup*)



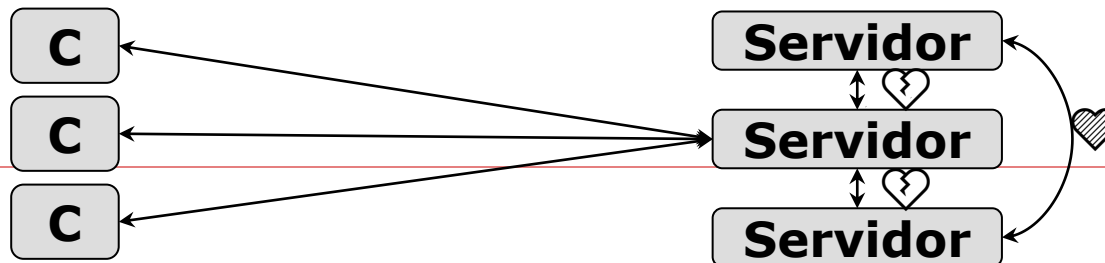
## ❑ **Desafio: atrasos na comunicação** entre os servidores

- A réplica primária falhou ou a conexão está lenta?

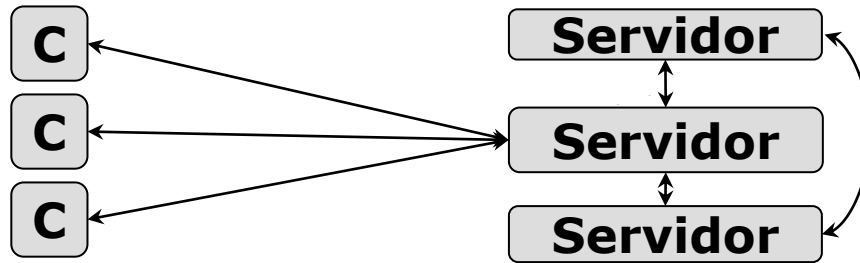


## ❑ **Solução: detector de falhas**

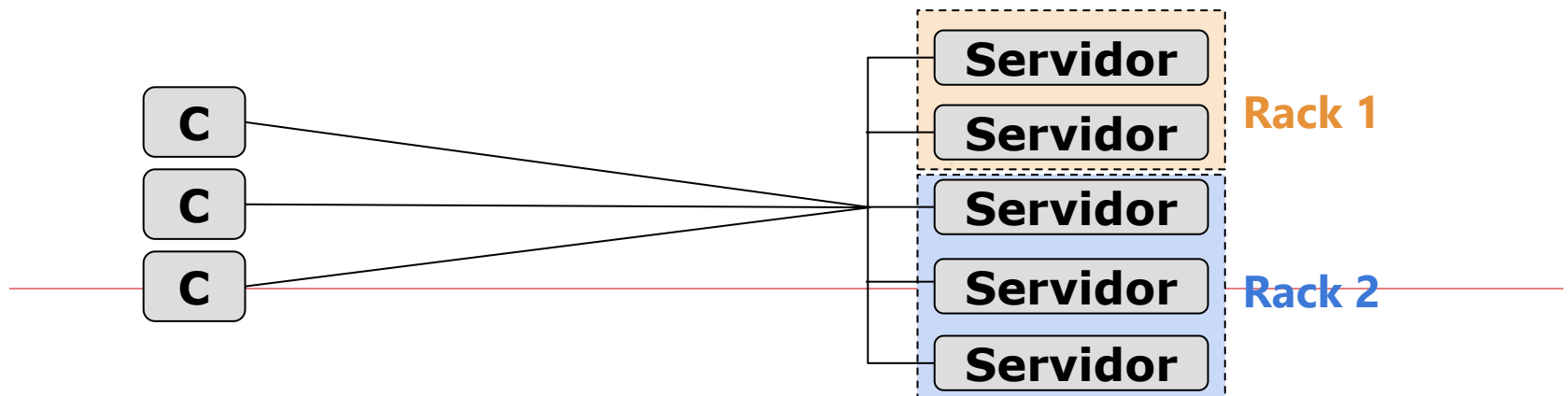
- *Heartbeat* periódico. Pode haver engano se houver atraso atípico



- ❑ **Desafio:** seu sistema popularizou e ficou **sobrecarregado**

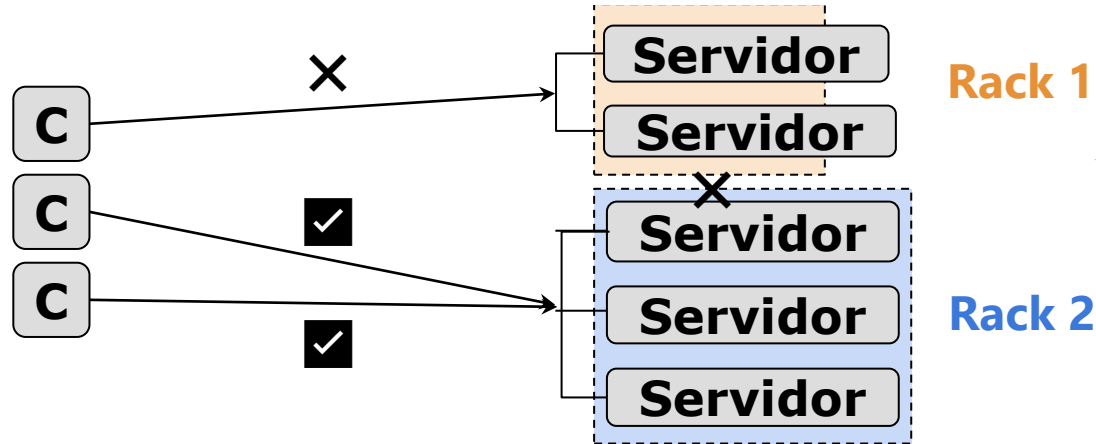


- ❑ **Solução:** replicação ativa e **balanceamento de carga**



## ❑ **Desafio:** falha na comunicação entre grupos de servidores

- **Split brain!** Pode gerar inconsistência nos dados

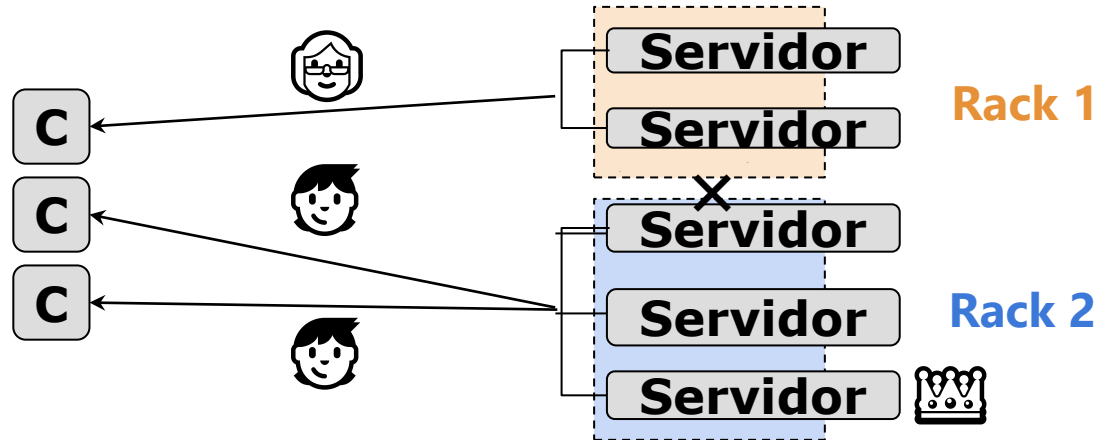



## ❑ **Solução: Quorum**

- Só aplica ações se maioria de servidores confirmarem
- Apenas quem acessa **Rack 2** teria serviço disponível
- Precisa de  $2f + 1$  servidores para tolerar  $f$  falhas

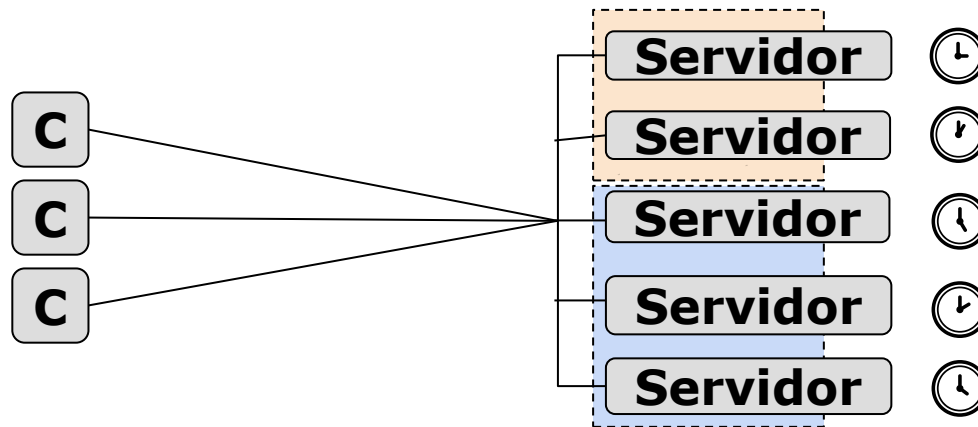


- ❑ **Desafio:** parte dos clientes podem ler dados desatualizados



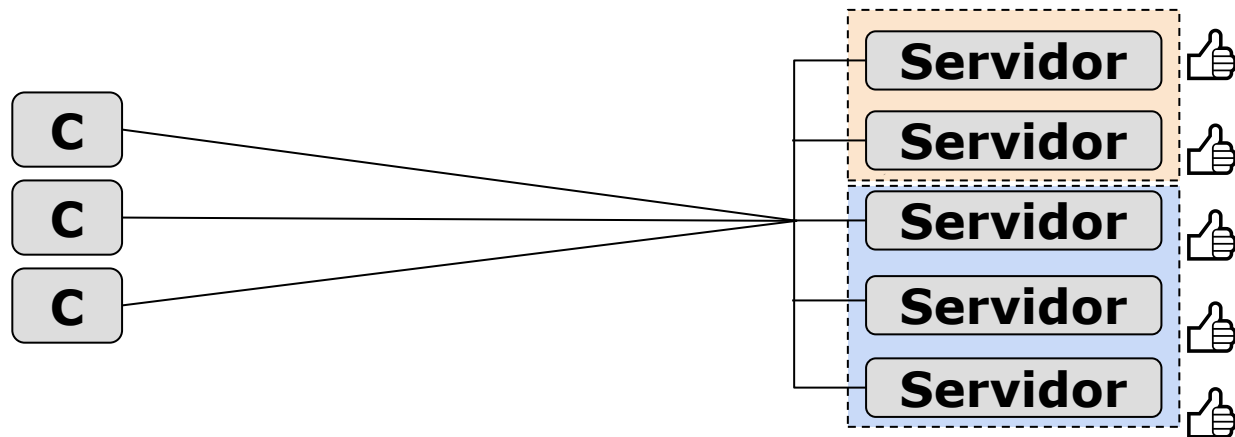
- ❑ **Solução: eleição de líder**
  - Controla e coordena replicação nos servidores
  - Só retorna valor a cliente se o **líder**  garantir que está consistente na maioria dos servidores

- ❑ **Desafio:** os relógios dos servidores marcam horas diferentes
  - Como ordenar os eventos que chegam nos diferentes servidores?



- ❑ **Solução 1: sincronização de relógios**
    - Ajusta relógios periodicamente (atrasos dificultam)
  - ❑ **Solução 2: relógios lógicos**
    - Contador incremental de eventos indica ordem
-

- ❑ **Desafio: tolerar falhas** e obter **consenso** dos servidores
  - Concordar em quais dados devem ser armazenados, na sua ordem e quando torná-los visíveis aos clientes



- ❑ **Solução:** aplicar algoritmos de **consenso distribuído**
    - Junta tudo que falamos e mais um pouco
    - Nós vamos abordar esses tópicos durante a disciplina...
-