



# UNIDADE I

---

## Sistemas Distribuídos

Prof. Me. Michel Fernandes

# Sistemas distribuídos – Definição

- Um **sistema distribuído** é um conjunto de computadores interligados via rede, mas, para o usuário final das aplicações, executadas por meio desse sistema, aparenta ser um sistema único.
- Sistemas que não dependem apenas de uma máquina.
- Definem a forma como os diversos objetos distribuídos se comunicam e interagem entre si.

Dois aspectos:

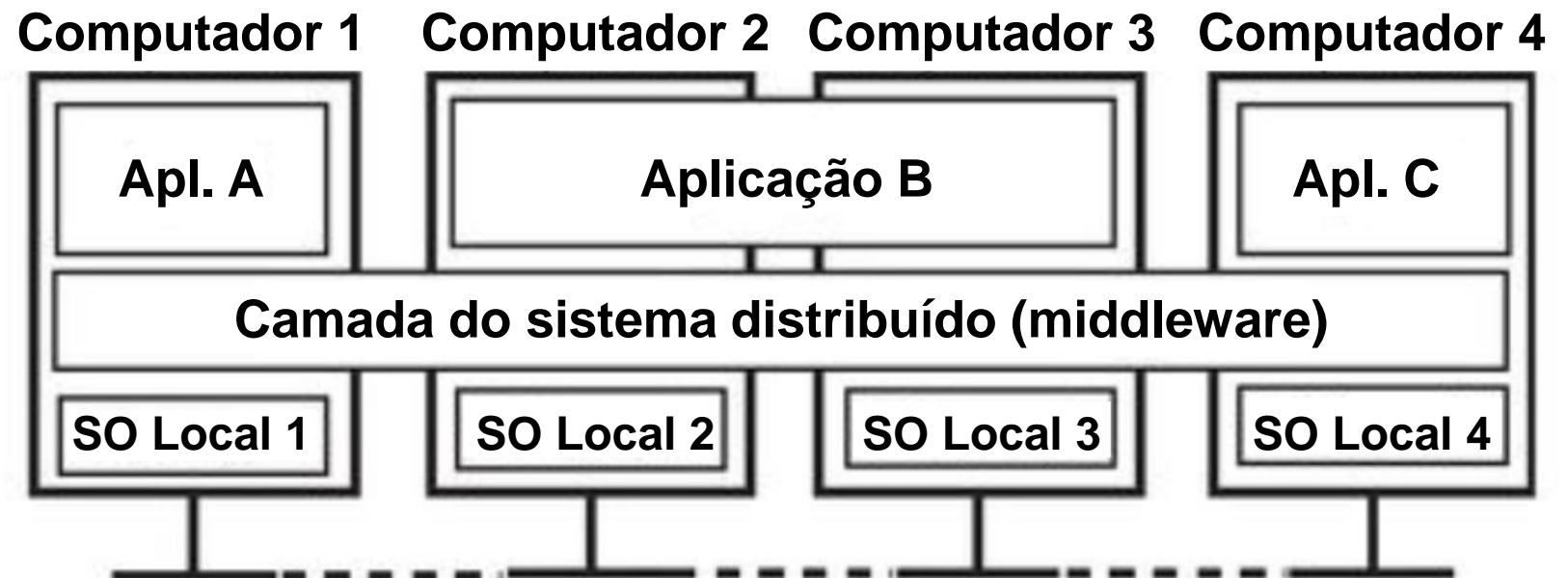
- Computadores independentes;
- Um único sistema middleware.

# Sistemas distribuídos – Camadas

- Middleware funciona como uma camada de tradução para interligar o sistema operacional com os programas.
- Fator essencial para o bom funcionamento de aplicações distribuídas.

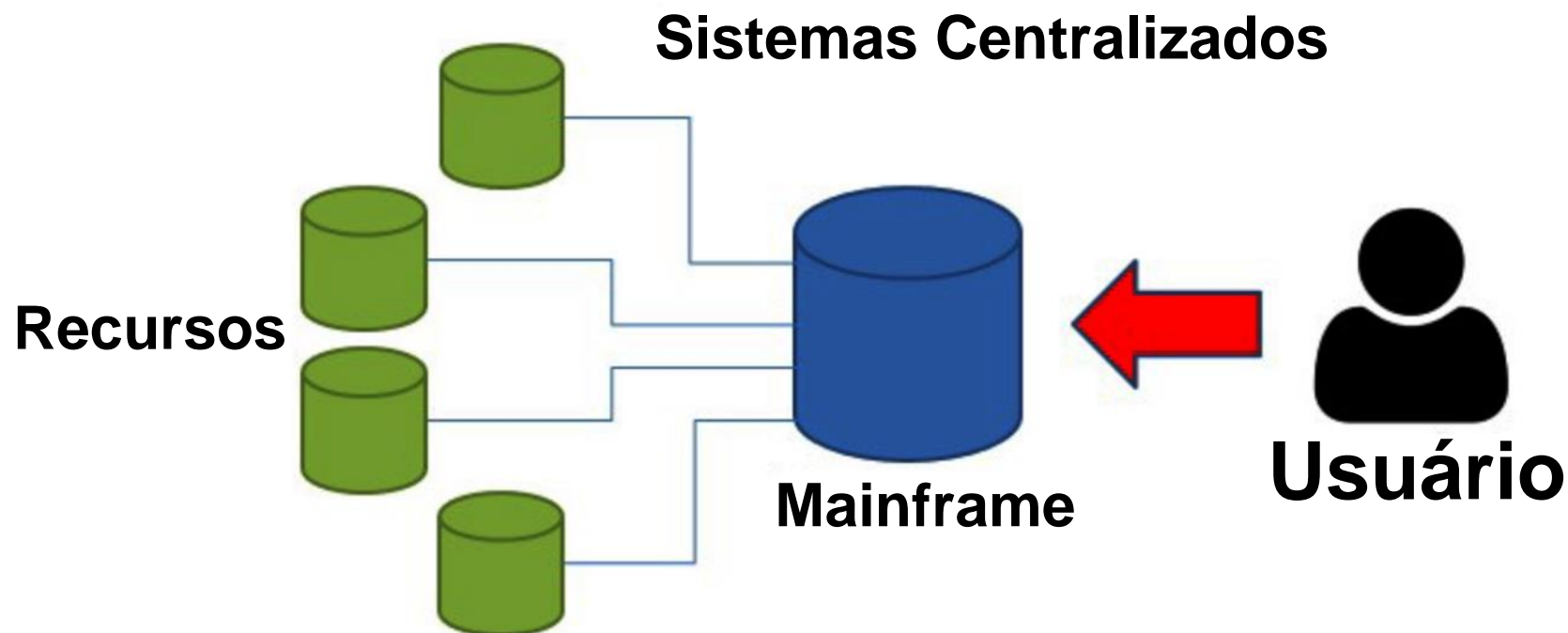
Exemplos de middleware:

- Os serviços web ou web services;
- RMI Java e
- CORBA.



# Sistemas centralizados

- Conjunto de máquinas que utiliza seus recursos e com uma máquina centralizadora (servidor / mainframe).
- Usuário acessa um mainframe que o redireciona para os recursos necessários, assim como servidores de aplicação ou banco de dados.
- Maior desvantagem: **falta de escalabilidade** e produtividade.



# Sistemas distribuídos – Objetivos

- Disponibilidade alta e facilidade de acesso ao sistema e a todos os seus recursos.
- Importante: o usuário deve desconhecer a distribuição de recursos do sistema.
- O sistema deve ser aberto e heterogêneo.
  
- Fazer o link entre usuários e recursos:
  - Compartilhamento de recursos;
  - Segurança;
  - Reduzir a comunicação indesejada.

# Sistemas distribuídos – Escalabilidade

- Propriedade que mede a capacidade do sistema em lidar facilmente com uma quantidade crescente de trabalho.

Três medidas para medir a escalabilidade:

- Número de processos e/ou usuários (escalabilidade de tamanho);
- Distância máxima entre os nós (escalabilidade geográfica);
- Número de domínios administrativos (escalabilidade administrativa).

# Sistemas distribuídos – Transparência

- Formas de ocultação para os usuários e desenvolvedores de aplicação.
- Deseja-se que o sistema seja percebido como único, em vez de uma coleção de partes independentes.
- É uma das métricas de sucesso de um SD.
- 7 formas de transparência.




Transparência	Descrição
Acesso	Oculta diferenças na representação de dados e no modo de acesso a um recurso
Localização	Oculta o lugar em que um recurso está localizado
Migração	Oculta que um recurso pode ser movido para outra localização
Relocação	Oculta que um recurso pode ser movido para uma outra localização enquanto em uso
Replicação	Oculta que um recurso é replicado
Concorrência	Oculta que um recurso pode ser compartilhado por diversos usuários concorrentes
Falha	Oculta a falha e a recuperação de um recurso

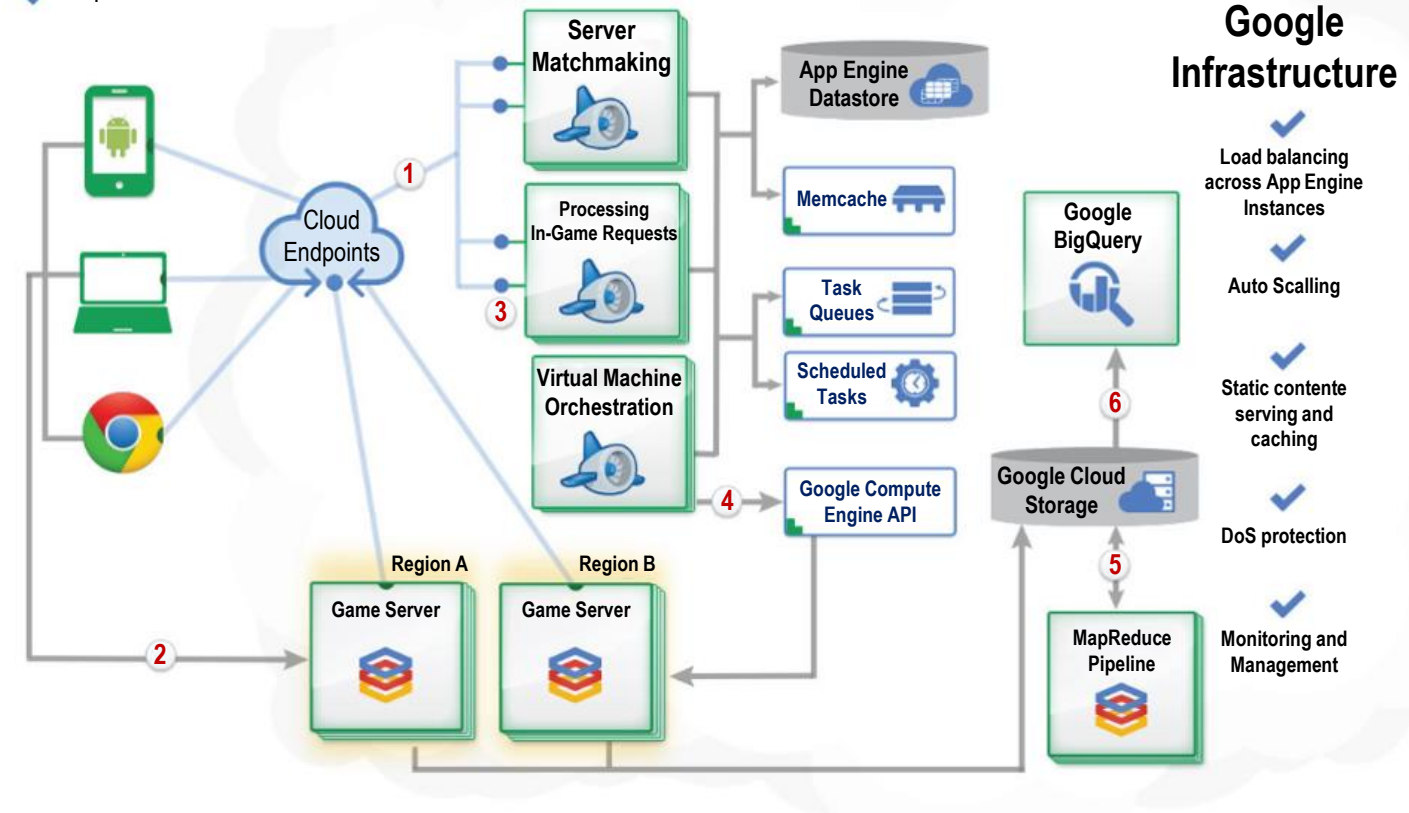
# Sistemas distribuídos – Exemplos

## Jogos on-line

- Vários tipos de “computadores”
- Vários tipos de papel (role)
- Várias formas de comunicação
- Várias camadas lógicas (abstrações)

### Dedicated Server Gaming Solution on the Google Cloud Platform

-  Your Application Code running on Google App Engine, Google Compute Engine, and Client Devices
-  Google Cloud Platform Services
-  Capabilities Included



Fonte: Adaptado de: <https://cloud.google.com/files/DedicatedGameServerSolution.pdf>



# Sistemas distribuídos – Exemplos

- Jogos on-line
- Pesquisas na web (Google)
- Serviços de streaming (YouTube, Netflix, Amazon Prime)
- Bancos de dados distribuídos
- Cloud computing
- WhatsApp
- Redes sociais
- Diversas aplicações pela web

# Sistemas distribuídos – Algoritmos

- Nenhuma máquina tem informação completa do estado do sistema.
- As máquinas tomam decisões com as informações locais.
- A falha no algoritmo não “quebra” todo o sistema.
- Não existe uma suposição de um relógio global.

# Sistemas distribuídos – Técnicas de escalabilidade

- Aumenta a disponibilidade dos recursos.
- Balanceamento de carga.
- Oculta a latência de comunicação caso haja grande dispersão geográfica.

# Interatividade

Uma característica fundamental dos sistemas distribuídos é a transparência, que é uma forma de ocultar para os usuários e desenvolvedores de aplicação a localização de um recurso.

Qual tipo de transparência ocorre quando o sistema oculta a falha e a recuperação de um recurso?

- a) Transparência de acesso.
- b) Transparência de localização.
- c) Transparência de migração.
- d) Transparência de falha.
- e) Transparência de concorrência.

# Resposta

Uma característica fundamental dos sistemas distribuídos é a transparência, que é uma forma de ocultar para os usuários e desenvolvedores de aplicação a localização de um recurso.

Qual tipo de transparência ocorre quando o sistema oculta a falha e a recuperação de um recurso?

- a) Transparência de acesso.
- b) Transparência de localização.
- c) Transparência de migração.
- d) **Transparência de falha.**
- e) Transparência de concorrência.

# Tipos de sistemas distribuídos

- Computação em cluster (cluster computing);
- Computação em grade (grid computing);
- Computação em nuvem (cloud computing);
- Sistemas de informação distribuídos;
- Sistemas distribuídos pervasivos.

# Computação em cluster

- Característica homogênea entre os nós.

## Hardware:

- Conjunto de PCs ou estações de trabalho semelhantes.
- Conexão entre os hardwares: rede local (LAN).

## Software:

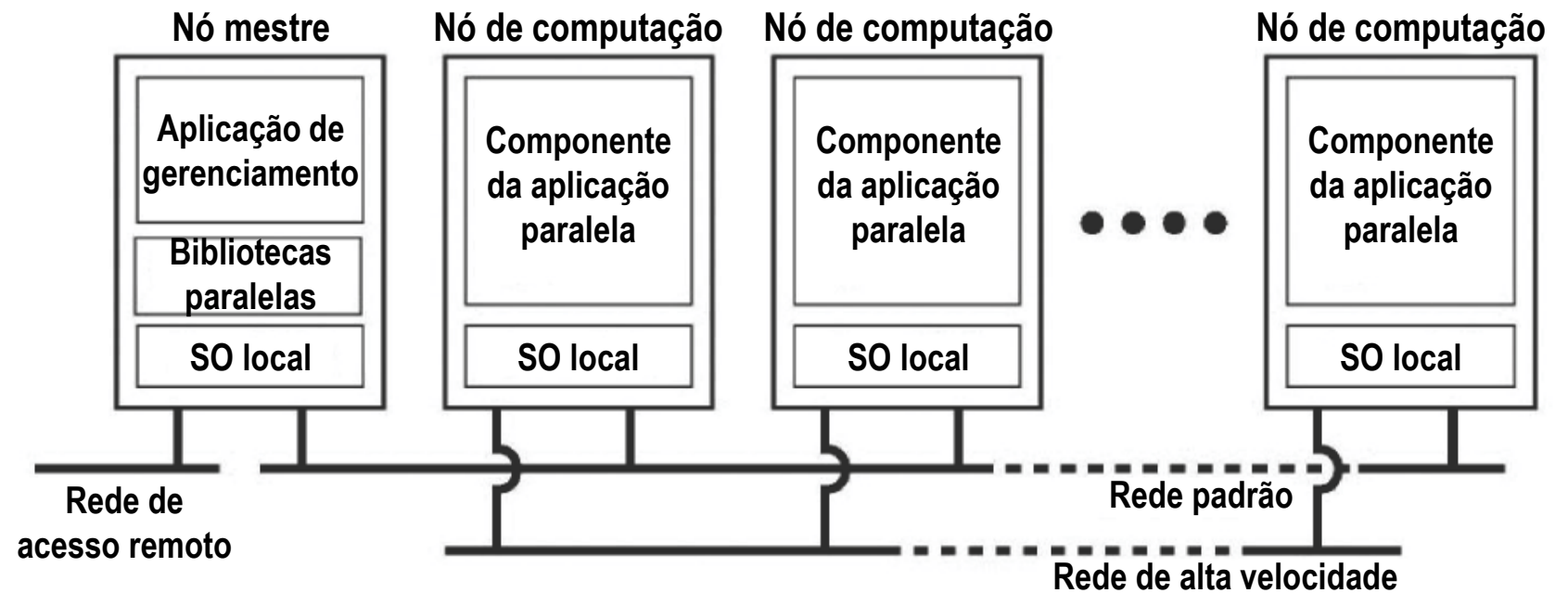
- Normalmente, as máquina possuem o mesmo SO.
- Geralmente, um único programa executado em paralelo.
  - Aplicada frequentemente para computação paralela.
  - Forte acoplamento entre os nós.

# Computação em cluster

- Cluster utilizando sistemas operacional Linux

Nó mestre é responsável por:

- Alocar tarefas aos nós, organizar a fila de tarefas e interface com usuários;
- Prover transparência de localização e migração.





# Computação em grade

- Grid computing

Característica da heterogeneidade:

- Hardware de diferentes organizações são reunidos e dispersos entre elas.
- Para permitir a colaboração de um grupo de pessoas ou instituições.

Exemplo: PlanetLab (<http://www.planet-lab.org>)

- Rede de pesquisa mundial para criar novos serviços de rede.

# Computação em nuvem

- Cloud computing

A computação em nuvem permite o uso de um recurso de computação:

- como uma máquina virtual (VM);
  - um armazenamento;
  - ou uma aplicação.
- Semelhantes ao consumo da eletricidade, pois é um serviço terceirizado.
  - Em vez de ter que construir e manter infraestruturas de computação em casa ou na empresa.
  - Exemplo: uso para *backup*.

# Sistemas de informação distribuídos

Sistemas corporativos para integrar diversas aplicações onde a interoperabilidade se mostrou “penosa”:

- Sistemas de processamento de transações.
- Integração de aplicações empresariais.

# Sistemas pervasivos

- **Instabilidade** é o comportamento esperado desses sistemas.

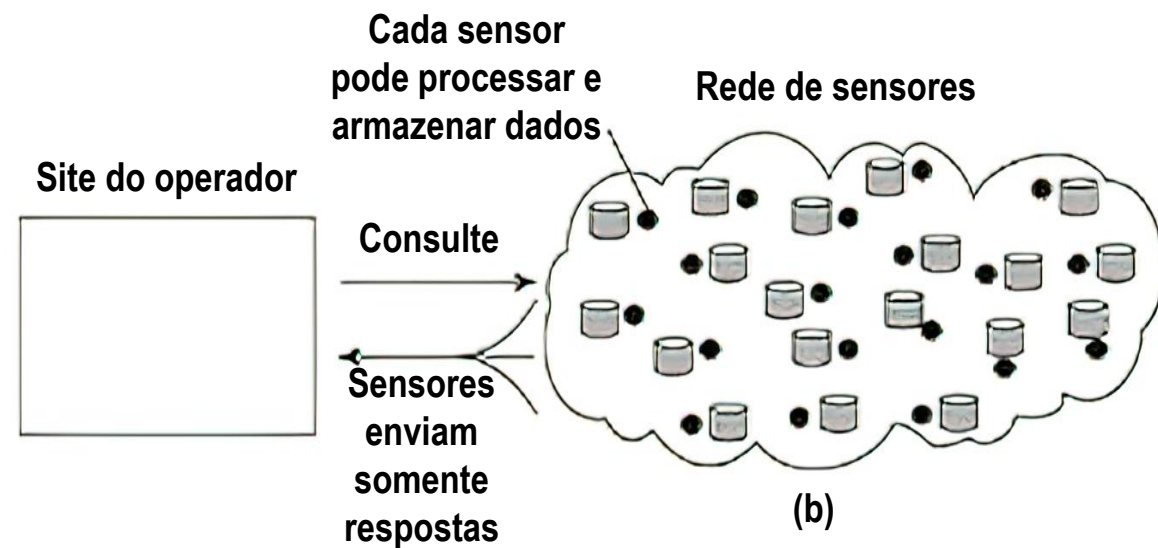
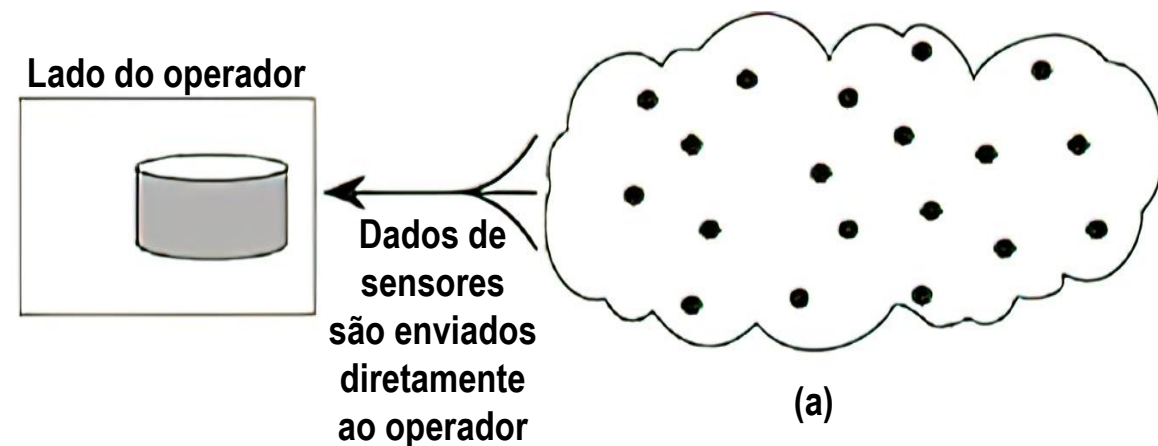
Dispositivos de computação móveis e embarcados pequenos.

- Alimentação por bateria;
- Mobilidade;
- Conexão sem fio.

# Sistemas pervasivos – Redes de sensores sem fio

Exemplo de SD onde os sensores possuem aplicações que comunicam-se entre si:

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



# Interatividade

Com a evolução dos sistemas computacionais e de redes de comunicação, diversas empresas não possuem mais um datacenter dentro das suas instalações. Elas contratam empresas provedoras de serviços de processamento ou armazenamento para executar suas aplicações. Qual o nome desse tipo de sistema distribuído?

- a) Computação em cluster.
- b) Computação em grid.
- c) Sistemas pervasivos.
- d) Computação em nuvem.
- e) Internet das Coisas.

## Resposta

Com a evolução dos sistemas computacionais e de redes de comunicação, diversas empresas não possuem mais um datacenter dentro das suas instalações. Elas contratam empresas provedoras de serviços de processamento ou armazenamento para executar suas aplicações. Qual o nome desse tipo de sistema distribuído?

- a) Computação em cluster.
- b) Computação em grid.
- c) Sistemas pervasivos.
- d) **Computação em nuvem.**
- e) Internet das Coisas.

# Estilos arquitetônicos

Um estilo arquitetônico é determinado por meio dos:

- **Componentes:** unidade modular com interfaces requeridas e fornecidas bem definidas, que é substituível dentro de seu ambiente.
- **Conexões:** o modo como os componentes estão ligados.
- **Dados intercambiados:** forma da troca dos dados entre componentes (repositório compartilhado?).
- **Formas de configuração:** maneiras como os componentes são configurados (em tempo de execução?).

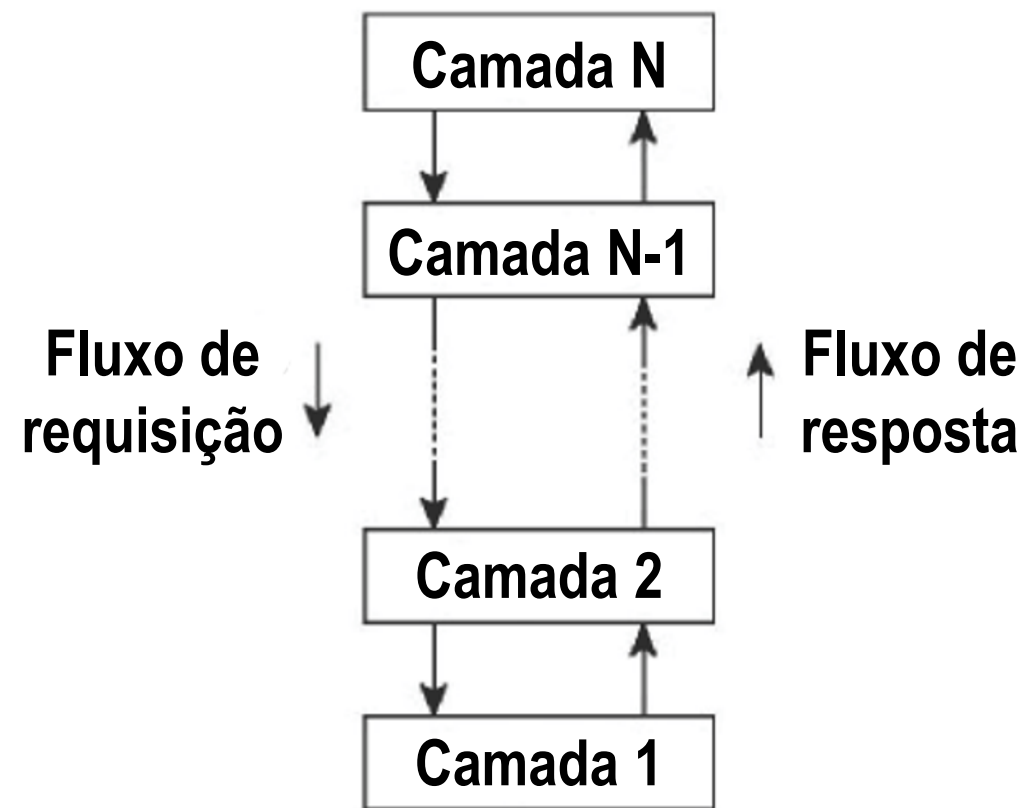


# Tipos de estilos arquitetônicos

- Arquiteturas de camadas;
- Arquiteturas baseadas em objetos;
- Arquiteturas centradas em dados;
- Arquiteturas baseadas em eventos.

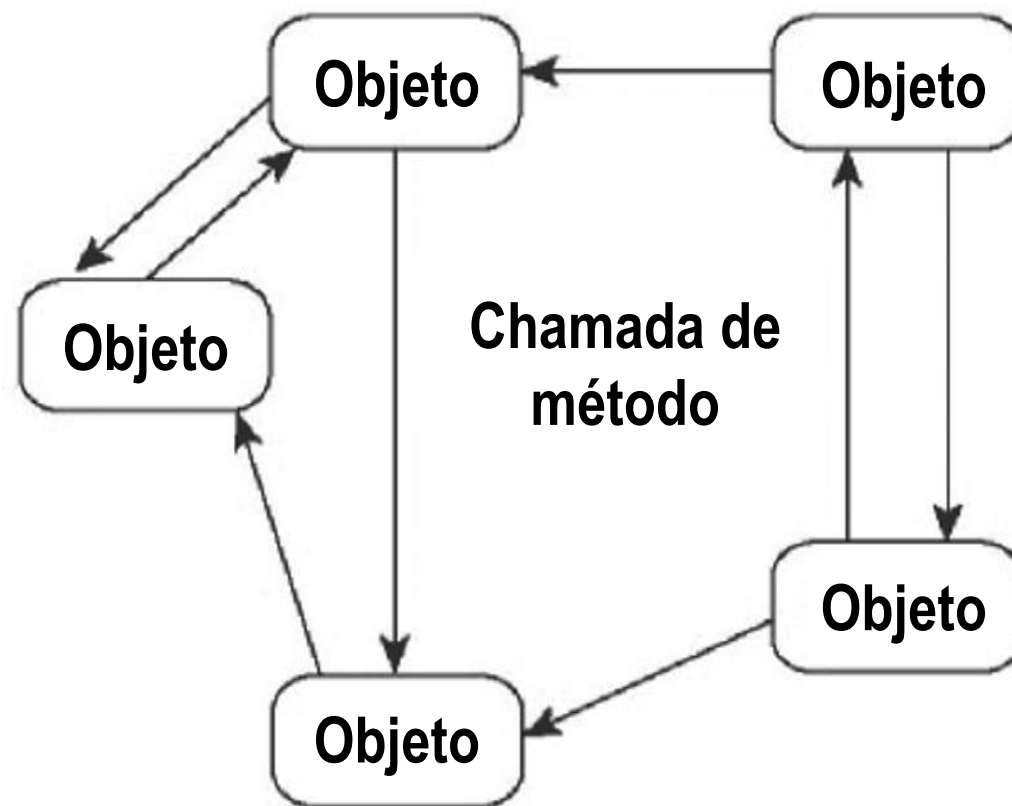
# Arquitetura em camadas

- Ideia semelhantes à estrutura de redes TCP/IP.
- Componentes são organizados em camadas.
- Componente da camada N tem permissão para chamar componentes da camada N-1.



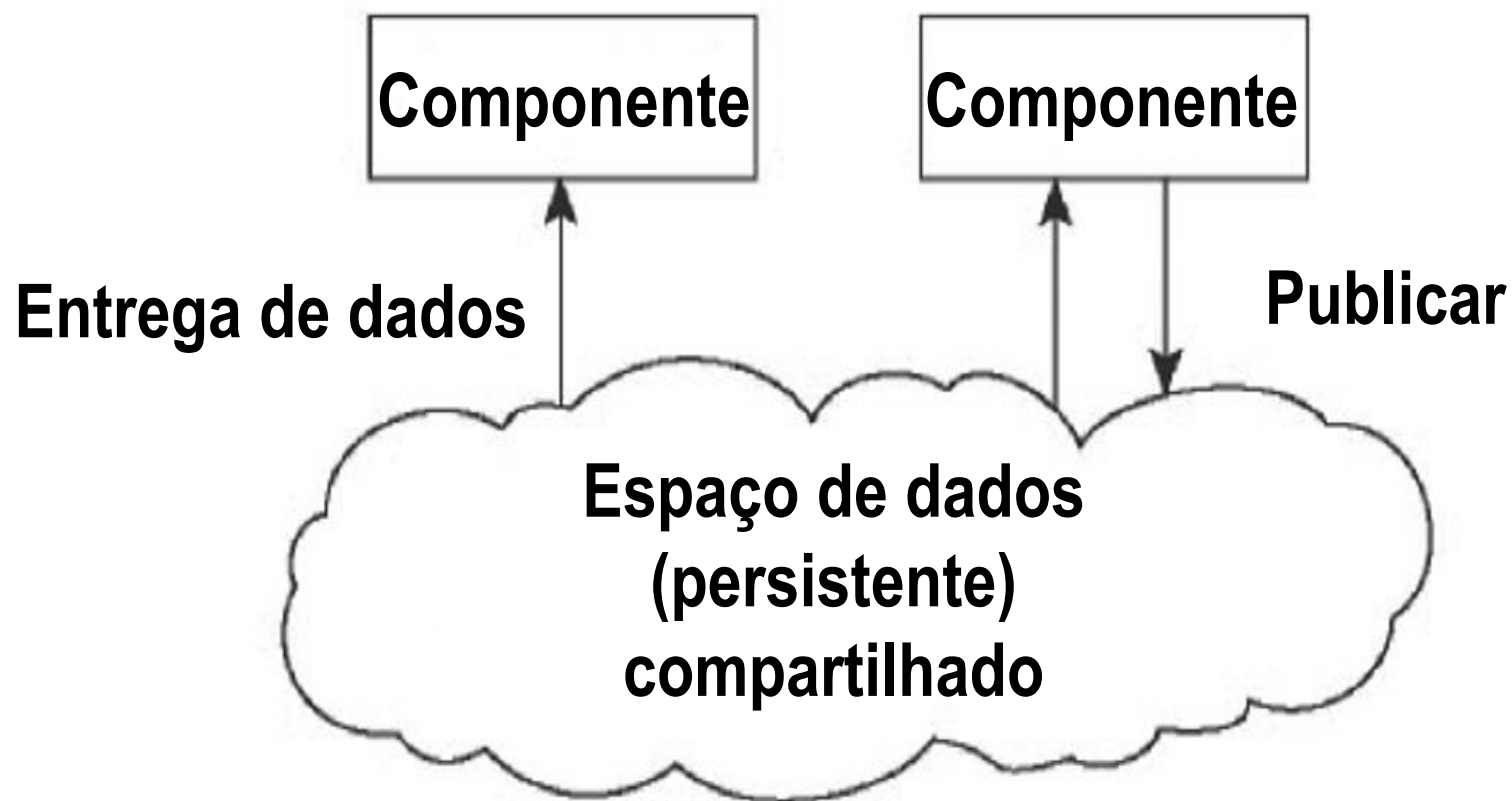
# Arquitetura baseada em objetos

- Objetos são os componentes.
- Objetos são conectados a outros por chamada remota.
- Amplamente explorado em sistemas cliente-servidor.



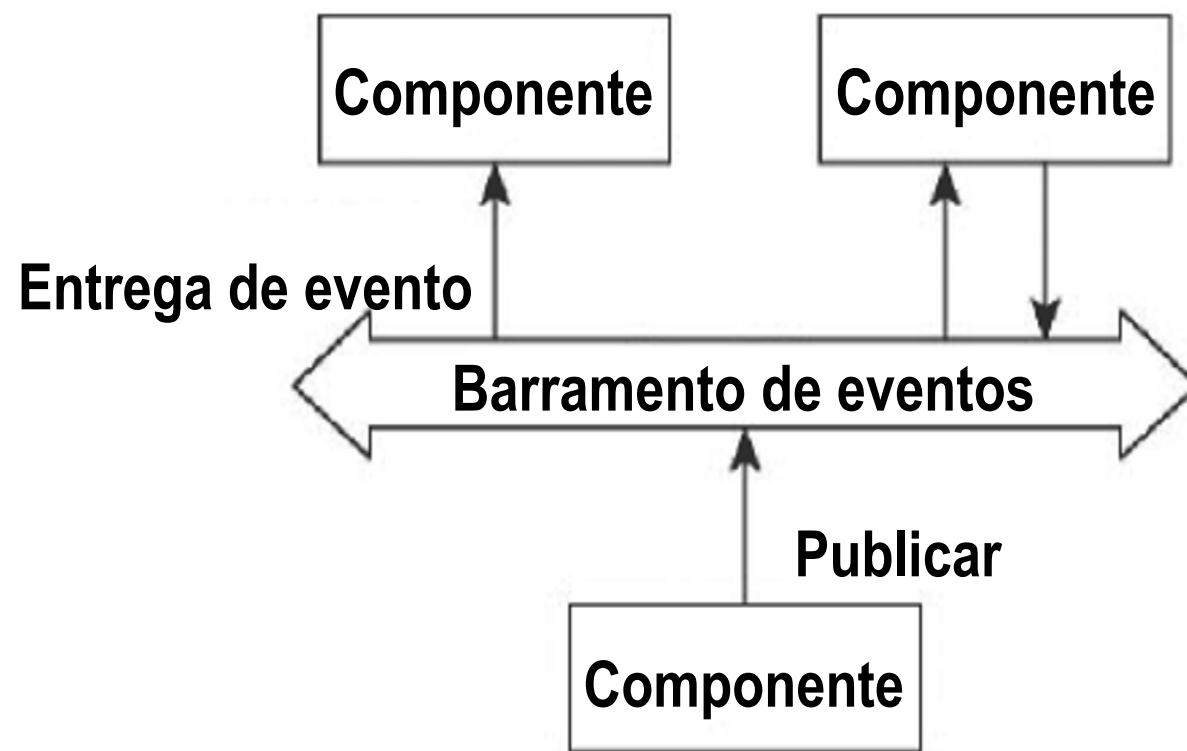
# Arquitetura centrada em dados

- Componentes se comunicam por meio de um repositório comum, como se fosse uma “caixa postal”.
- Fracamente acoplado.



# Arquitetura baseada em eventos

- Sistema publicar-subscriver (*publish-subscribe*).
- Componentes publicam eventos e certificam que somente os que subscreveram recebem estes eventos.
- Fracamente acoplados: não invocam explicitamente um ao outro.
- Exemplo: Apache Kafka.



# Arquitetura de sistemas

Perguntas importantes para definição da arquitetura:

**Como os Sistemas Distribuídos são organizados na prática?**

**Onde são executados os componentes de software?**

**Como eles se comunicam?**

**Há um elemento central?**

# Arquiteturas de sistemas

## Arquiteturas centralizadas

- Cliente-servidor: Netflix, site de notícias, mobile banking etc.

## Arquiteturas descentralizadas

- Sistemas Peer-to-Peer, como o Chord.

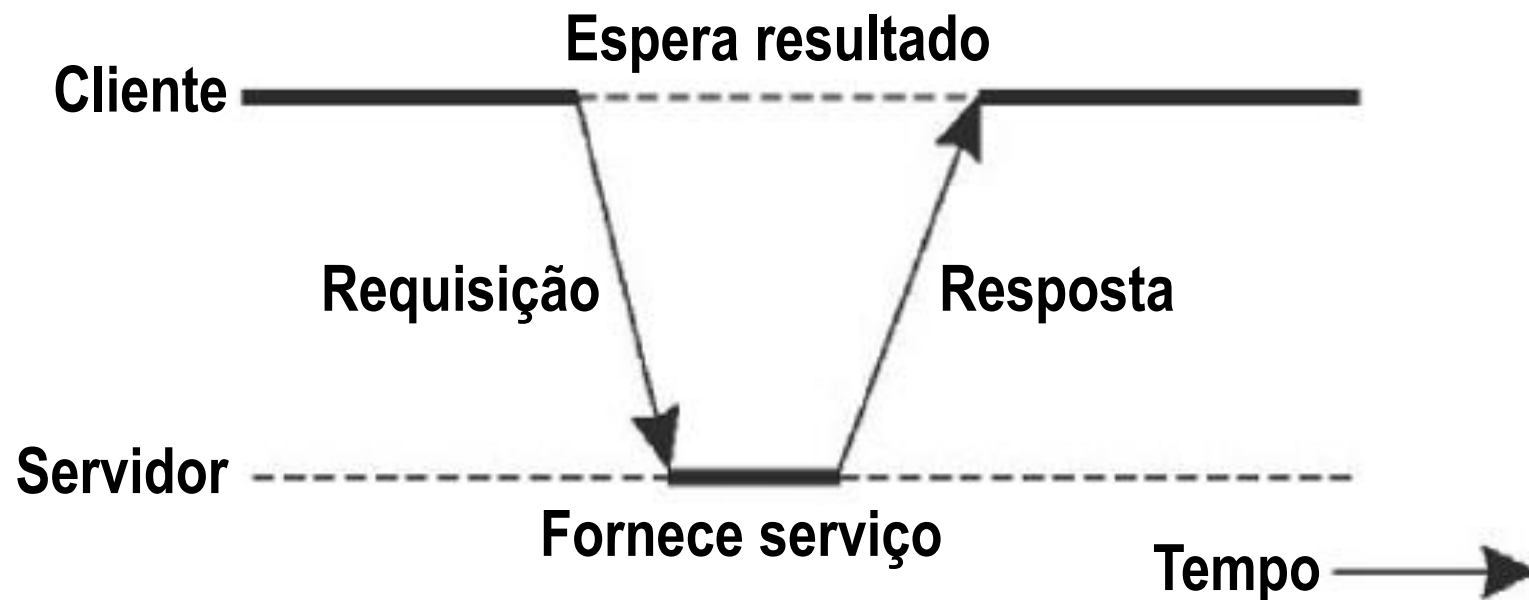
## Arquiteturas híbridas

- Sistemas Peer-to-Peer, como o BitTorrent, Skype e WhatsApp.

# Arquitetura cliente-servidor

Processos são divididos em dois grupos:

- **Servidor:** processo que implementa um serviço específico.
- **Cliente:** processo que requisita um serviço ao servidor.
- Forma de interação: requisição-resposta.

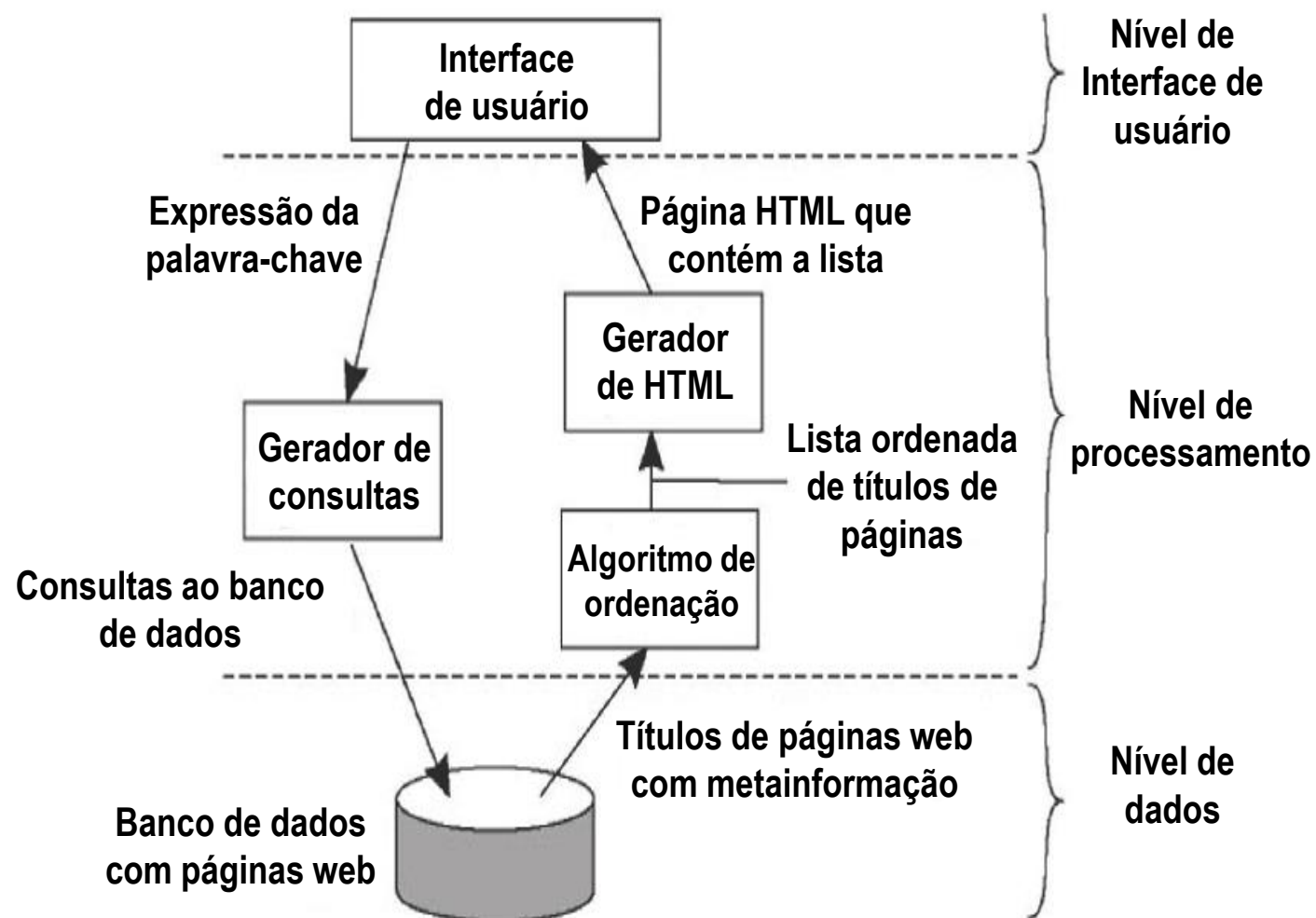




# Arquitetura centralizada com camadas de aplicação

Considerando muitas aplicações e a sua escalabilidade:

- Nível de interface de usuário;
- Nível de processamento;
- Nível de dados.



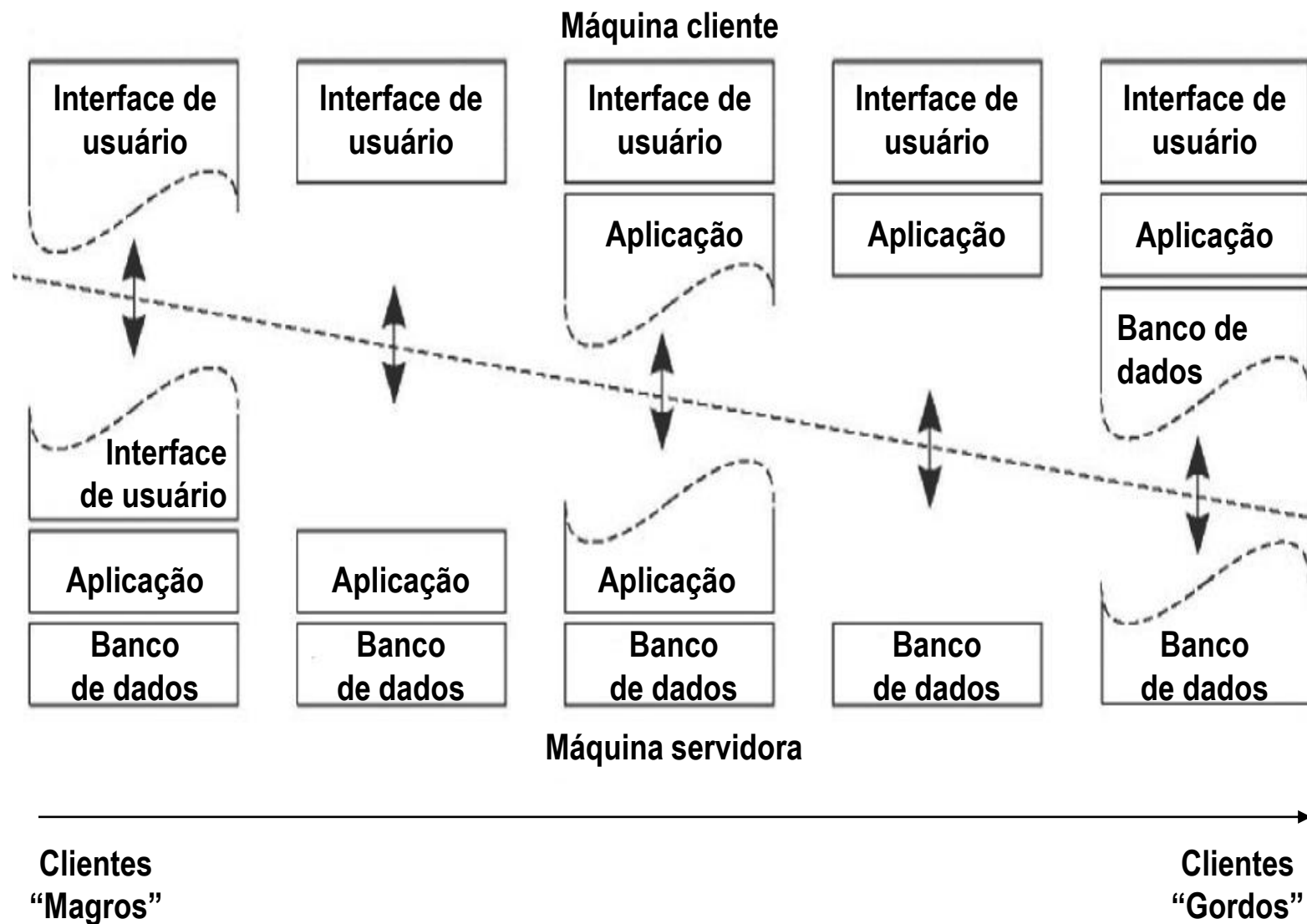
# Arquitetura centralizada com arquiteturas multidividadas

Gerenciamento de sistema:

- Clientes gordos (fat clients);
- Clientes magros (thin clients).

Distinção clara:

- Arquitetura de duas divisões (cliente e servidor).

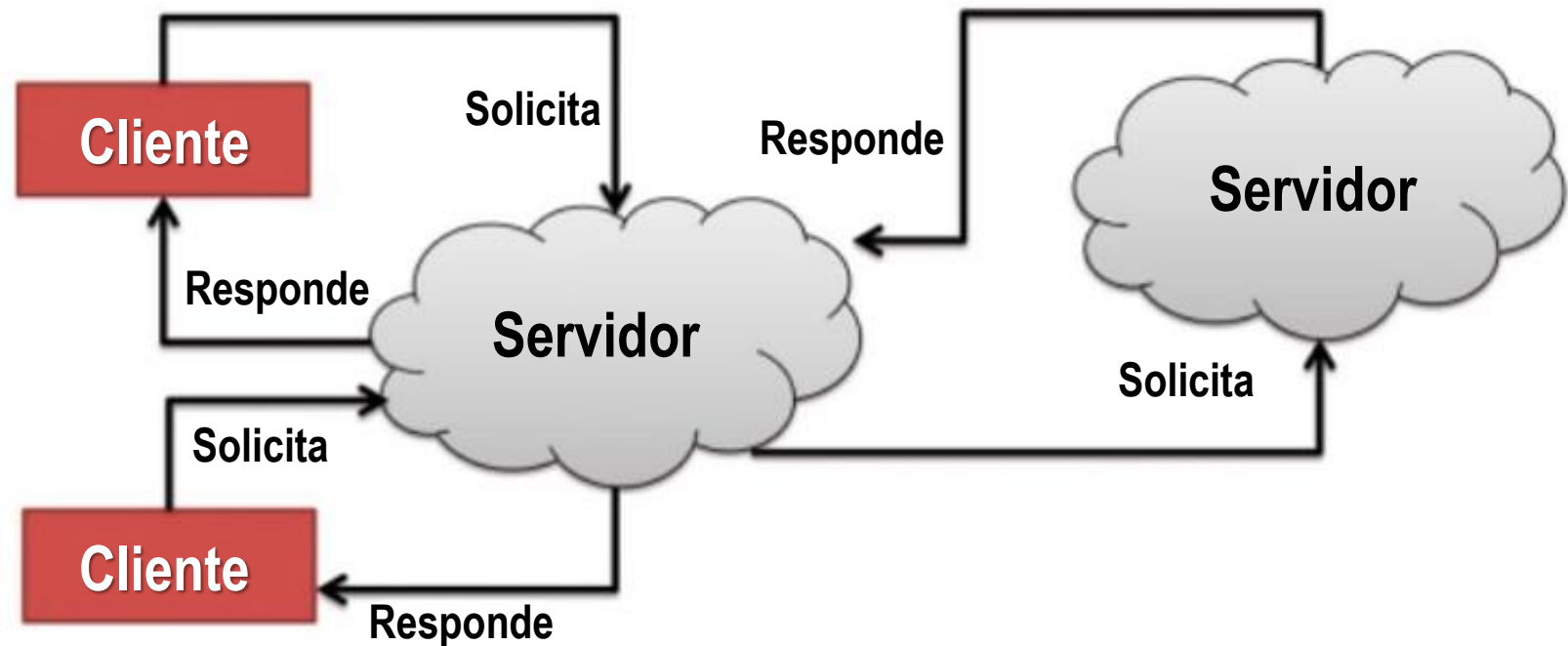


# Arquitetura centralizada com arquiteturas multidividadas

Arquitetura de três divisões:

1. Cliente
2. Servidor
3. Servidor que pode agir como cliente

- Recebe e faz a requisição.



# Interatividade

Na arquitetura cliente-servidor, qual o modelo de interação entre os componentes?

- a) Chamadas de sistemas.
- b) Chamadas de métodos remotos.
- c) Solicitação e resposta.
- d) Comunicação aleatória.
- e) Publicação e subscrição.

# Resposta

Na arquitetura cliente-servidor, qual o modelo de interação entre os componentes?

- a) Chamadas de sistemas.
- b) Chamadas de métodos remotos.
- c) **Solicitação e resposta.**
- d) Comunicação aleatória.
- e) Publicação e subscrição.

# Arquiteturas descentralizadas

- Cliente-servidor possuem duas distribuições: vertical e horizontal.

## Distribuição vertical

- Componentes logicamente diferentes em máquinas diferentes.
- Cada máquina vai executar um conjunto específico e predeterminado de funções.
- As funções de cada um são bem definidas.

# Arquiteturas descentralizadas

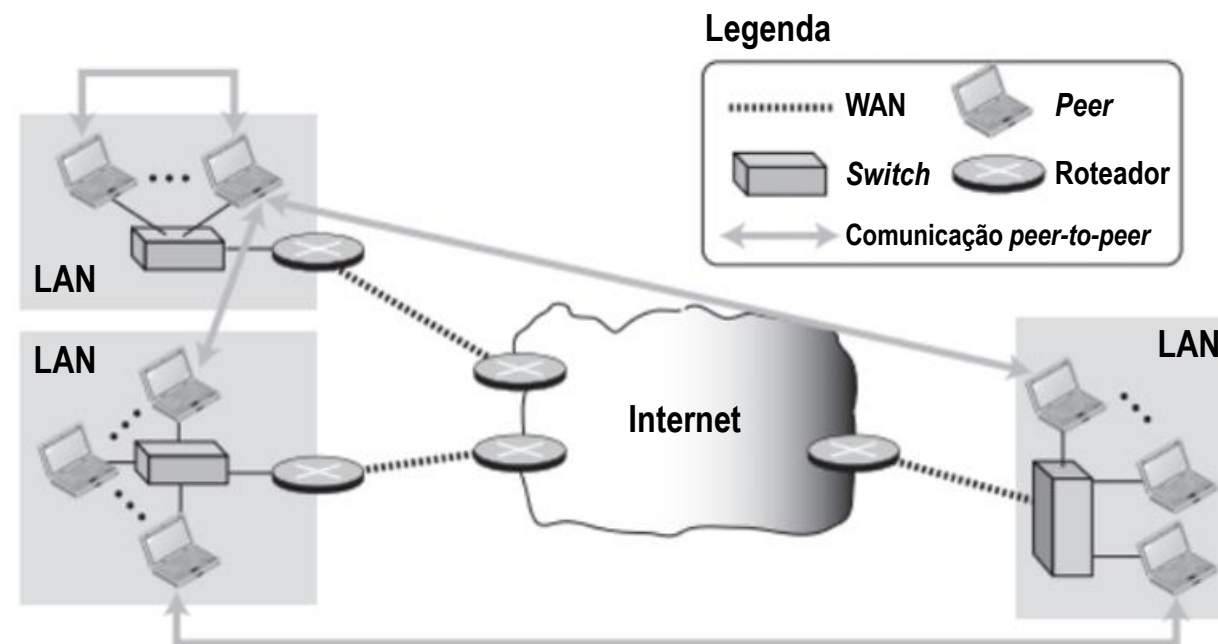
- Cliente-servidor possuem duas distribuições: vertical e horizontal.

## Distribuição horizontal

- Cliente ou servidor pode ser fisicamente subdividido em partes logicamente equivalentes.
- Pode possuir porção própria de dados.
- Balanceamento de carga.
- Papéis não muito bem definidos.
- Exemplo: compartilhamento de arquivos Peer-to-Peer, em que cada nó envia uma parte de um arquivo.

# Arquiteturas descentralizadas – Peer-to-Peer

- Par-a-par.
- Processos são todos iguais e flat.
- Cada processo age ao mesmo tempo como cliente e servidor (servente).
- O cliente é quem inicia a requisição.
- Rede de sobreposição: rede em que os nós são formados pelos processos e os enlaces denotam os canais de comunicação.
- Arquiteturas estruturadas e não estruturadas.



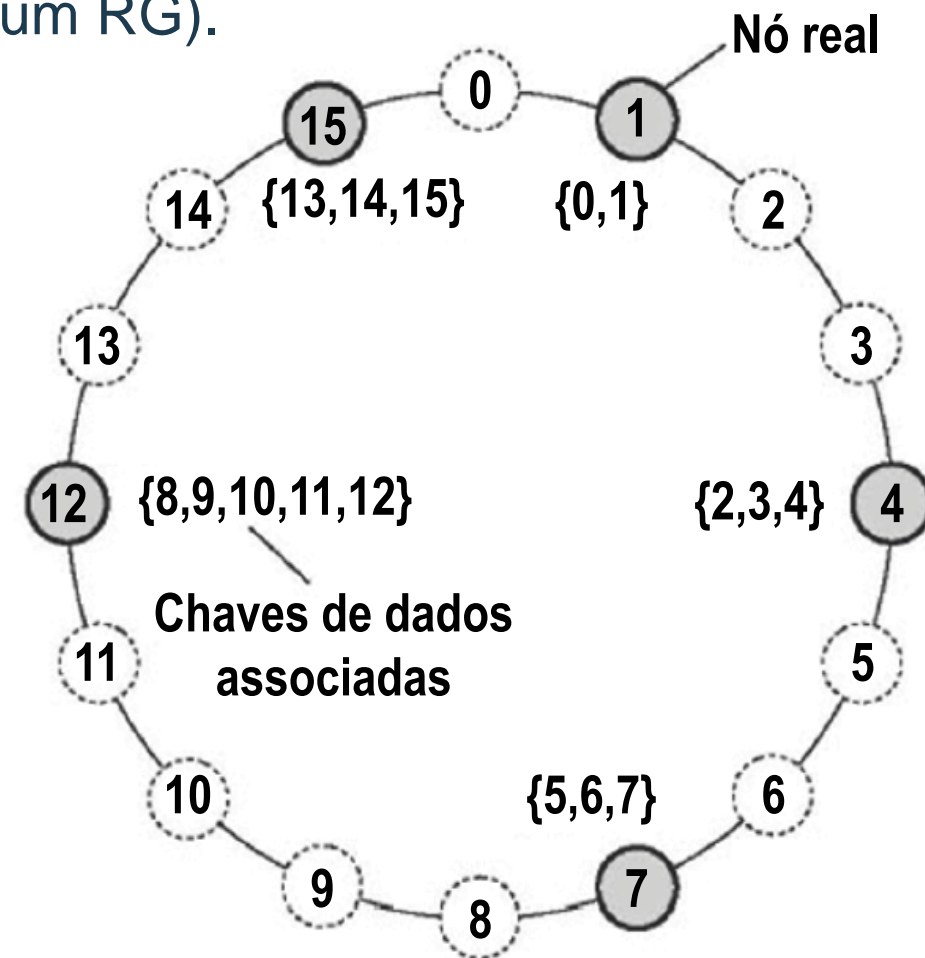


# Arquiteturas descentralizadas – Peer-to-Peer estruturada

- A rede de sobreposição é construída usando um mecanismo determinístico e “estruturado”.
- Mecanismo usado comumente: DHT (Distributed Hash Table).
- Dados e nós recebem uma chave aleatória (128~160 bits).
- Desafio: dada uma chave de um dado, mapear para o identificador de um nó.
- Neste caso, a requisição é roteada entre os nós até alcançar o nó com o dado solicitado.

# Arquiteturas descentralizadas – Peer-to-Peer estruturada

- **Chord:** implementação de um DHT para redes Peer-to-Peer.
- Nós estão logicamente organizados em um anel.
- Cada nó recebe um identificador aleatório (semelhante a um RG).



# Arquiteturas descentralizadas – Peer-to-Peer não estruturadas

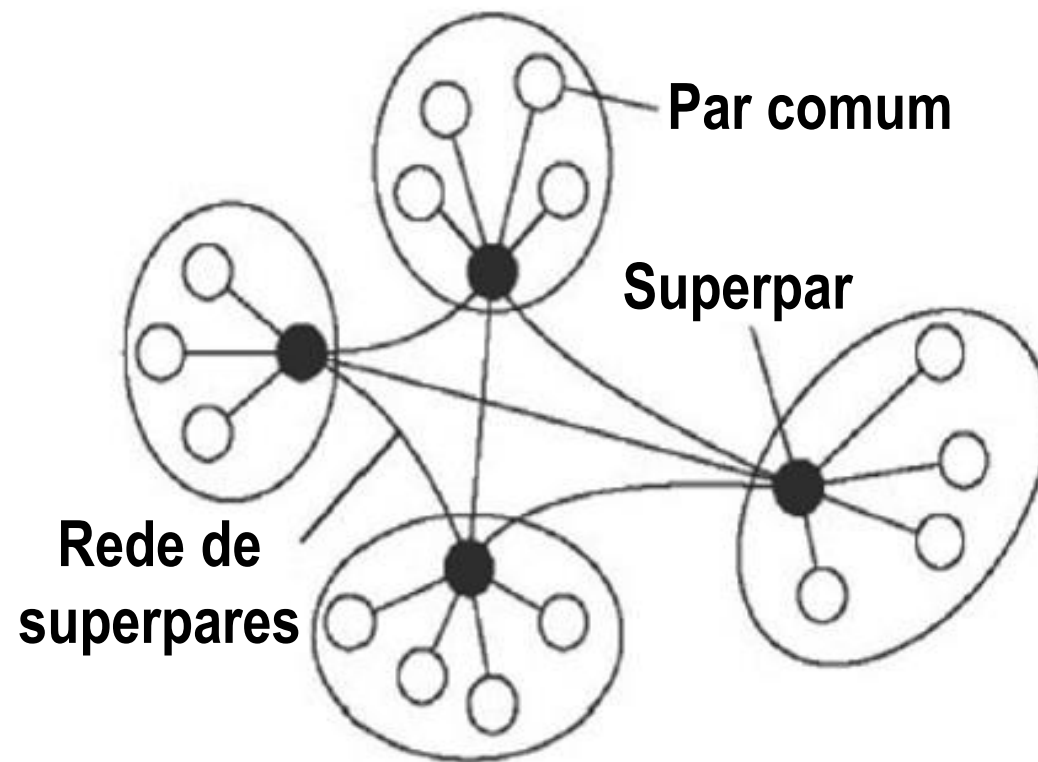
- Algoritmos aleatórios são utilizados para construir a rede de sobreposição.
- Cada nó mantém uma lista de nós vizinhos.
- Dados são também armazenados aleatoriamente.

Como é realizada a busca?

- Inunda toda a rede. Tempestade de broadcast.

# Arquiteturas descentralizadas – Peer-to-Peer superpares

- Com o crescimento da rede, localizar itens de dados em sistemas P2P não estruturados pode ser problemático.
- Nós “diretórios” e/ou intermediários.
- Sempre que um nó comum se junta à rede, se liga a um dos superpares.
- Dificuldade: Seleção do líder.
- Os superpares devem ter vida longa e alta disponibilidade.



# Arquiteturas híbridas

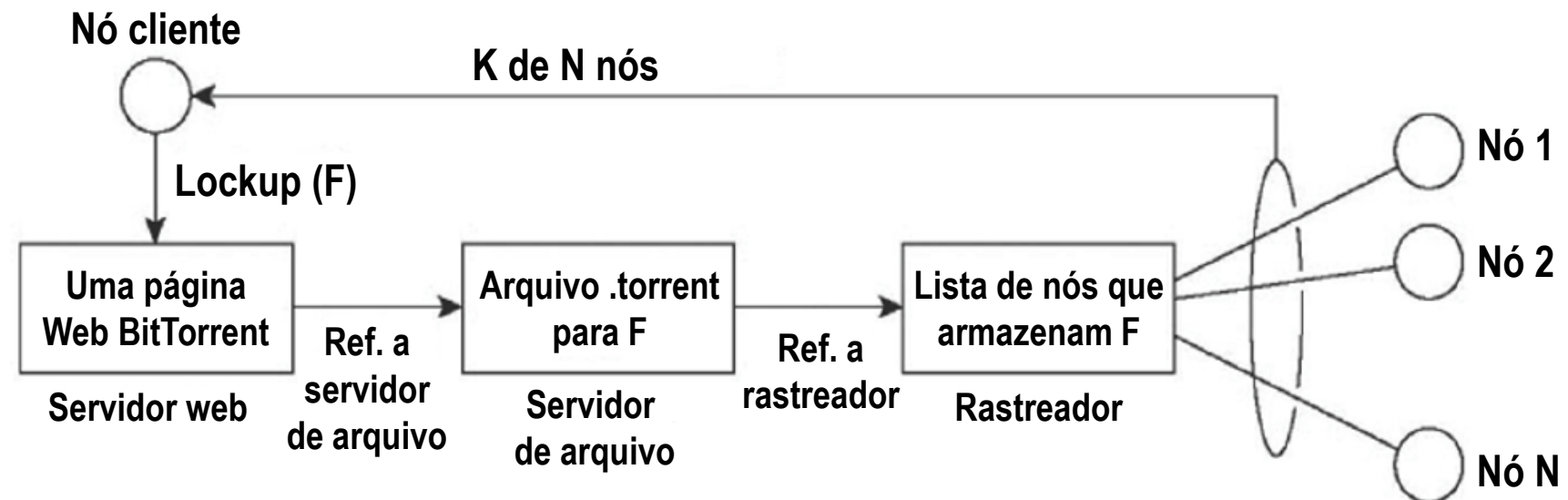
- Soluções de cliente-servidor são combinadas com a forma de funcionamento da arquitetura descentralizada.

Híbrida = centralizada + descentralizada

- Centralizada: serviço de diretório.
- Descentralizada: distribuição do conteúdo.
- Exemplos: Skype, BitTorrent, WhatsApp.

# Sistemas distribuídos colaborativos

- Sistemas distribuídos colaborativos.
- Inicialmente, um esquema de procura pelo cliente-servidor tradicional.
- Subsequentemente, o nó junta-se para dar um mecanismo descentralizado de colaboração.



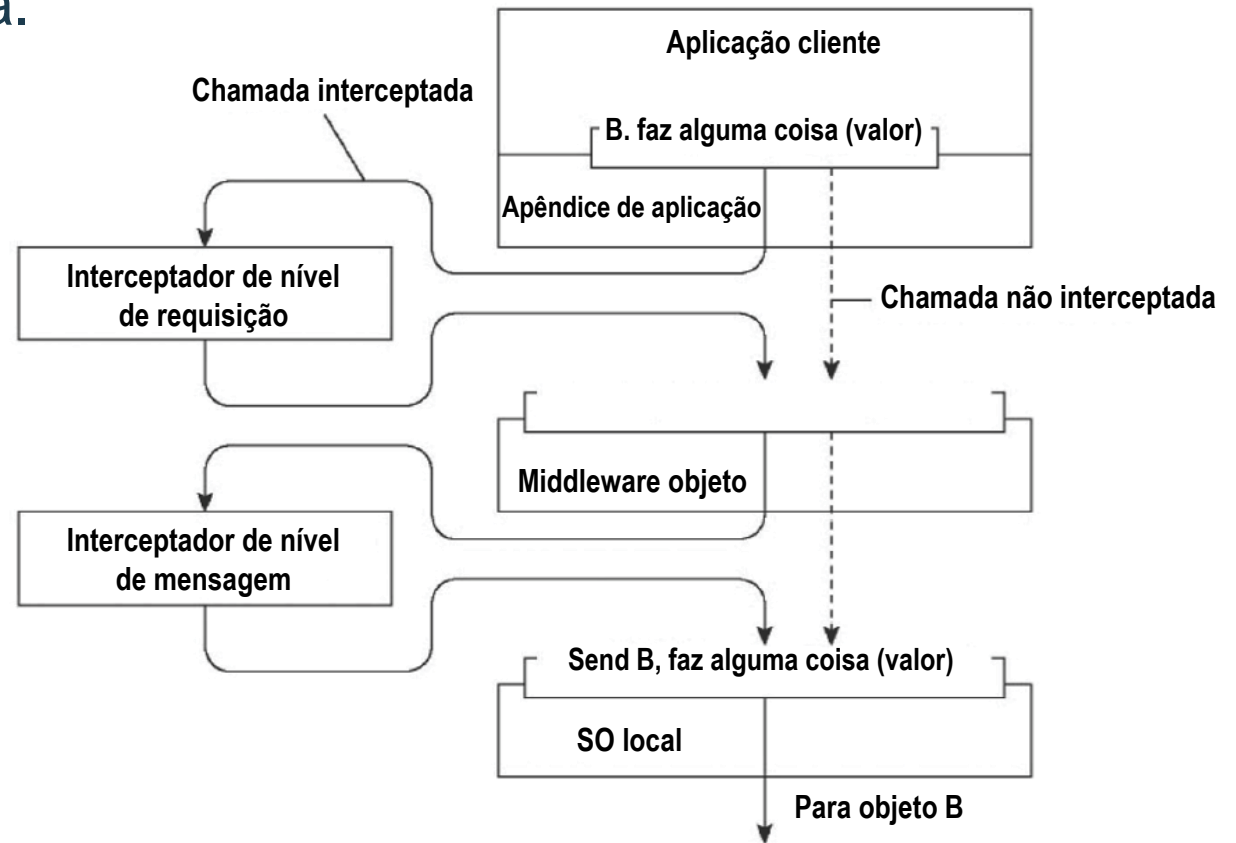
# Arquitetura vs. middleware

- Os middlewares como CORBA eram todos baseados em objetos
- É simples e específico para aplicações com objetos remotos, porém, não é trivial criar ou estender para outros tipos de aplicações.
- Necessidade de reconfiguração dinâmica em tempo de execução.

# Interceptadores

Uma solução para reconfigurar a necessidade da aplicação?

- **Interceptadores.**
- É uma construção de software que interromperá o fluxo de controle usual de execução.
- Em uma chamada desvia para outra chamada.
- Em tempo de execução.





# Interatividade

Em arquiteturas descentralizadas estruturadas, um desafio para essa estrutura é o mapeamento do identificador de um nó conhecendo a chave de um dado. Qual o mecanismo para identificar os nós que contêm os dados procurados?

- a) Chave primária.
- b) Superpares.
- c) Servidor de nomeação.
- d) Protocolo de comunicação TCP.
- e) Tabela de Hash Distribuída (DHT).

## Resposta

Em arquiteturas descentralizadas estruturadas, um desafio para essa estrutura é o mapeamento do identificador de um nó conhecendo a chave de um dado. Qual o mecanismo para identificar os nós que contêm os dados procurados?

- a) Chave primária.
- b) Superpares.
- c) Servidor de nomeação.
- d) Protocolo de comunicação TCP.
- e) Tabela de Hash Distribuída (DHT).

# Referências

- COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. *Sistemas distribuídos: conceitos e projeto*. 5. ed. Porto Alegre: Bookman, 2013.
- FOROUZAN, B. A.; MOSHARRAF, F. *Redes de computadores*. Porto Alegre: Grupo A, 2013.
- MONTEIRO, E. R.; JUNIOR, R. C. M.; LIMA, B. S. *et al. Sistemas distribuídos*. Porto Alegre: Grupo A, 2020.
- PEREIRA, C. S. *Sistemas distribuídos*. Londrina: Editora e Distribuidora Educacional S.A., 2019.
- TANENBAUM, Andrew S.; STEEN, Maarten Van. *Distributed systems*. 4. ed. Distributed-systems.net, 2023.
  - TANENBAUM, Andrew S.; STEEN, Maarten Van. *Sistemas distribuídos: princípios e paradigmas*. 2. ed. São Paulo: Prentice Hall Brasil, 2007.

**ATÉ A PRÓXIMA!**