ACH2147 - Desenvolvimento de Sistemas de Informação Distribuídos

Aula 06 – Midleware e Tipos de Arquitetura

Norton Trevisan Roman

7 de abril de 2022

Organização do *Middleware*

Problema

No caso de integração de sistemas legados, as interfaces oferecidas pelos diversos componentes provavelmente não servirão para todas as aplicações de interesse

Organização do *Middleware*

Problema

No caso de integração de sistemas legados, as interfaces oferecidas pelos diversos componentes provavelmente não servirão para todas as aplicações de interesse

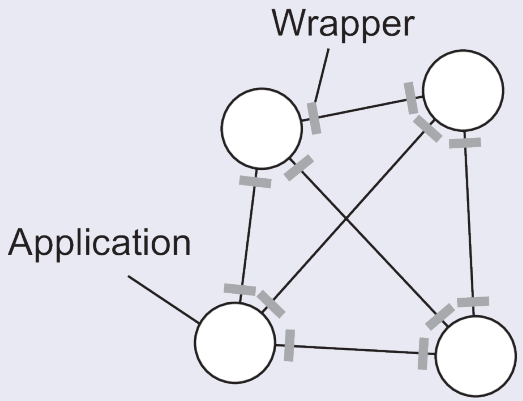
Solução

- Construir um adaptador, ou *wrapper*, que ofereça uma interface aceitável aos clientes
  - Dentro dele, as funções de sua interface são transformadas nas funções disponíveis nos componentes legados

Organização do *Middleware*

E como organizar os *wrappers*?

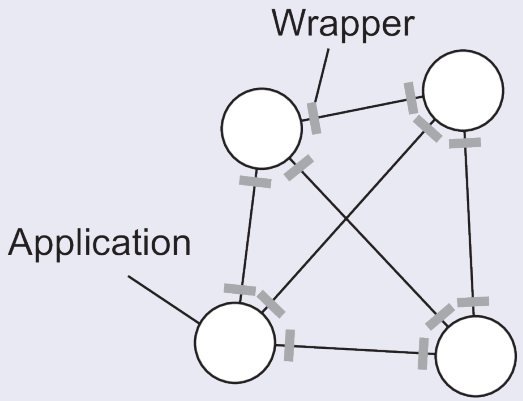
- Possibilidade 1:
  - Cada aplicação possui um *wrapper* para cada outra aplicação
  - Requer $n \times (n-1) = O(n^2)$ *wrappers*



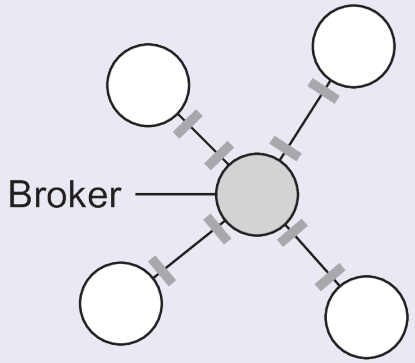
Organização do *Middleware*

E como organizar os *wrappers*?

- Possibilidade 1:
  - Cada aplicação possui um *wrapper* para cada outra aplicação
  - Requer $n \times (n-1) = O(n^2)$ *wrappers*



- Possibilidade 2:
  - Usar um **broker** – um componente centralizado que lida com todos os acessos entre as aplicações
  - Requer $2 \times n = O(n)$ *wrappers*



Organização do *Middleware*

Arquiteturas × Middleware

- Em muitos casos, arquiteturas/sistemas distribuídos são desenvolvidos de acordo com um estilo arquitetural específico
  - O estilo escolhido pode não ser o melhor em todos os casos
  - É necessário adaptar o comportamento do *middleware* dinamicamente

Organização do *Middleware*

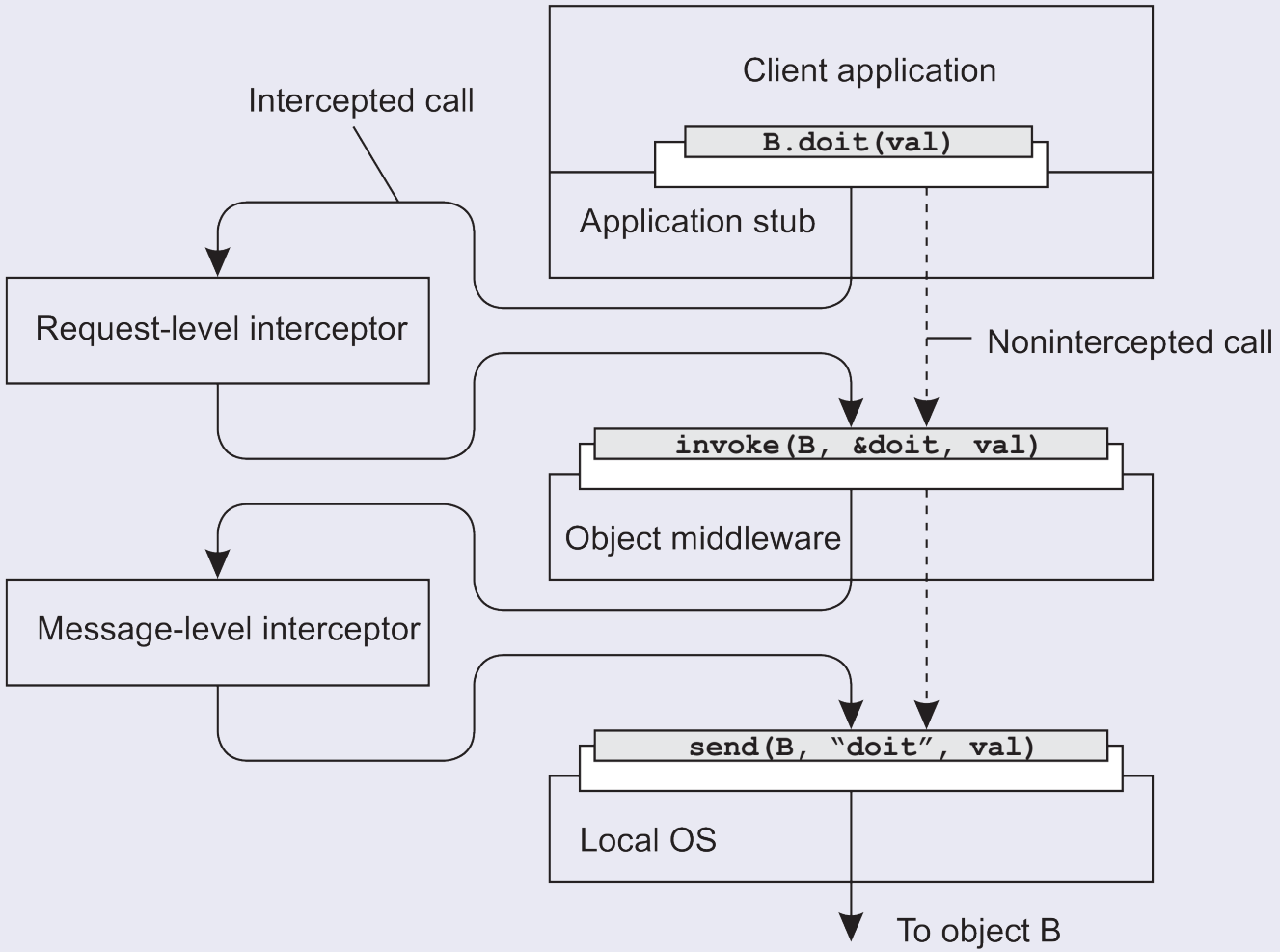
Arquiteturas × Middleware

- Em muitos casos, arquiteturas/sistemas distribuídos são desenvolvidos de acordo com um estilo arquitetural específico
  - O estilo escolhido pode não ser o melhor em todos os casos
  - É necessário adaptar o comportamento do *middleware* dinamicamente
- Isso pode ser feito via **interceptadores** (*interceptors*)

Organização do *Middleware*

Interceptadores

São construções de software que interceptam o fluxo de controle normal quando um objeto remoto é invocado



Tipos de Arquitetura

- Arquiteturas Centralizadas
- Arquiteturas Descentralizadas
- Arquiteturas Híbridas

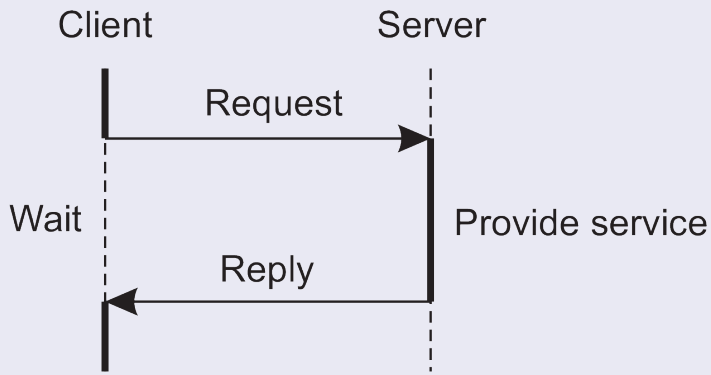
Tipos de Arquitetura

- **Arquiteturas Centralizadas**
- Arquiteturas Descentralizadas
- Arquiteturas Híbridas

Arquiteturas Centralizadas

Arquitetura básica Cliente-Servidor

- Características
  - Existem processos que oferecem serviços (servidores)
  - Existem processos que usam esses serviços (clientes)
  - Clientes e servidores podem estar em máquinas diferentes
  - Clientes seguem um modelo requisição/resposta ao usar os serviços



Arquiteturas Centralizadas

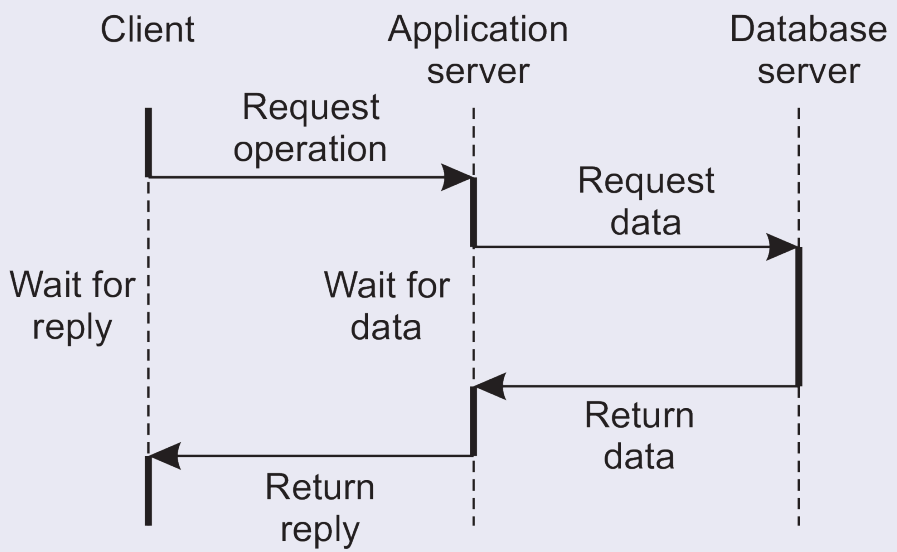
Arquiteturas Centralizadas Multicamadas

- Organizações tradicionais:
  - uma camada:
    configurações de terminal
    burro/mainframe
  - duas camadas:
    configuração
    cliente-servidor único.
  - três camadas: cada
    camada em uma máquina
    separada

Arquiteturas Centralizadas Multicamadas

- Organizações tradicionais:
  - uma camada: configurações de terminal burro/mainframe
  - duas camadas: configuração cliente-servidor único.
  - três camadas: cada camada em uma máquina separada

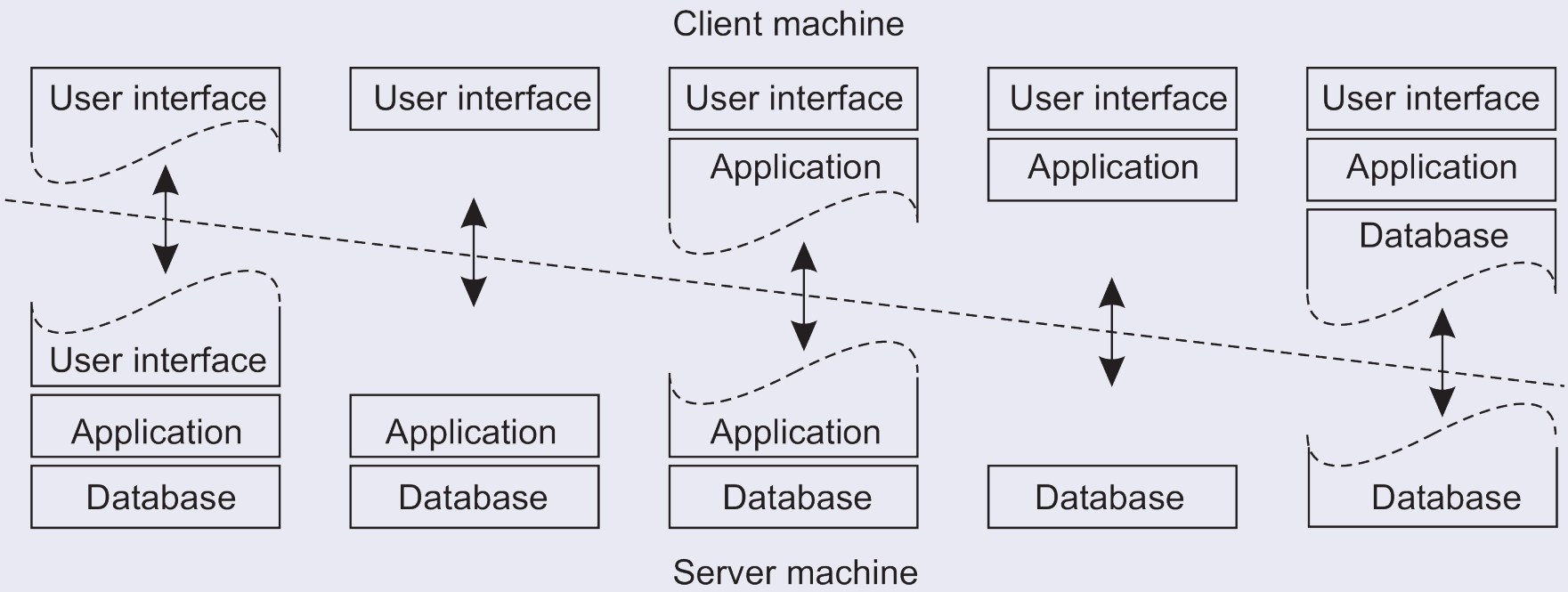
Rede de 3 camadas



Nela, o servidor por vezes também age como cliente

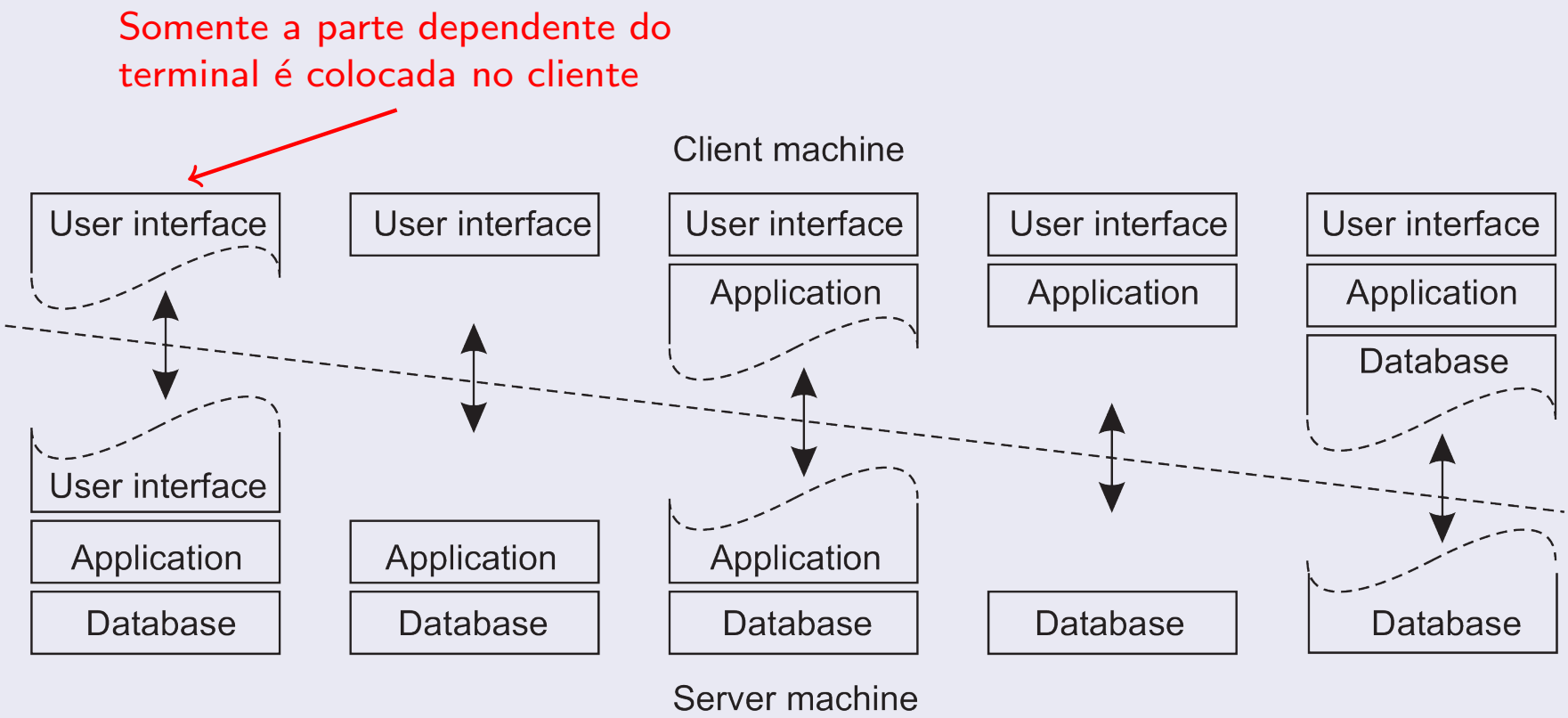
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



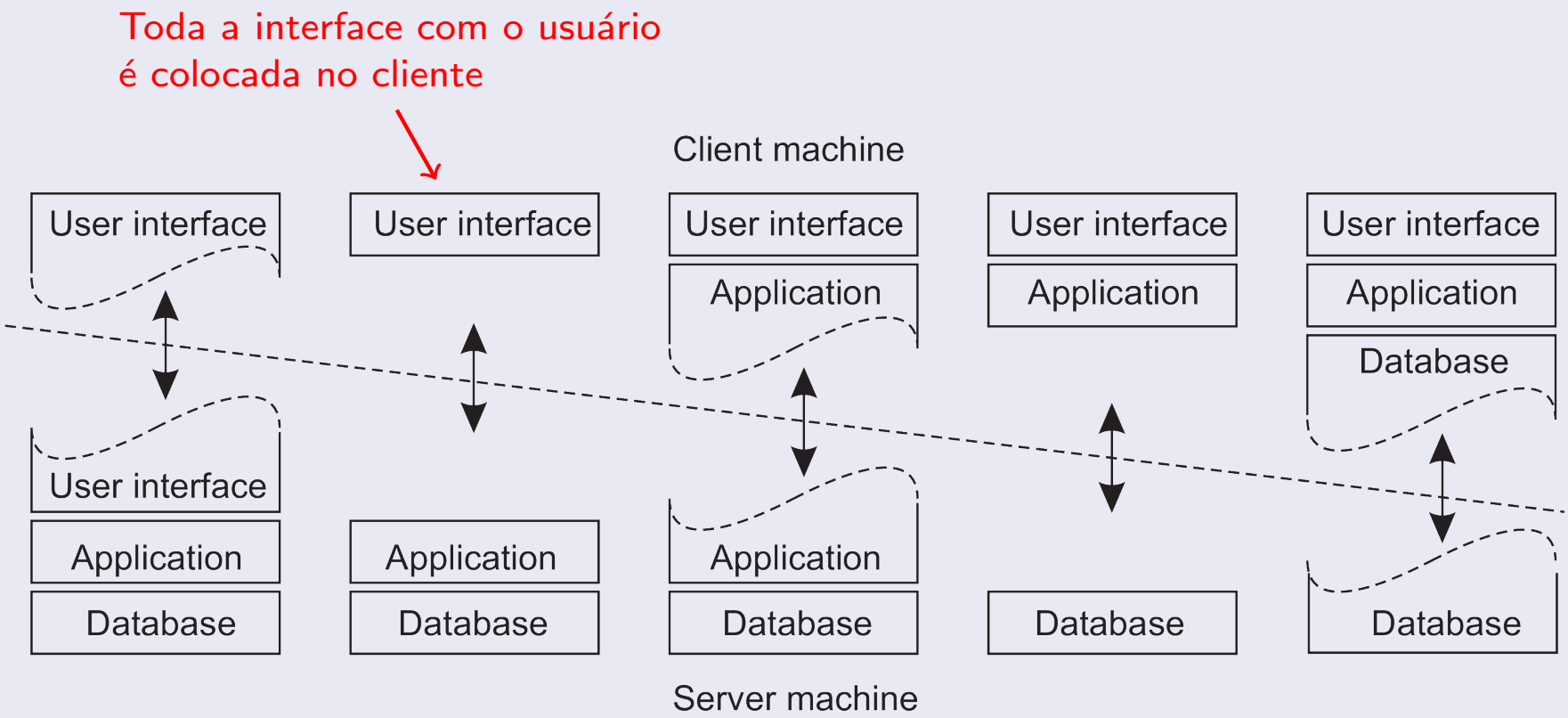
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



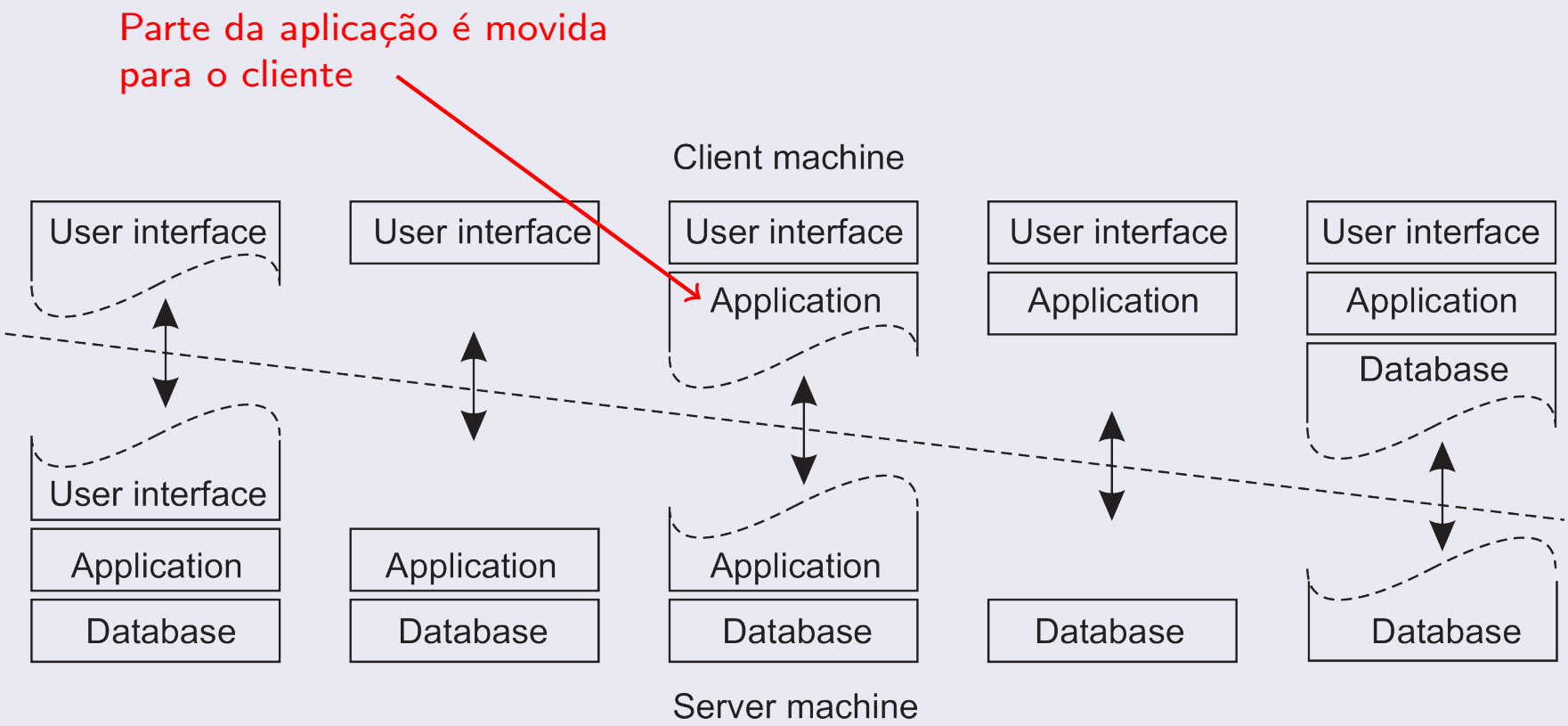
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



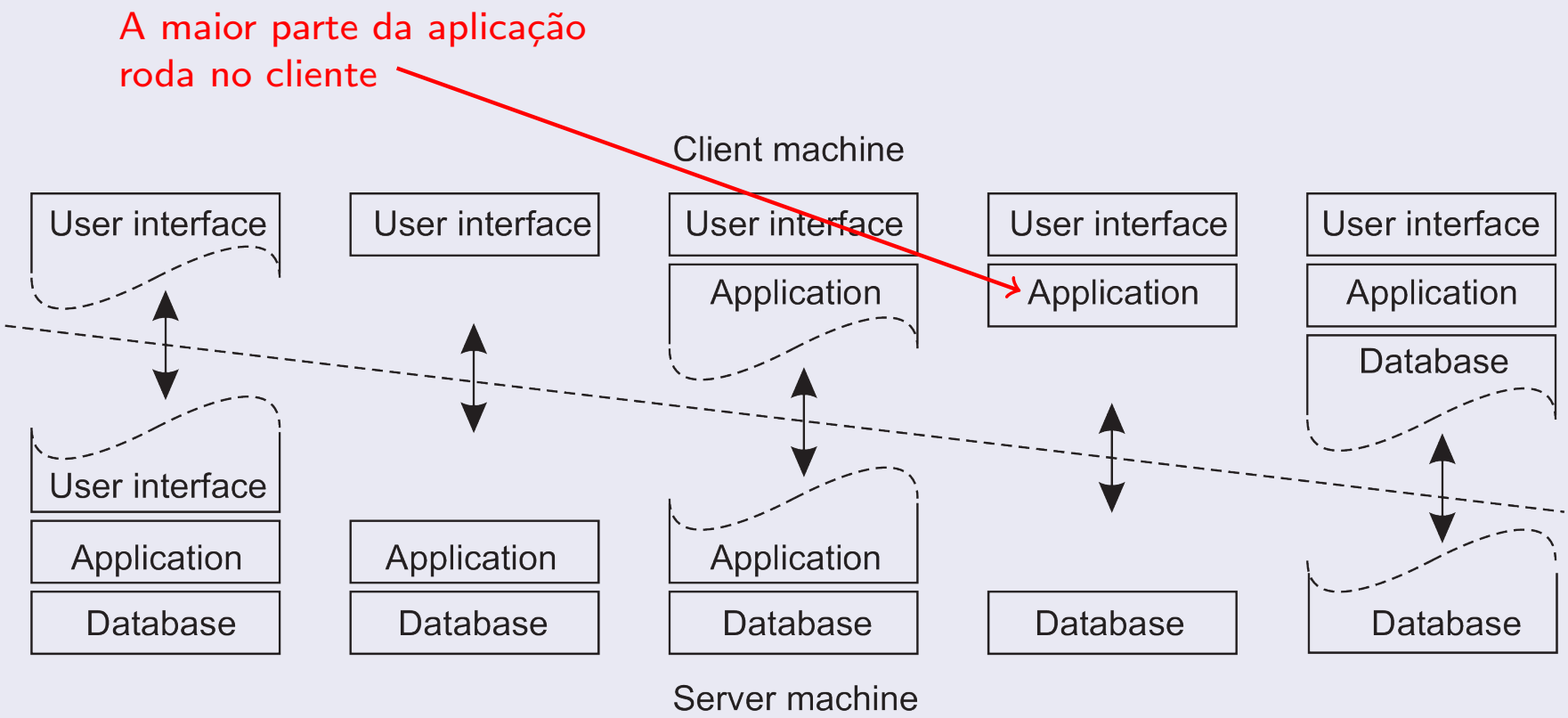
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



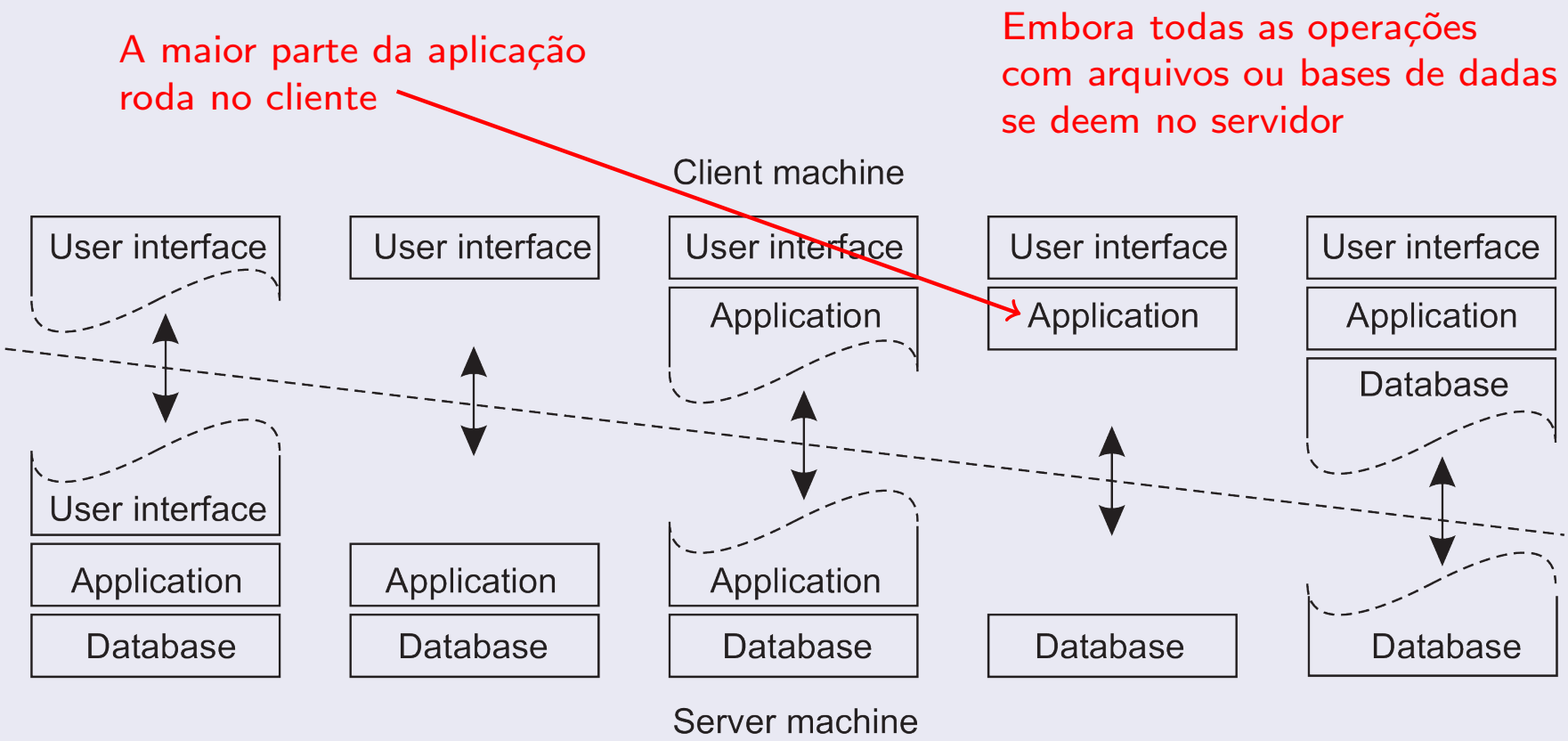
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



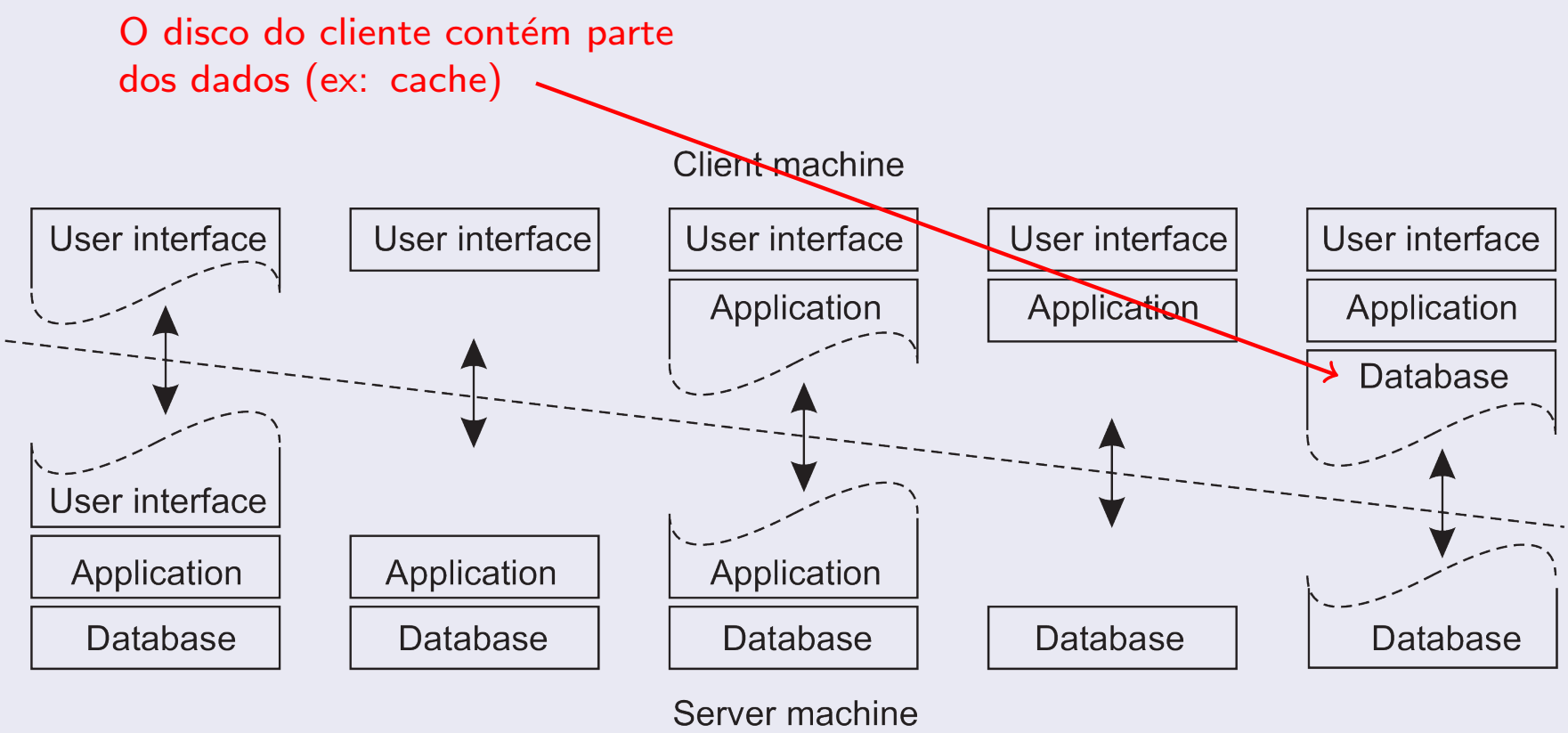
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



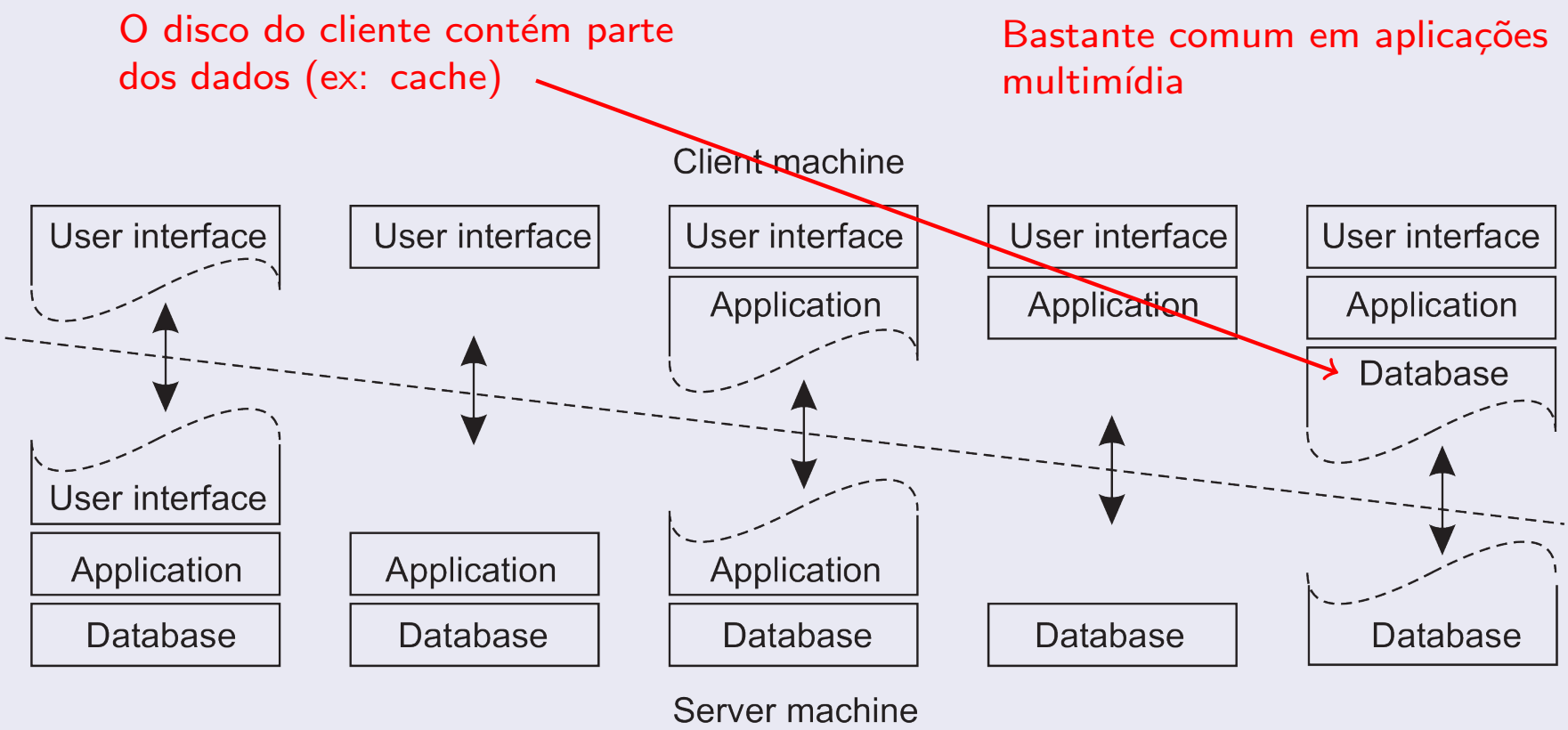
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



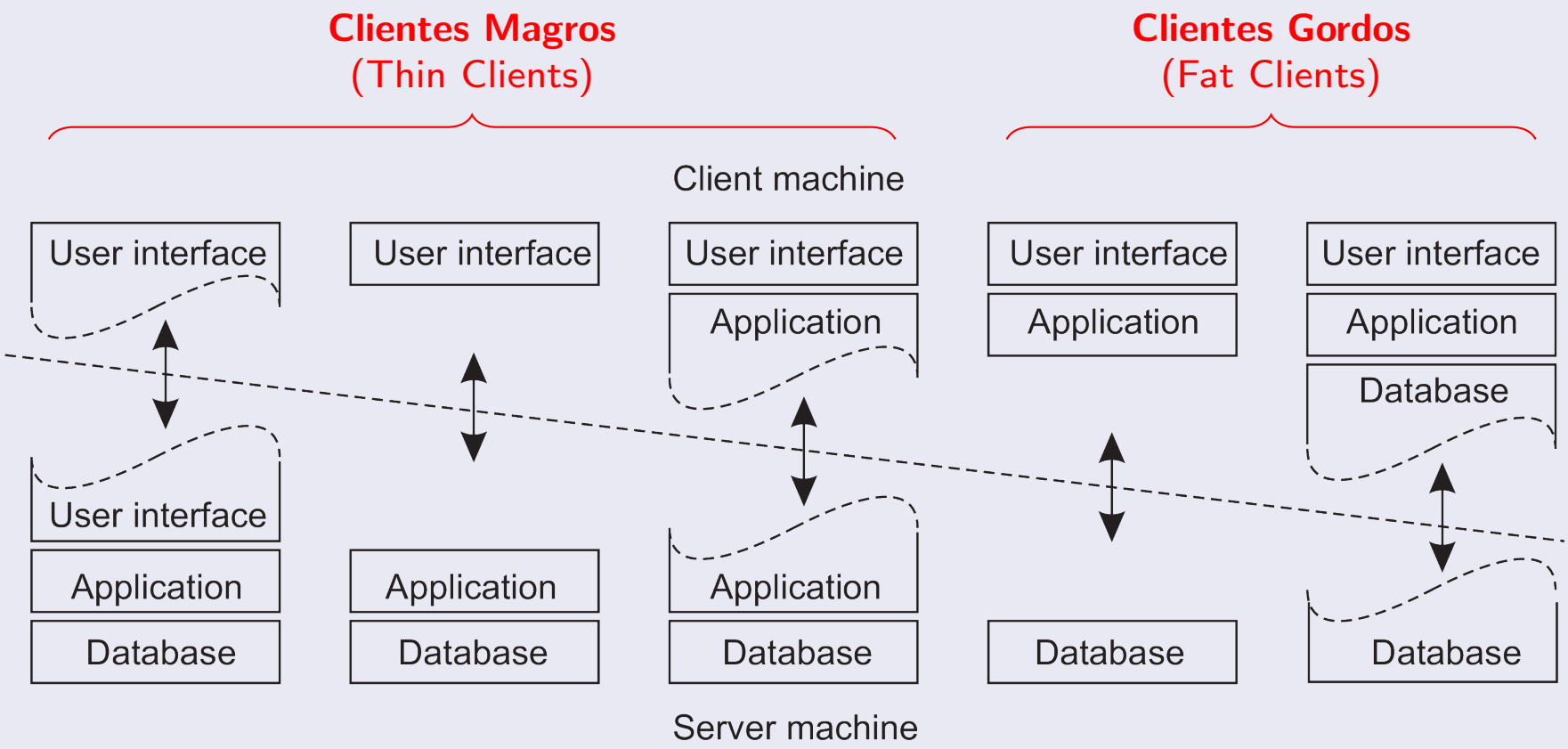
Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



Arquiteturas Centralizadas Multicamadas

Configurações tradicionais em 2 camadas



Arquitetura: Organização

Vertical × Horizontal

- Distribuição Vertical
  - Surge da divisão de aplicações distribuídas em 3 camadas lógicas
  - Os componentes de cada camada são então rodados em servidores distintos

Arquitetura: Organização

Vertical × Horizontal

- Distribuição Vertical
  - Surge da divisão de aplicações distribuídas em 3 camadas lógicas
  - Os componentes de cada camada são então rodados em servidores distintos
- Distribuição Horizontal
  - Cliente ou servidor podem ser separados em partes logicamente equivalentes
  - Cada parte trabalha em sua própria porção dos dados, balanceando a carga

Tipos de Arquitetura

- Arquiteturas Centralizadas
- **Arquiteturas Descentralizadas**
- Arquiteturas Híbridas

Arquiteturas Descentralizadas

Arquiteturas Peer-to-peer

- Arquitetura distribuída horizontalmente

Arquiteturas Descentralizadas

Arquiteturas Peer-to-peer

- Arquitetura distribuída horizontalmente
- Características
  - Todos os processos são iguais
  - As funções que precisam ser executadas são representadas por cada processo
  - Muito da interação entre processos é simétrica
    - Cada processo agirá como cliente e servidor ao mesmo tempo

Arquiteturas Descentralizadas

Arquiteturas Peer-to-peer

- Podem ser:
  - **Estruturados**: os nós são organizados seguindo uma estrutura de dados distribuída específica (anel, árvore, etc.)
  - **Não-estruturados**: os nós selecionam aleatoriamente seus vizinhos
  - **Híbridos**: alguns nós são designados, de forma organizada, a executar funções especiais

Arquiteturas Descentralizadas

Arquiteturas Peer-to-peer

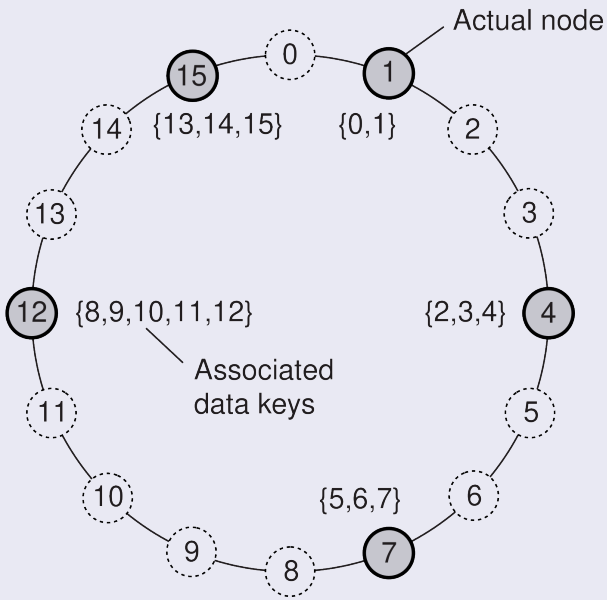
- Podem ser:
  - **Estruturados**: os nós são organizados seguindo uma estrutura de dados distribuída específica (anel, árvore, etc.)
  - **Não-estruturados**: os nós selecionam aleatoriamente seus vizinhos
  - **Híbridos**: alguns nós são designados, de forma organizada, a executar funções especiais

Praticamente todos os casos são exemplos de redes de overlay: rede na qual cada processo tem uma lista local de outros nós com os quais ele pode se comunicar

Arquiteturas Descentralizadas

Peer-to-peer Estruturado

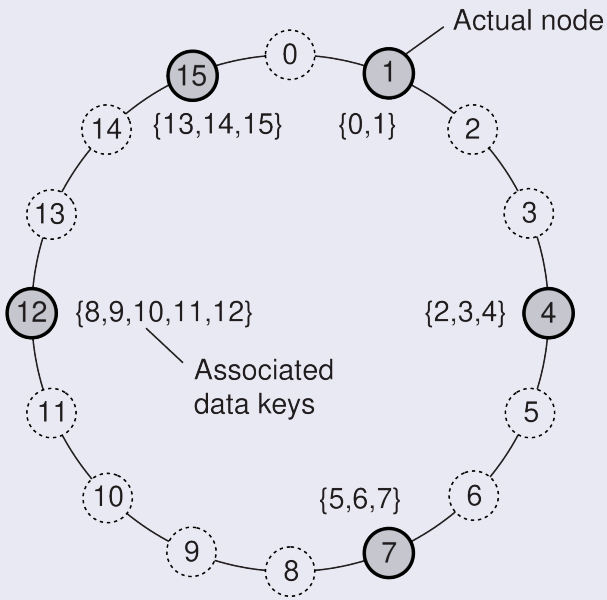
- Ideia básica
  - Organizar os nós em uma rede overlay estruturada (ex: um anel), e fazer com que alguns nós se tornem responsáveis por alguns serviços com base unicamente em seus IDs



Arquiteturas Descentralizadas

Peer-to-peer Estruturado

- Ideia básica
  - Organizar os nós em uma rede overlay estruturada (ex: um anel), e fazer com que alguns nós se tornem responsáveis por alguns serviços com base unicamente em seus IDs

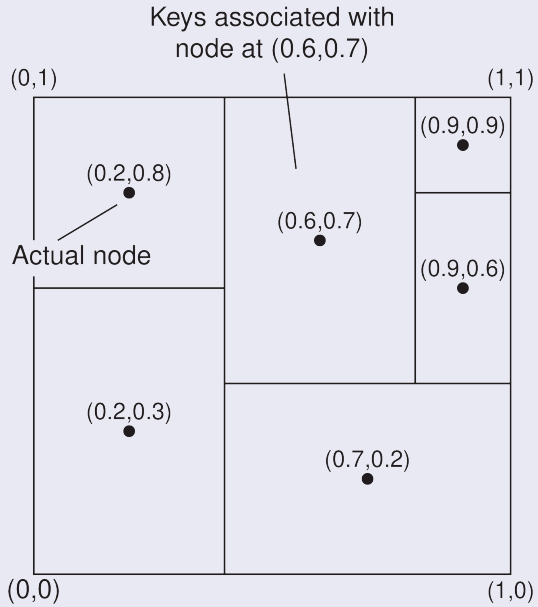


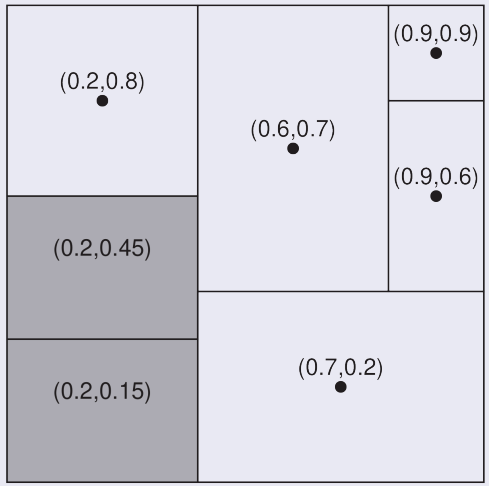
- Cada item é unicamente associado a uma chave
  - O sistema provê uma operação `LOOKUP(key)` que irá fazer o roteamento de uma requisição até o nó correspondente

Arquiteturas Descentralizadas

Peer-to-peer Estruturado

- Outro exemplo
  - Organize os nós em um espaço d-dimensional e faça todos os nós ficarem responsáveis por um dado em uma região específica. Quando um nó for adicionado, divida a região.





Arquiteturas Descentralizadas

Peer-to-peer Não-Estruturado

- Cada nó mantém uma lista de seus vizinhos
  - Cada participante mantém uma visão parcial da rede, consistindo de c outros nós
  - Cada nó P seleciona periodicamente um nó Q de sua visão parcial
  - P e Q trocam informação **&&** trocam membros de suas respectivas visões parciais

Arquiteturas Descentralizadas

Peer-to-peer Não-Estruturado

- O resultado lembra um grafo aleatório
  - Em que uma aresta $< u, v >$ existe apenas com uma certa probabilidade $P(< u, v >)$

Arquiteturas Descentralizadas

Peer-to-peer Não-Estruturado

- O resultado lembra um grafo aleatório
  - Em que uma aresta $< u, v >$ existe apenas com uma certa probabilidade $P(< u, v >)$
- Quando um nó se junta à rede, ele contata um nó bem conhecido para obter uma lista inicial de nós no sistema
  - Essa lista pode então ser usada para encontrar mais nós no sistema

Arquiteturas Descentralizadas

Peer-to-peer Não-Estruturado

- Problema
  - A busca de um dado não pode mais seguir uma rota predeterminada

Arquiteturas Descentralizadas

Peer-to-peer Não-Estruturado

- Problema
  - A busca de um dado não pode mais seguir uma rota predeterminada
- Precisamos então buscar pelo dado
  - Inundação (*flooding*)
  - Caminhada aleatória (*random walk*)

Arquiteturas Descentralizadas

P2P Não-Estruturado: Busca

- **Inundação**:
  - O nó inicial u faz uma requisição a todos seus vizinhos
  - Se o nó receptor já recebeu antes essa requisição, ele a ignora
  - Do contrário, ele verifica se possui o dado localmente
  - Se ele não possuir o dado, repassa a requisição a seus vizinhos
  - Do contrário, ele envia o dado a quem enviou a requisição, que o repassará recursivamente a quem enviou a ele a requisição

Arquiteturas Descentralizadas

P2P Não-Estruturado: Busca

- **Caminhada Aleatória**:
  - O nó inicial u passa a requisição a um vizinho escolhido aleatoriamente v
  - Se v não tiver o dado, ele repassa a requisição a um de seus vizinhos, também escolhido aleatoriamente, e assim por diante

Arquiteturas Descentralizadas

P2P Não-Estruturado: Busca

- **Caminhada Aleatória**:
  - O nó inicial u passa a requisição a um vizinho escolhido aleatoriamente v
  - Se v não tiver o dado, ele repassa a requisição a um de seus vizinhos, também escolhido aleatoriamente, e assim por diante
- Caminhadas aleatórias impõem muito menos tráfego à rede
  - Mas podem levar muito mais tempo para encontrar um nó que possua o dado buscado

Arquiteturas Descentralizadas

P2P Não-Estruturado: Busca

- Em ambos os casos uma requisição precisa ter um limite
  - Criar um valor TTL (*time-to-live*) $\rightarrow$ um número máximo de vezes que ela pode ser repassada
  - Alternativamente, quando um nó receber uma requisição, ele pode verificar com o nó inicial se o pedido já não foi atendido

Arquiteturas Descentralizadas

Redes Superpeer

- Uma alternativa à busca em P2Ps não estruturados é o uso de **Servidores de Índices**
  - Nós especiais, que mantêm um índice dos dados na rede

Arquiteturas Descentralizadas

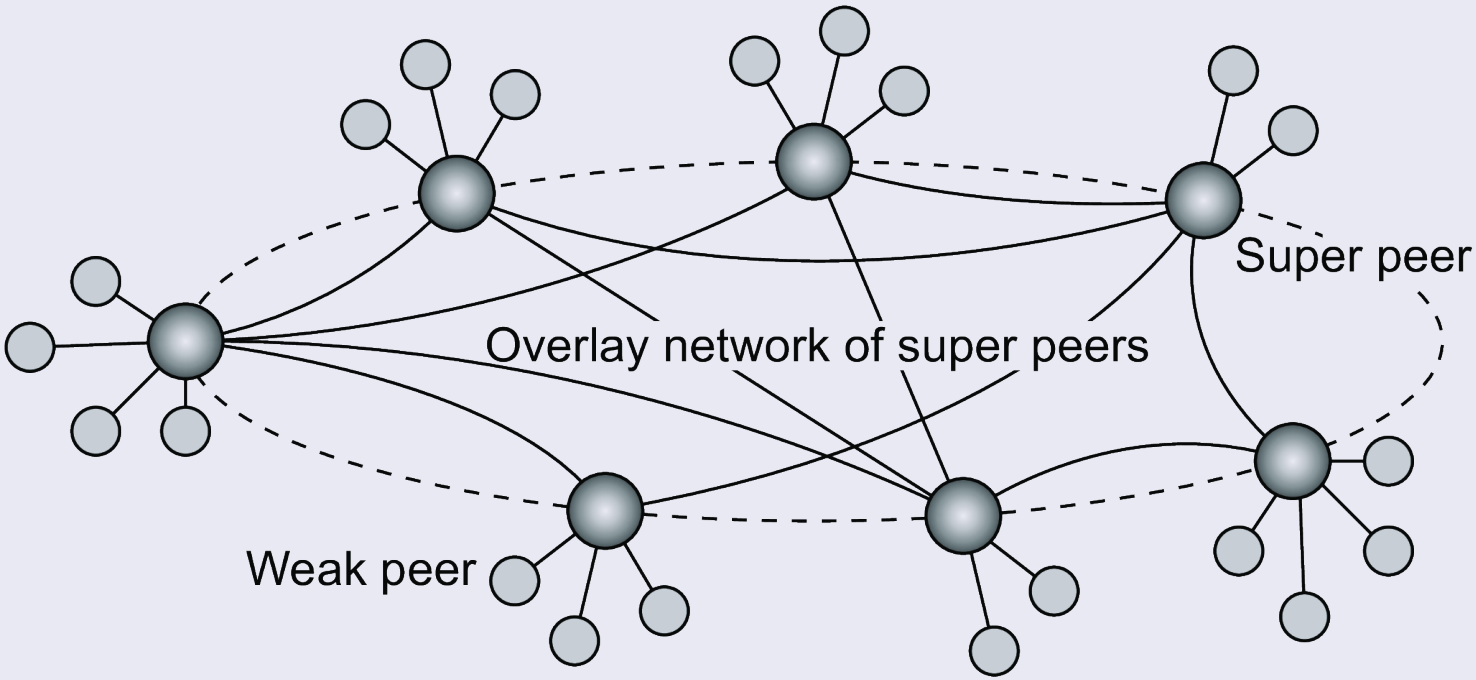
Redes Superpeer

- Uma alternativa à busca em P2Ps não estruturados é o uso de **Servidores de Índices**
  - Nós especiais, que mantêm um índice dos dados na rede
- Outra seria o uso de **Brokers**
  - Nós que coletam dados sobre o uso e disponibilidade de recursos para um determinado grupo de nós próximos uns dos outros
  - Podem assim rapidamente selecionar nós com recursos suficientes

Arquiteturas Descentralizadas

Redes Superpeer

- Nós agindo como brokers ou servidores de índices são conhecidos como **superpeers** (Ex: Skype)



Arquiteturas Descentralizadas

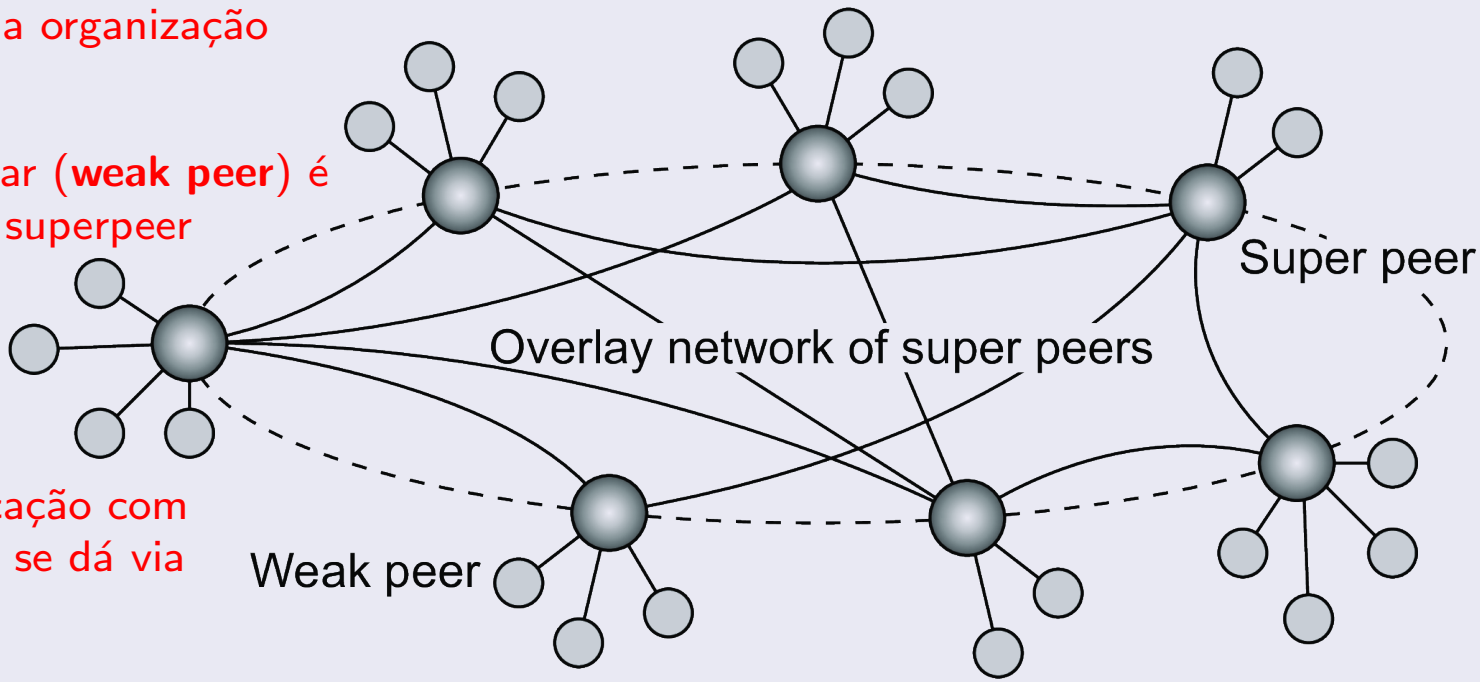
Redes Superpeer

- Nós agindo como brokers ou servidores de índices são conhecidos como **superpeers** (Ex: Skype)

Uma rede P2P com super-peers gera uma organização hierárquica

Cada nó regular (**weak peer**) é cliente de um superpeer

Toda comunicação com um weak peer se dá via seu superpeer



Tipos de Arquitetura

- Arquiteturas Centralizadas
- Arquiteturas Descentralizadas
- **Arquiteturas Híbridas**

Arquiteturas Híbridas

Dificuldade

- O principal problema com soluções descentralizadas é como iniciar a conexão
  - Frequentemente, feita via um esquema cliente-servidor tradicional
  - Uma vez que o nó se juntou ao sistema, ele pode então usar um esquema totalmente descentralizado para colaboração

Arquiteturas Híbridas

Dificuldade

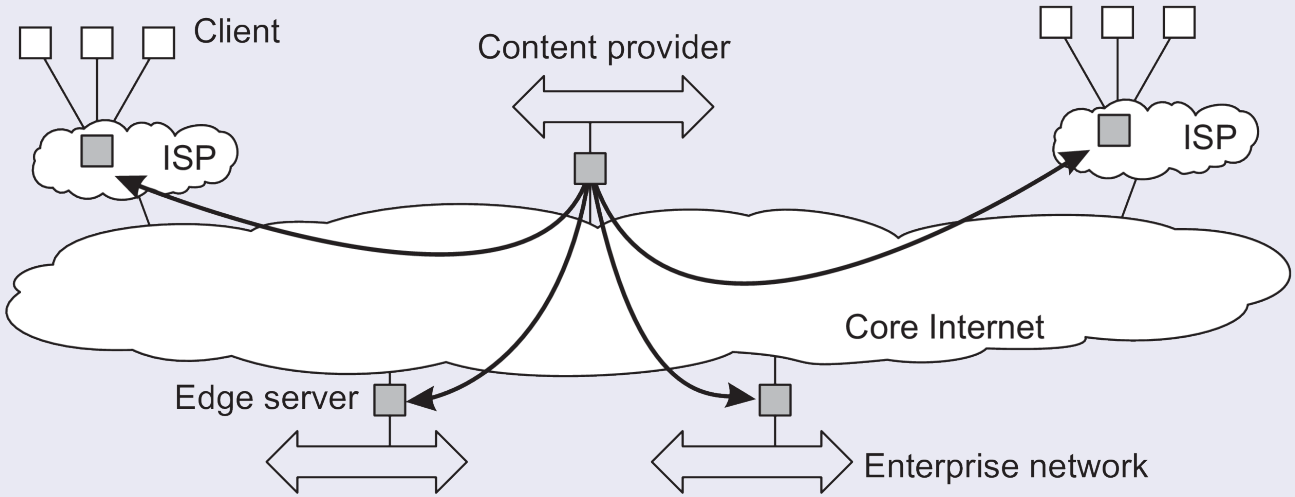
- O principal problema com soluções descentralizadas é como iniciar a conexão
  - Frequentemente, feita via um esquema cliente-servidor tradicional
  - Uma vez que o nó se juntou ao sistema, ele pode então usar um esquema totalmente descentralizado para colaboração

- Nas arquiteturas híbridas, soluções cliente-servidor são combinadas com organizações descentralizadas
  - Em geral, um componente centralizado é usado para lidar com as requisições iniciais

Arquiteturas Híbridas

Arquitetura de Servidor de Borda (*Edge-server*)

- Sistemas em que os servidores são colocados "na borda" da rede
  - Na divisa entre as redes corporativas e a internet
  - Ex: Provedores de internet (*Internet Service Providers – ISPs*), através dos quais usuários conectam-se à rede



Arquiteturas Híbridas

Sistemas Ditribuídos Colaborativos

- Trata-se de sistemas que em geral forçam a colaboração entre os nós
  - A obtenção de um serviço exige o fornecimento de outro

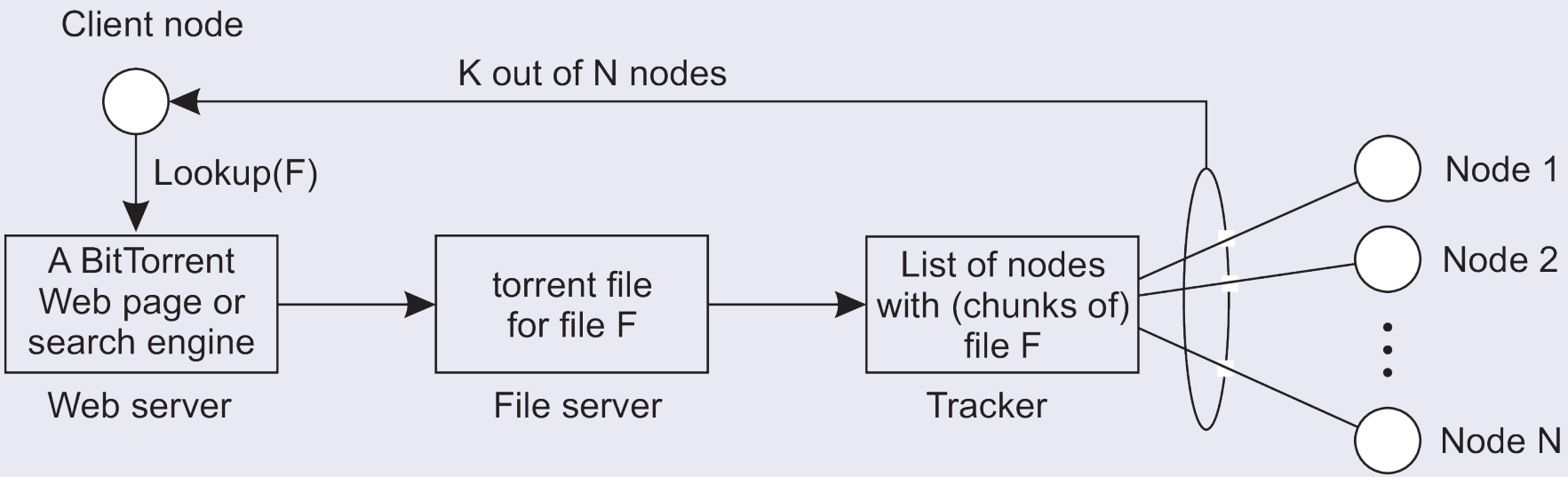
Arquiteturas Híbridas

Sistemas Ditribuídos Colaborativos

- Trata-se de sistemas que em geral forçam a colaboração entre os nós
  - A obtenção de um serviço exige o fornecimento de outro
- Ex: BitTorrent
  - Assim que um nó identifica de onde o arquivo será baixado, ele se junta a uma swarm (multidão) de pessoas que, em paralelo, receberão pedaços do arquivo da fonte e redistribuirão esses pedaços entre os outros

Arquiteturas Híbridas

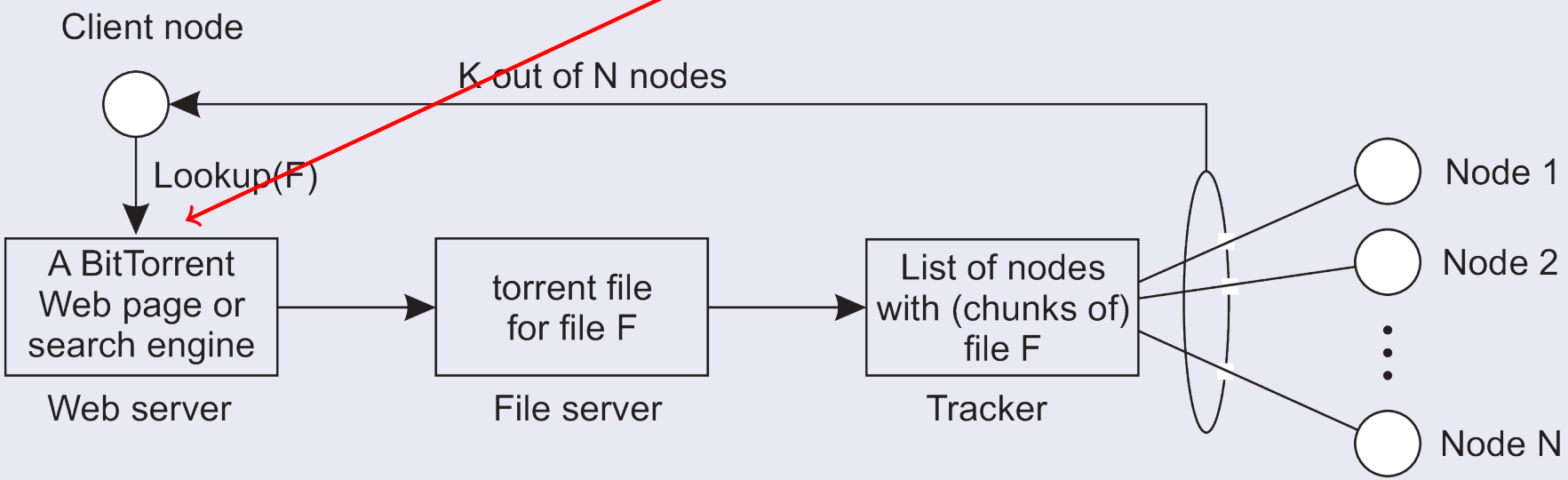
S.D. Colaborativos: BitTorrent



Arquiteturas Híbridas

S.D. Colaborativos: BitTorrent

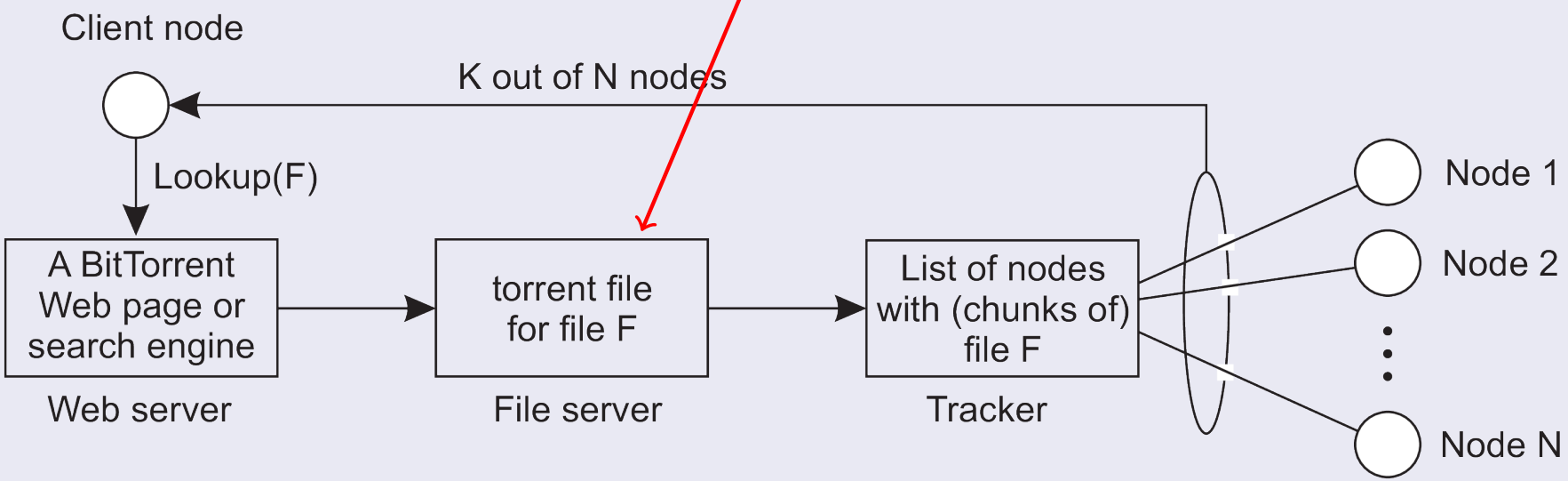
Para baixar um arquivo, o usuário acessa um diretório global (geralmente um *website* conhecido)



Arquiteturas Híbridas

S.D. Colaborativos: BitTorrent

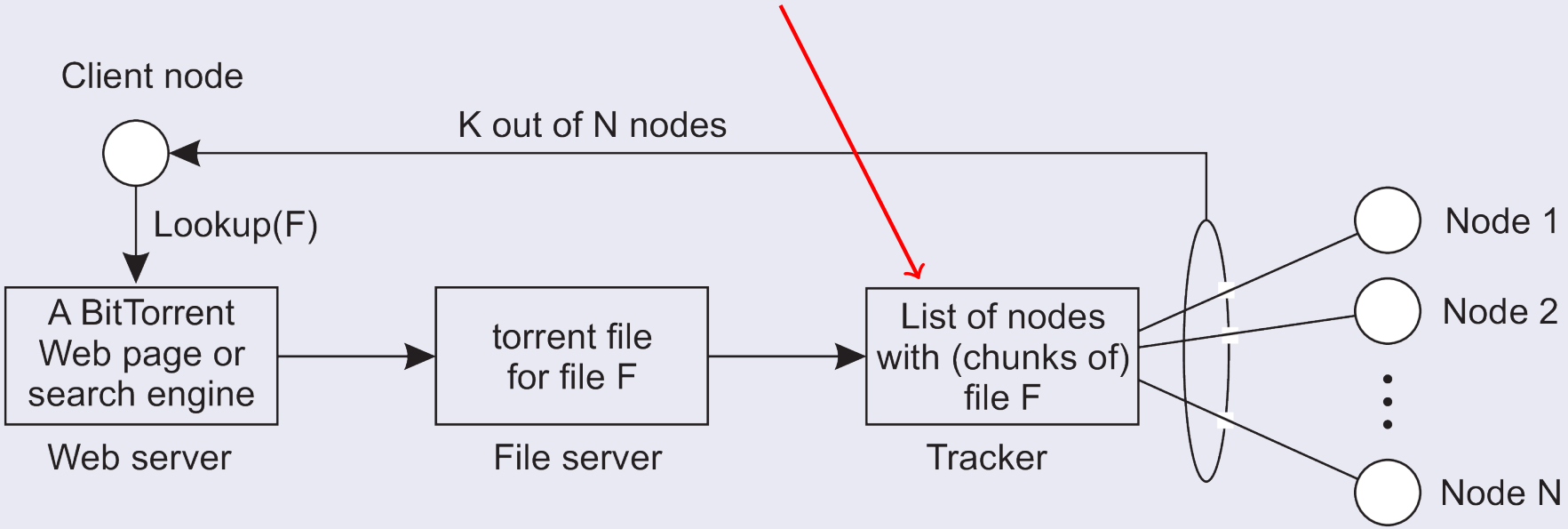
Esse diretório possui referências para arquivos torrent (que contêm a informação necessária para baixar um arquivo específico)



Arquiteturas Híbridas

S.D. Colaborativos: BitTorrent

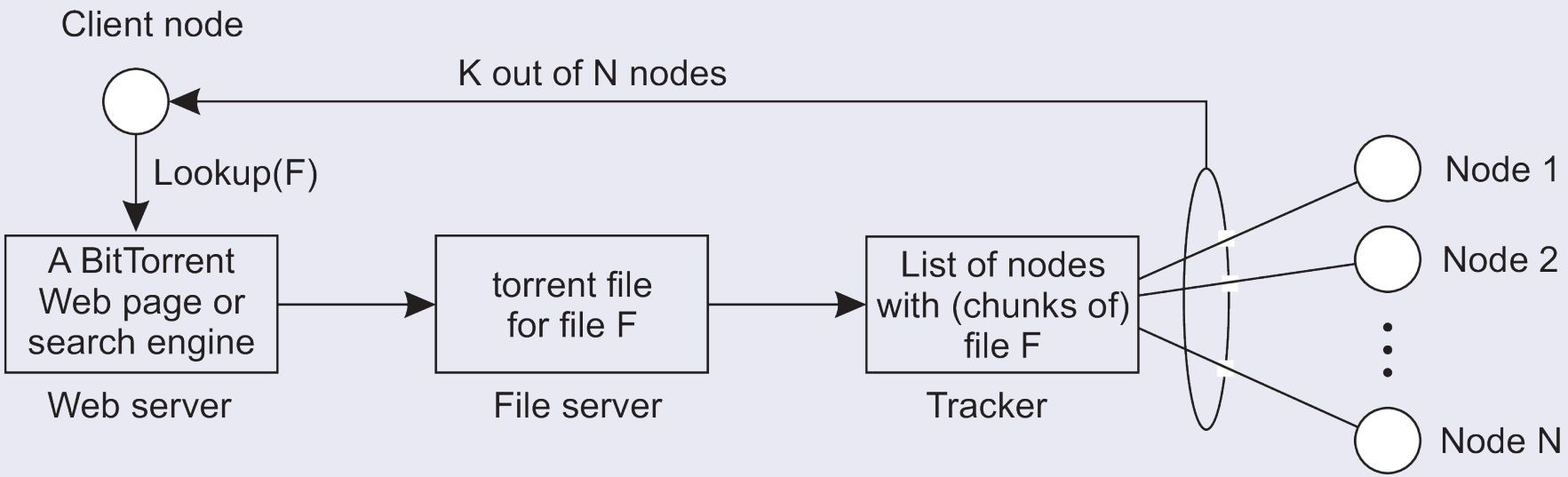
Em especial, o arquivo torrent contém um link para um **tracker** (servidor que registra os nós **ativos** que possuem partes do arquivo desejado)



Arquiteturas Híbridas

S.D. Colaborativos: BitTorrent

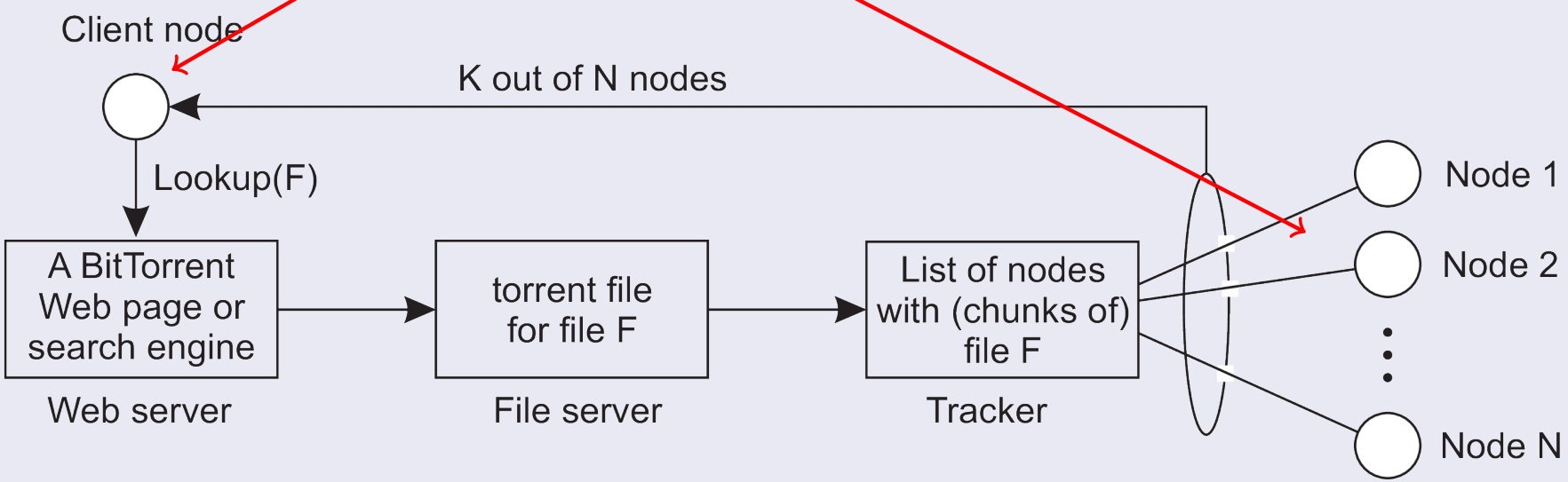
Um nó ativo é um nó que está atualmente baixando esse mesmo arquivo de interesse



Arquiteturas Híbridas

S.D. Colaborativos: BitTorrent

Uma vez identificados os nós de onde se pode baixar o arquivo, o nó cliente se torna ativo, se juntando à multidão



Arquiteturas Híbridas

S.D. Colaborativos: BitTorrent

Nesse ponto, ele é forçado a fornecer partes do arquivo para *download* também → é forçado a colaborar