Sistemas Distribuídos - Capítulo 2 - Aula 2

Aula passada

- Introdução, metas e tipos de Sistemas Distribuídos, Exemplos

Aula de hoje

- Estilos Arquitetônicos
- Arquitetura de Sistemas
- Arquiteturas e Middleware

Por quê definir uma arquitetura?

- SDs são complexas peças de *software*
- Componentes estão espalhados por diversas máquinas
- Sistemas devem ser organizados adequadamente!
  - Organização lógica do conjunto de componentes
  - Como organizar os componentes fisicamente?

Estilos Arquitetônicos (1/3)

Um componente é uma unidade modular com interfaces requeridas e fornecidas bem definidas que é substituível dentro do seu ambiente

Estilo Arquitetônico (2/3)

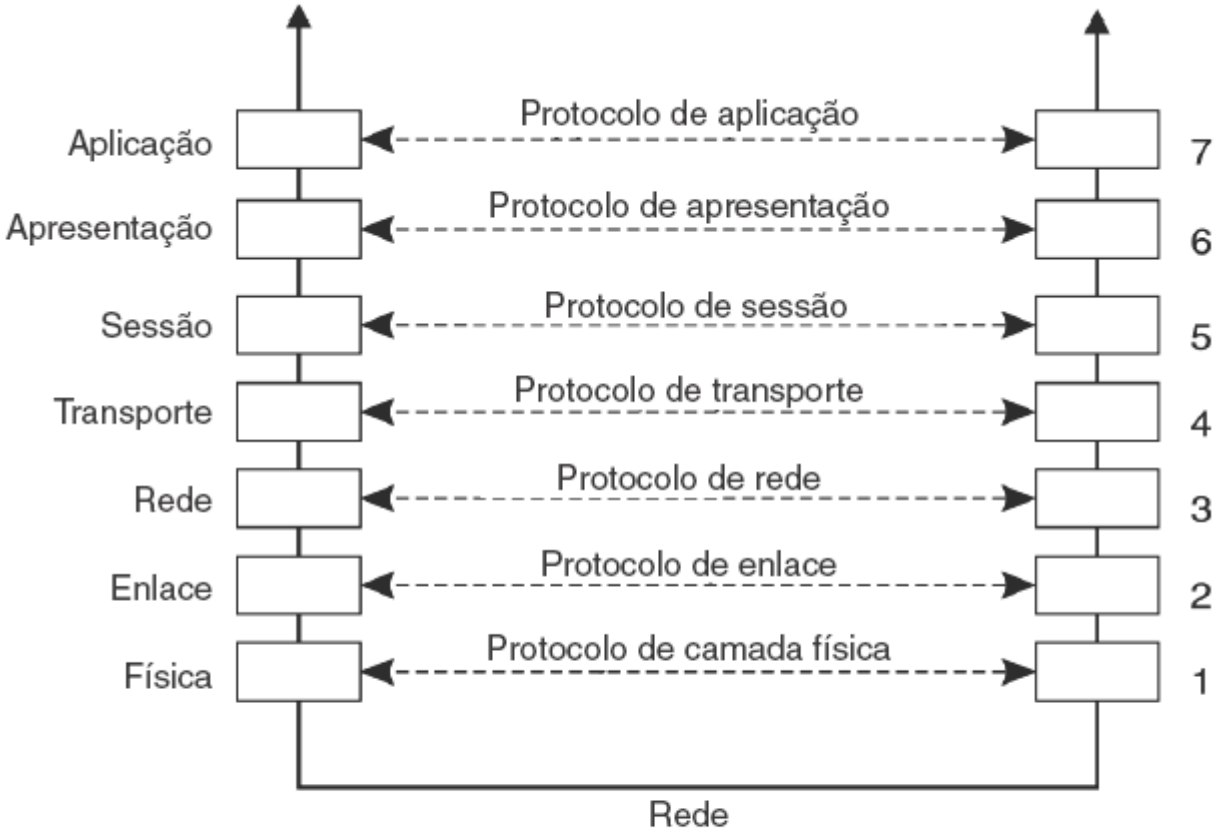
- É formulado em termos de componentes
- Modo como os componentes estão conectados
- Dados trocados entre componentes
- Maneira como os componentes são configurados em conjunto para formar um sistema

Estilo Arquitetônico (3/3)

- Arquiteturas em Camadas
- Arquiteturas baseadas em objetos
- Arquiteturas centradas em dados
- Arquiteturas baseadas em eventos

Arquiteturas em Camadas

- Idéia Básica: um componente na camada L_i tem permissão de chamar componentes na camada subjacente L_{i-1}

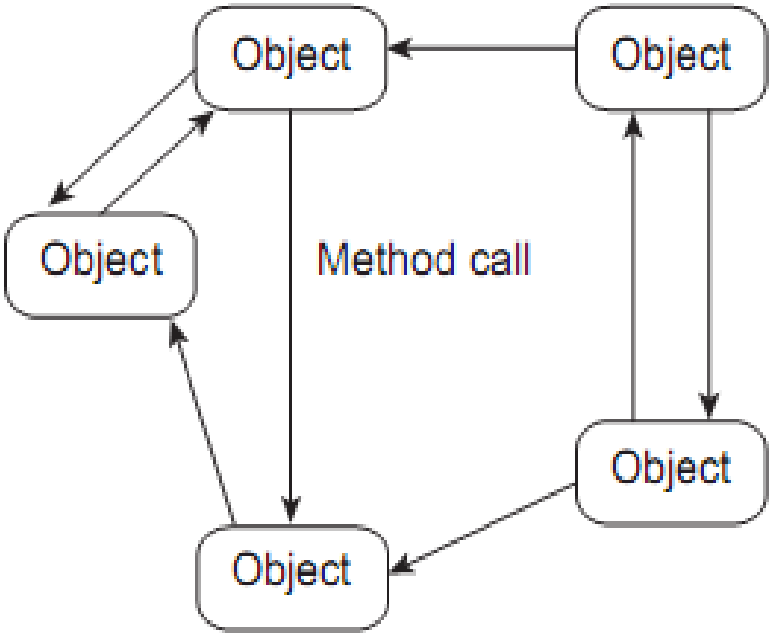


Arquiteturas baseadas em objetos (1/2)

- Idéia Básica: Cada objeto corresponde ao que definimos como componente, e esses componentes são conectados por meio de chamada de procedimento (remota), p. ex., Java RMI

- Exemplo: Aplicação distribuída de uma rede de locadoras, onde clientes podem alugar DVDs em diversas filiais.

Arquiteturas baseadas em objetos (2/2)

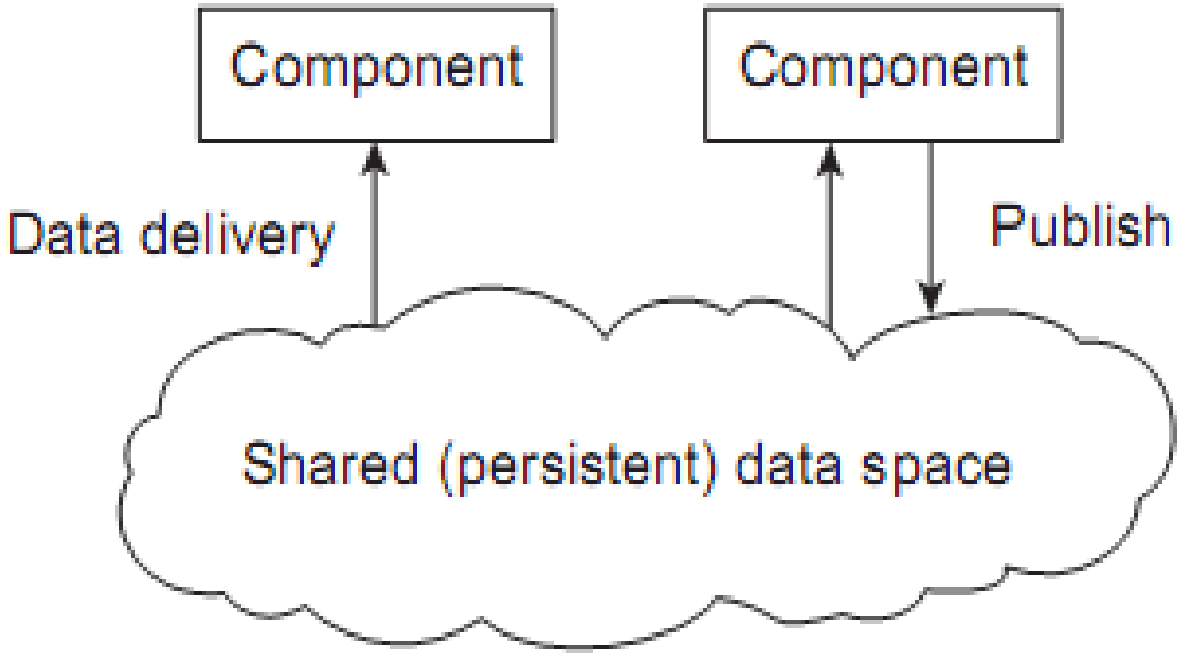


Arquiteturas centradas em dados (1/2)

- Idéia Básica: Processos se comunicam por meio de um repositório comum.

- Exemplo: Grande conjunto de aplicações em rede que dependem de um sistema distribuído de arquivos compartilhados o qual praticamente toda a comunicação ocorre por meio de arquivos: Web

Arquiteturas centradas em dados (2/2)



Arquiteturas centradas em eventos (1/3)

- Idéia Básica: Nesta arquitetura, processos demonstram o interesse por um evento ou conjunto de eventos (processo se subscreve) e esperam pela notificação de qualquer um desses eventos, gerados por um processo notificador. Em outras palavras, o produtor publica uma informação em um gerenciador de eventos (middleware),e os consumidores se subescrevem para receber as informações deste gerenciador. Eventos e notificações.
- Exemplo: Consultas em vários bancos de dados

Arquiteturas centradas em eventos (2/3)

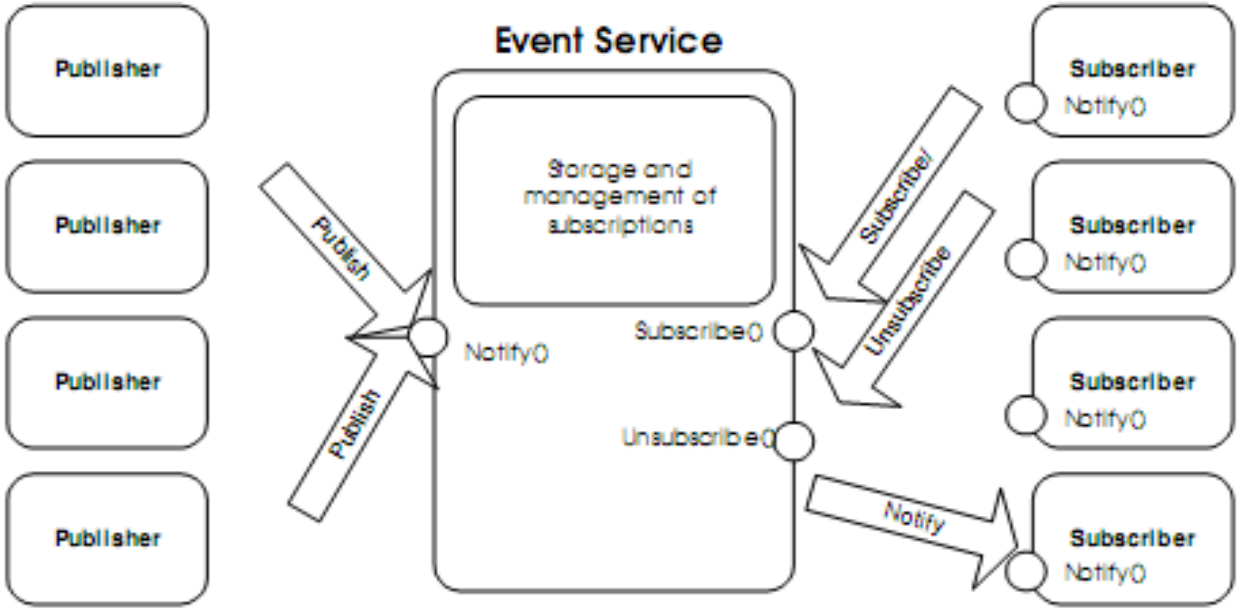


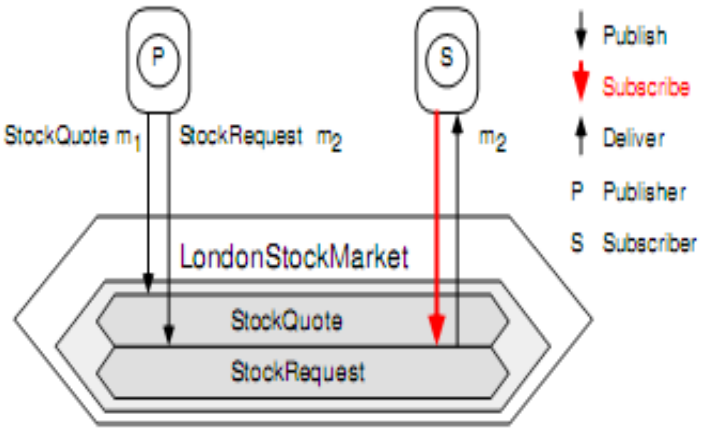
Figure 1: A simple object-based publish/subscribe system

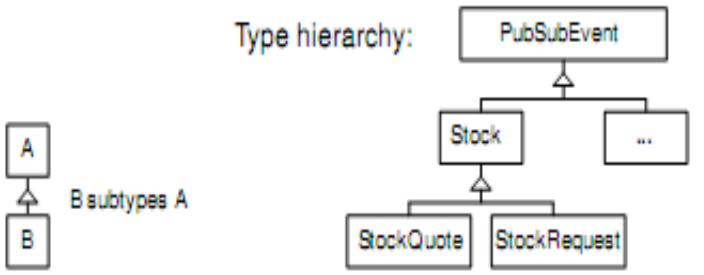
The Many Faces of Publish/Subscribe – Eugster et al. *ACM Comput. Surv.* (35)2:114-131

Arquiteturas centradas em eventos (3/3)

```
public class PubSubEvent implements Serializable {
  public String id;
}
public class Stock extends Event {
  public String company;
  public int amount;
  public String traderId;
}
public class StockQuote extends Stock {
  public float price;
}
public class StockRequest extends Stock {
  public float minPrice;
  public float maxPrice;
}
public class StockSubscriber implements Subscriber<Stock> {
  public void notify(Stock s) {
    System.out.print("Trader " + s.traderId);
    System.out.println(" deals with " + s.company);
  }
}
// ...
Subscriber<Stock> sub = new StockSubscriber();
EventService.subscribe<Stock>(sub);
```

(a)





(b)

The Many Faces of Publish/Subscribe – Eugster et al. *ACM Comput. Surv.* (35)2:114-131

Arquiteturas de Sistemas (1/2)

Como diversos sistemas distribuídos são realmente organizados?

Onde são colocados os componentes de *software*?

Como é estabelecida a interação entre as peças de *software*?

Arquiteturas de Sistemas (2/2)

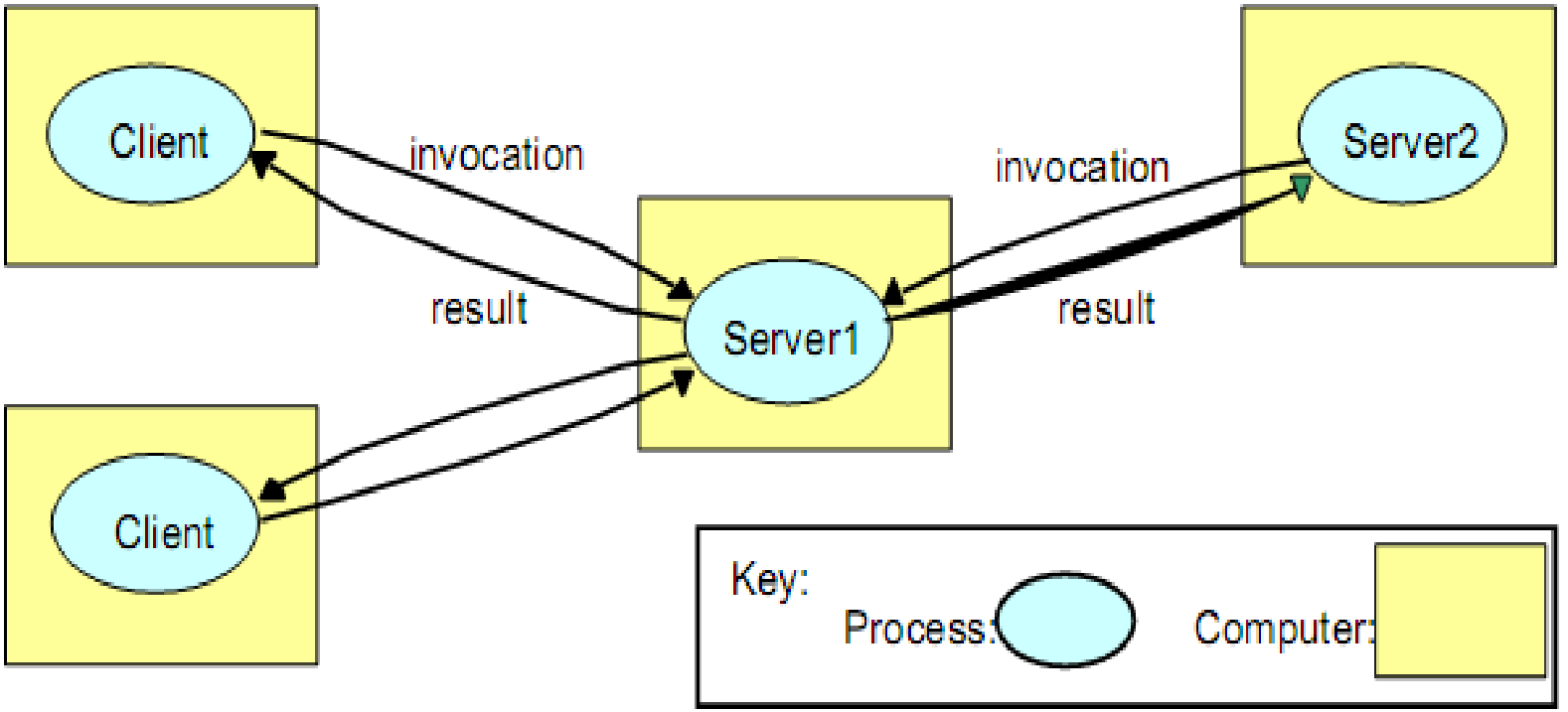
- Arquiteturas Centralizadas
  - Cliente-Servidor: vídeo sob demanda, terminais bancários
- Arquiteturas Descentralizadas
  - Peer-to-peer (P2P): Chord
- Arquiteturas Híbridas
  - Peer-to-peer (P2P): BitTorrent, PPLive

Arquiteturas Centralizadas (1/14)

Modelo Cliente-Servidor

- Processos são divididos em dois grupos (possível sobreposição)
- Servidor: processo que implementa um serviço específico
- Cliente: processo que requisita um serviço ao servidor. Requisição → Resposta

Arquiteturas Centralizadas (2/14)



Arquiteturas Centralizadas (3/14)

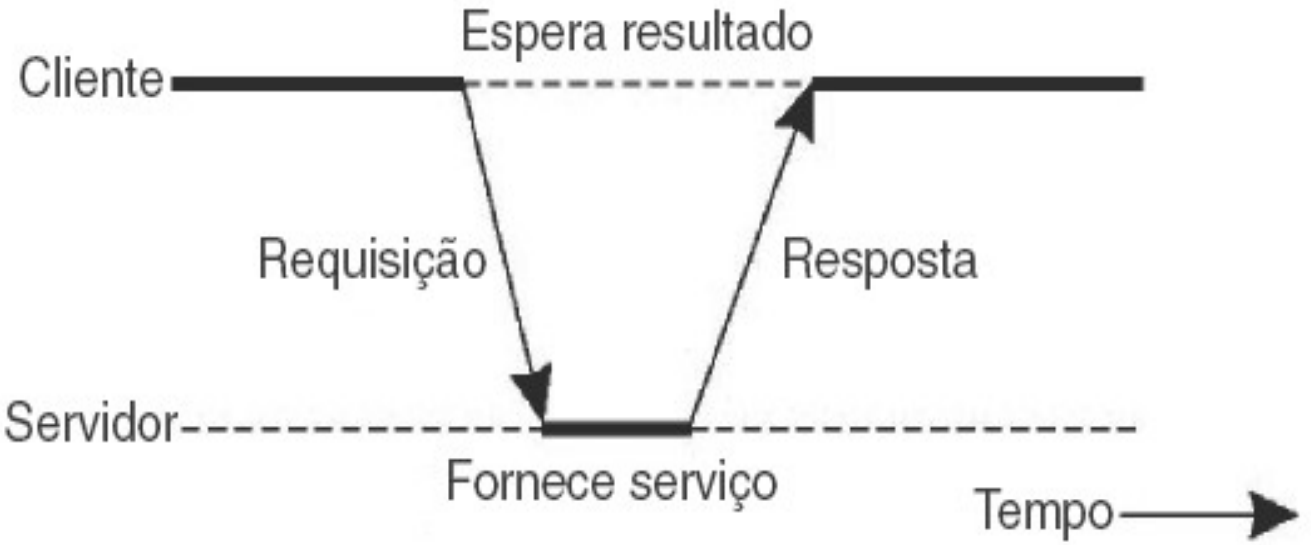


Figura 2.3 Interação geral entre um cliente e um servidor.

Arquiteturas Centralizadas (4/14)

- Modelo Cliente-Servidor: questões, questões!!
  - Clientes e Servidores *multithreads*?
  - Comunicação???
    - Qual o tipo de aplicação?
      - Sem conexão, não confiável (UDP)?
      - Com conexão, confiável (TCP)?

Arquiteturas Centralizadas (5/14)

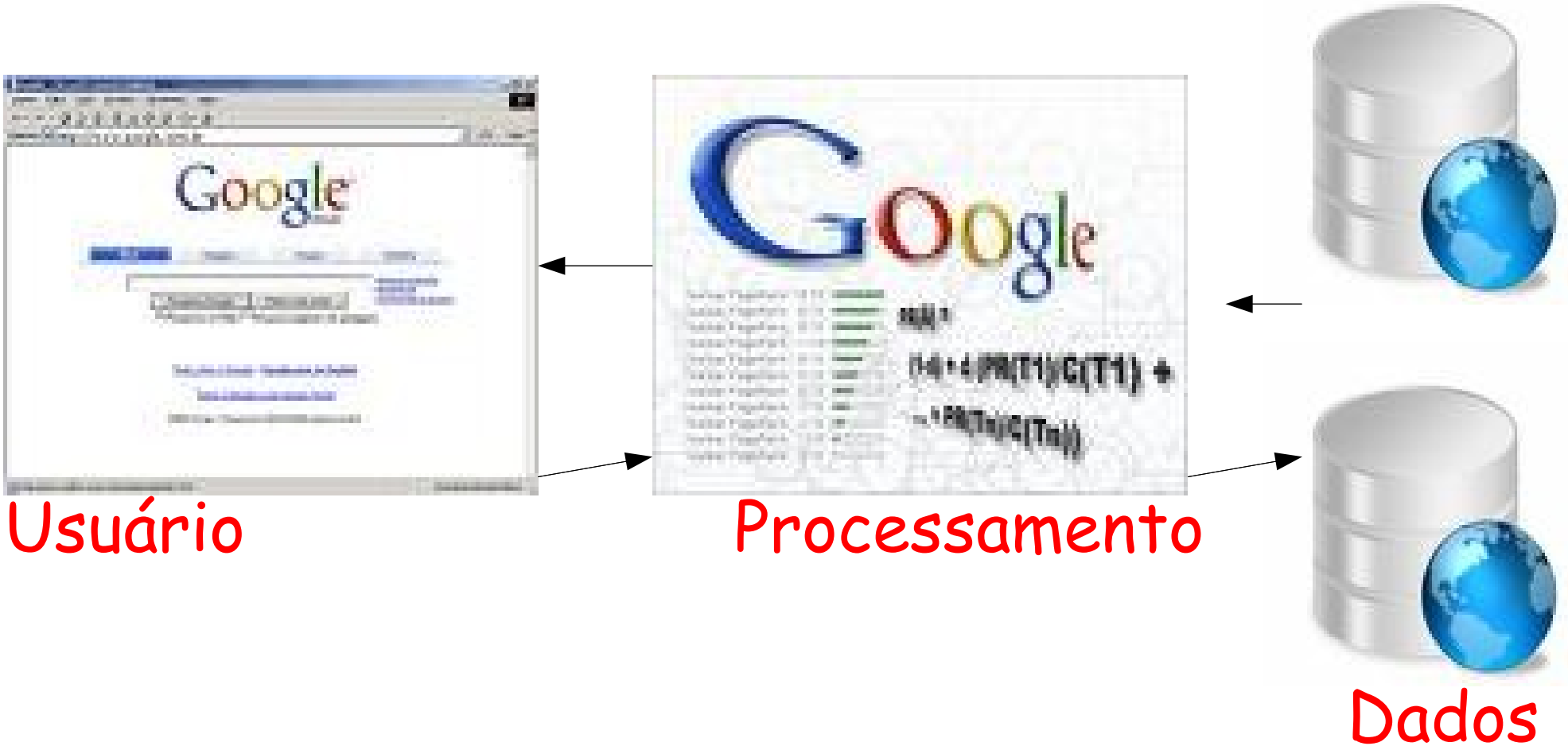
Camadas de Aplicação (estilo arquitetônico)

- Considerando aplicações cliente-servidor que visam dar suporte ao acesso de usuários a banco de dados:
  - Nível de interface
  - Nível de processamento
  - Nível de dados

Arquiteturas Centralizadas (6/14)

Camadas de Aplicação

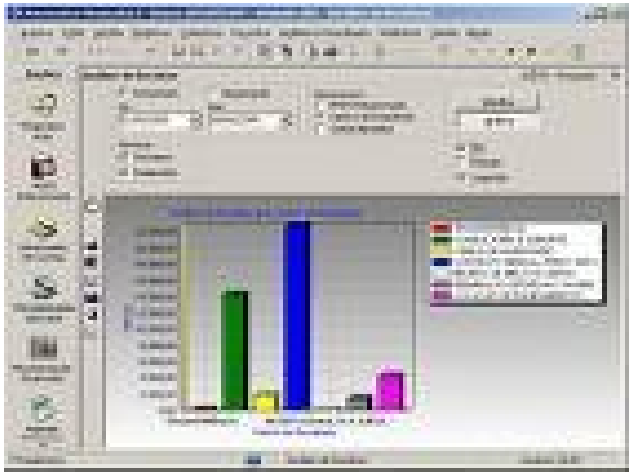
- Exemplo: Google



Arquiteturas Centralizadas (7/14)

Camadas de Aplicação

- Exemplo: Suporte à decisão (corretora de valores)



Usuário

Processamento

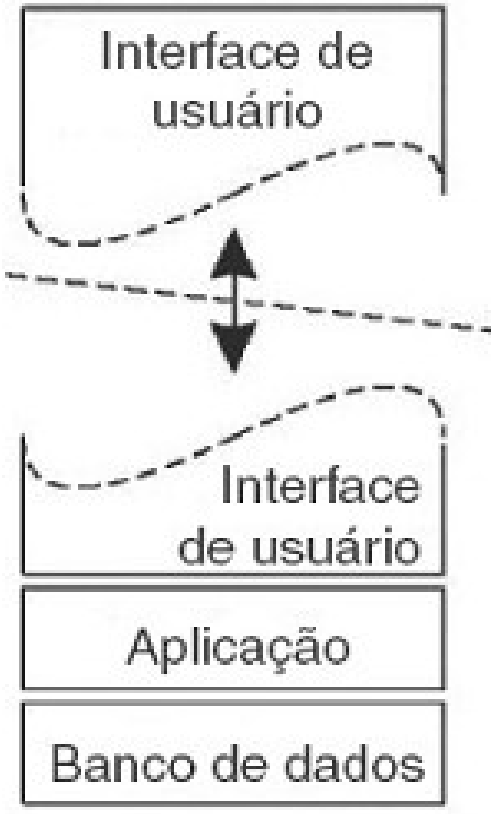
Dados

Arquiteturas Centralizadas (8/14)

Com a distinção entre três níveis lógicos, como distribuir fisicamente uma aplicação cliente-servidor por várias máquinas?

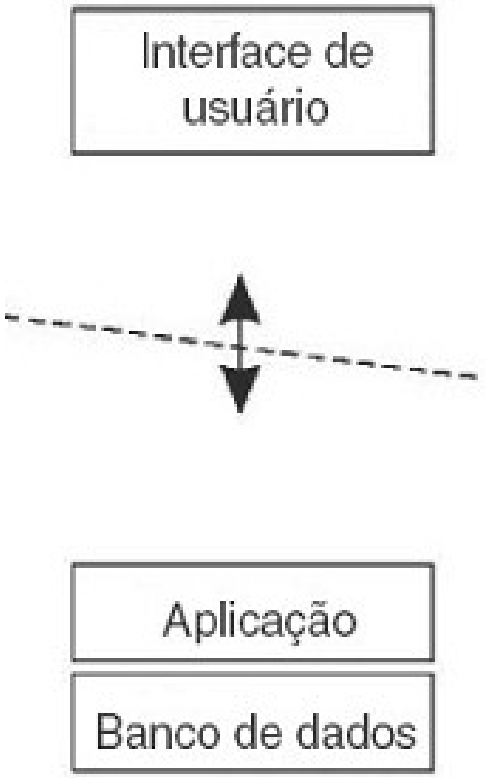
- Arquitetura de duas divisões físicas
- Arquitetura de três divisões físicas

Arquiteturas Centralizadas (9/14)



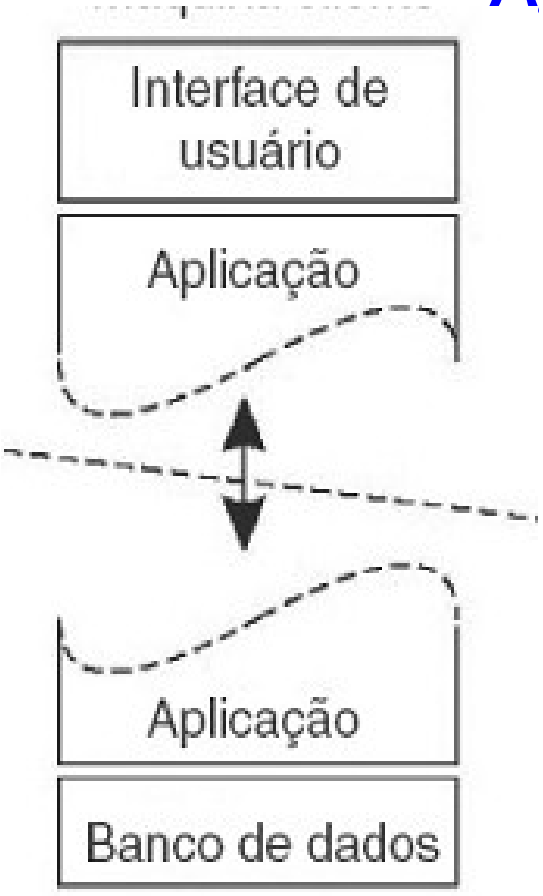
- Arquitetura de duas divisões físicas
  - Parte da interface é dependente de terminal
  - Aplicações controlam remotamente a apresentação dos dados

Arquiteturas Centralizadas (10/14)



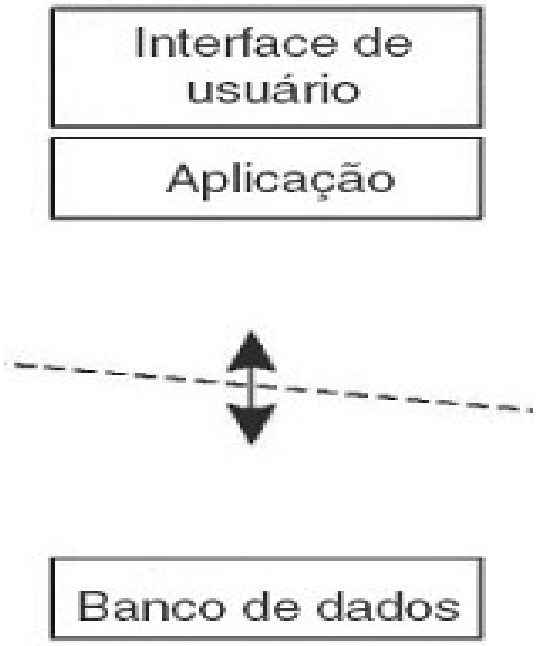
- Arquitetura de duas divisões físicas
  - Nesse modelo, o software cliente não faz nenhum processamento exceto o necessário para apresentar a interface da aplicação

Arquiteturas Centralizadas (11/14)



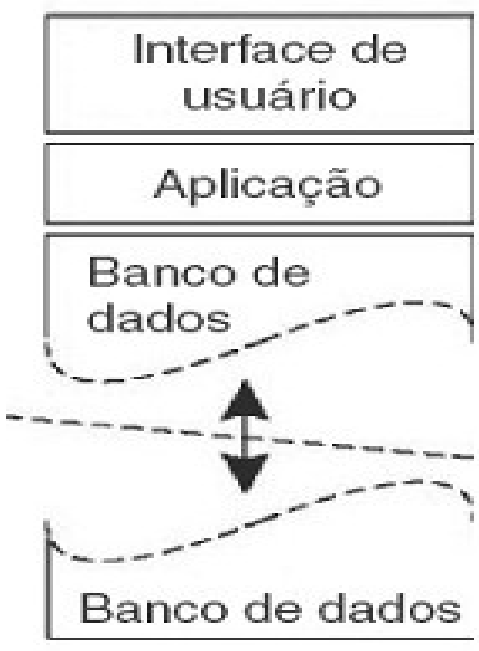
- Arquitetura de duas divisões físicas
  - Formulário que precise ser completamente preenchido antes do processamento. Cliente pode verificar a correção e consistência
  - Editor de texto com funções básicas no cliente e ferramentas avançadas no servidor

Arquiteturas Centralizadas (12/14)



- Arquitetura de duas divisões físicas
  - Pcs conectados por meio de uma rede a um sistema de arquivos distribuídos ou a um banco de dados

Arquiteturas Centralizadas (13/14)



- Arquitetura de duas divisões físicas
  - Consulta a Web, com browser um cliente pode construir gradativamente uma enorme cache em disco local com as páginas Web mais recentemente consultadas

Arquiteturas Centralizadas (14/14)

Arquiteturas de três divisões físicas

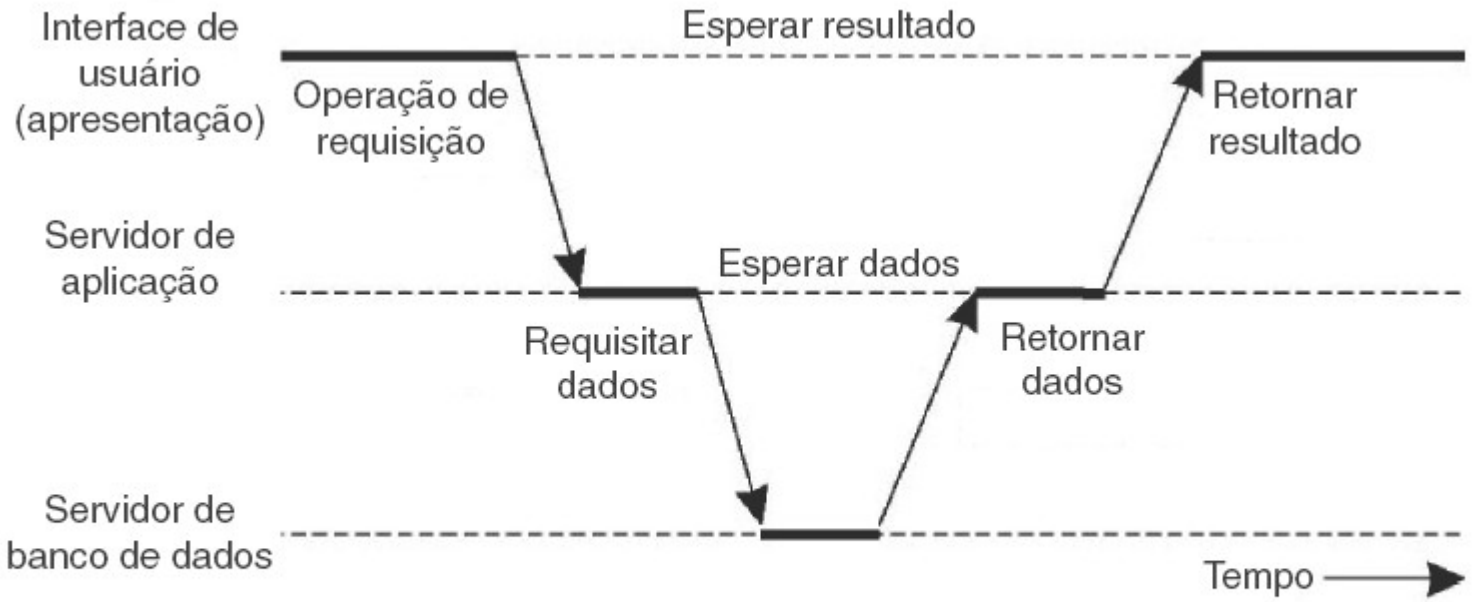


Figura 2.6 Exemplo de um servidor que age como cliente.

Arquiteturas Descentralizadas (1/8)

Clientes e servidores são fisicamente subdivididos em partes logicamente equivalentes, mas cada parte está operando em sua própria porção do conjunto completo de dados, o que equilibra a carga!!!!

Interação entre os processos é simétrica: cada processo agirá como um cliente e um servidor ao mesmo tempo

Arquiteturas Descentralizadas (2/8)

Sistemas P2P – questões, questões, questões!!

- Como organizar os peers em uma rede de sobreposição (*overlay*)?
- Como difundir o conteúdo?
- Como incentivar os peers a colaborarem?

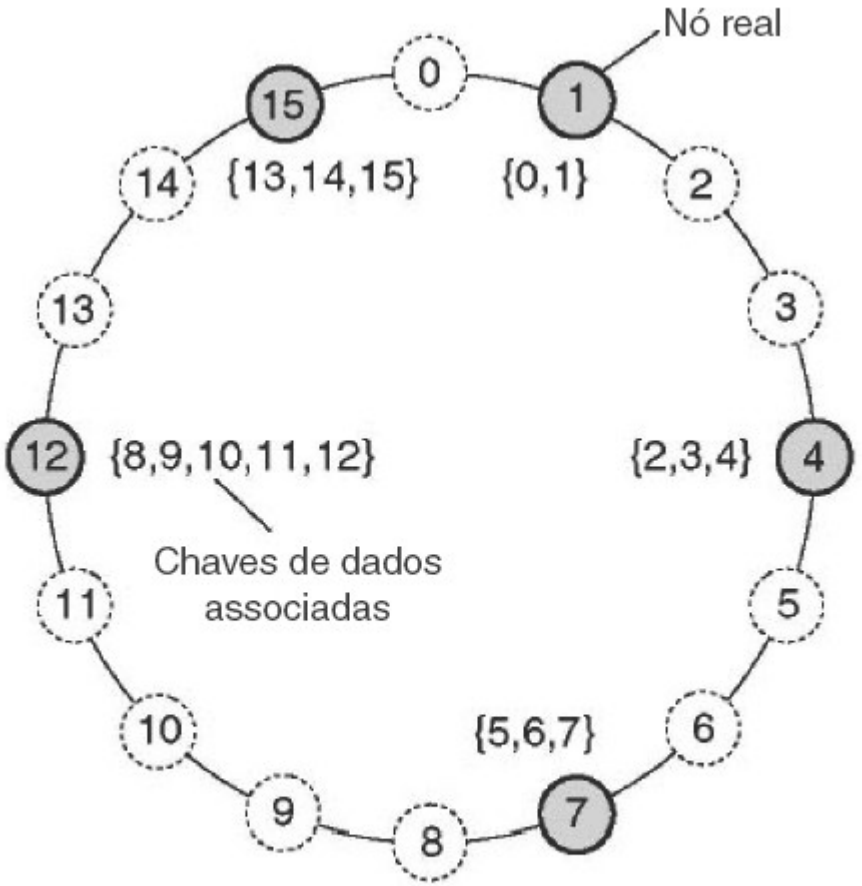
Arquiteturas Descentralizadas (3/8)

Considerando o *overlay* e modo de construção

- Redes Estruturadas → procedimento determinístico para definição do *overlay*, p.ex, tabela de hash distribuída (DHT)
- Redes Não-estruturadas → algoritmos aleatórios para construção da rede de sobreposição, gerando um grafo aleatório

Arquiteturas Descentralizadas (4/8)

Arquiteturas P2P estruturadas

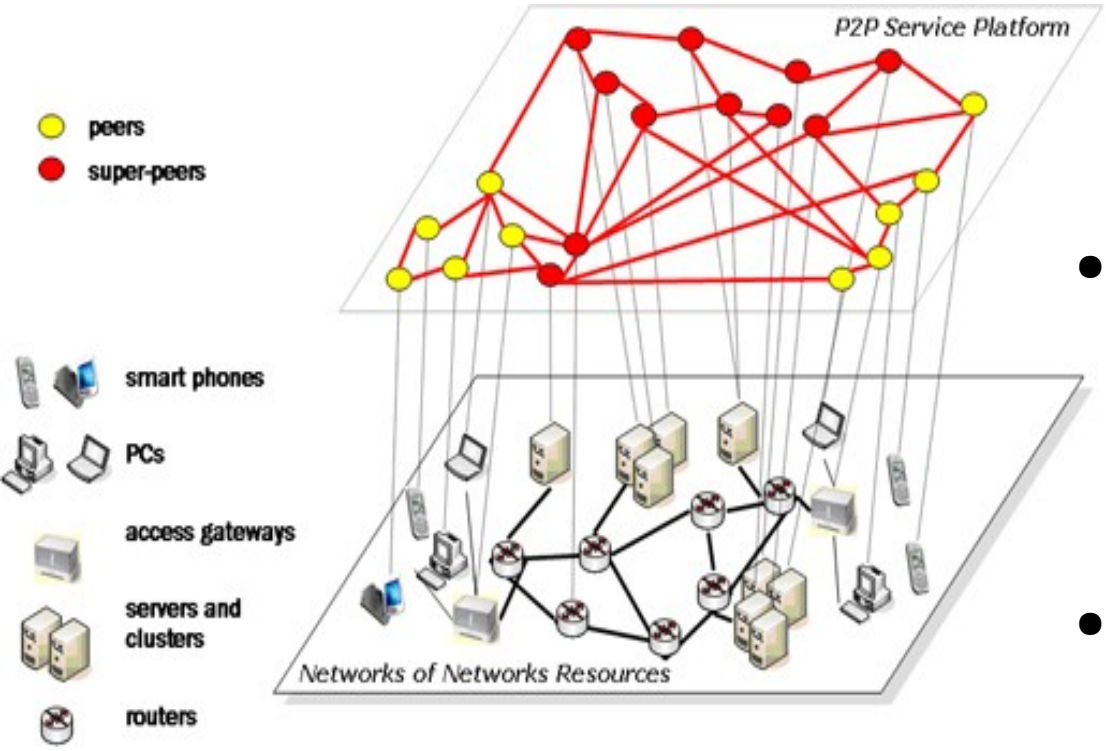


- Sistema Chord (Stoica et al, 2003)
- Nós estão logicamente organizados em um anel
- Item de dado com chave k é mapeado para o nó com id >=
- LOOKUP(k)

Mapeamento de itens de dados para nós em Chord.

Arquiteturas Descentralizadas (5/8)

Arquiteturas P2P não-estruturadas



- Algoritmos aleatórios
- Cada peer possui uma lista de vizinhos (visão parcial)
- Para encontrar dados, inundar a rede (no pior caso)
- Importante atualizar a lista de vizinhos
  - Mas como?

Arquiteturas Descentralizadas (6/8)

Arquiteturas P2P não-estruturadas

- *Threads* que solicitam aos vizinhos a visão parcial (*pull*) ou que empurram (*push*) a visão a seus vizinhos
- Algoritmos que atualizem a vizinhança a cada *x* unidades de informação enviadas

Arquiteturas Descentralizadas (7/8)

Arquiteturas P2P não-estruturadas

- Um dos problemas: como encontrar os dados de maneira eficiente
- Muitos sistemas utilizam nós especiais, que possuem um índice de itens de dados → *Superpeers*
  - Características especiais?
  - Como associar *peers* comuns a estes *superpeers*
  - Como escolher estes *peers*?

Arquiteturas Descentralizadas (7b/8)

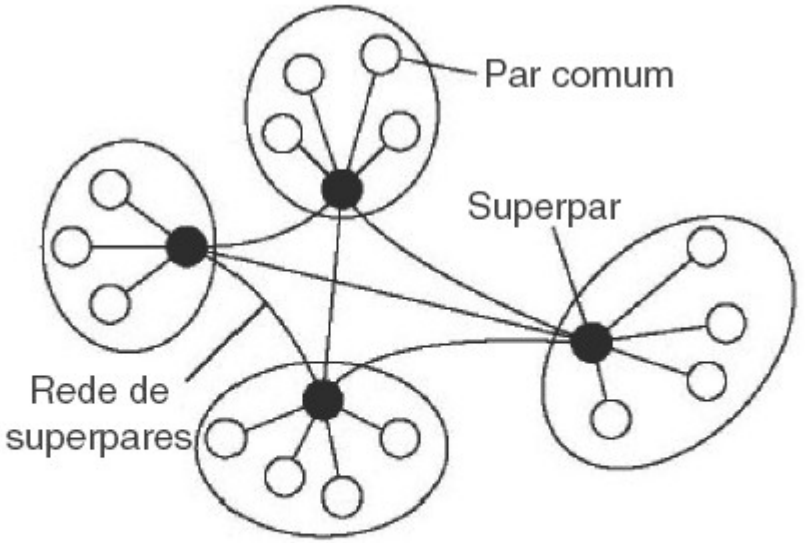
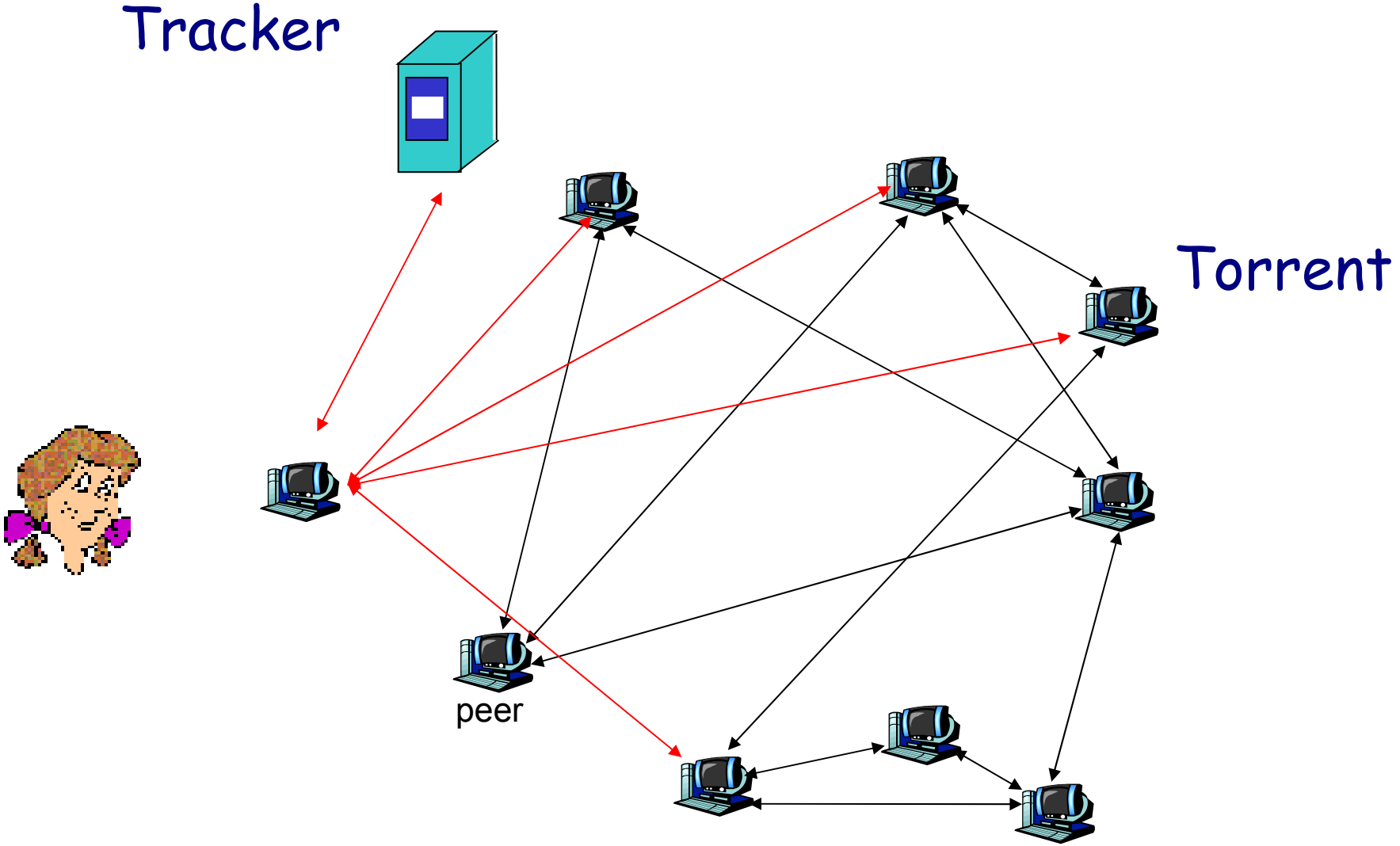


Figura 2.12 Organização hierárquica de nós em uma rede de superpares.

Arquiteturas Descentralizadas (8/8)

Arquiteturas Híbridas

BitTorrent(Cohenm, 2003)



Arquiteturas Descentralizadas

Arquiteturas Híbridas

BitTorrent(Cohenm, 2003)

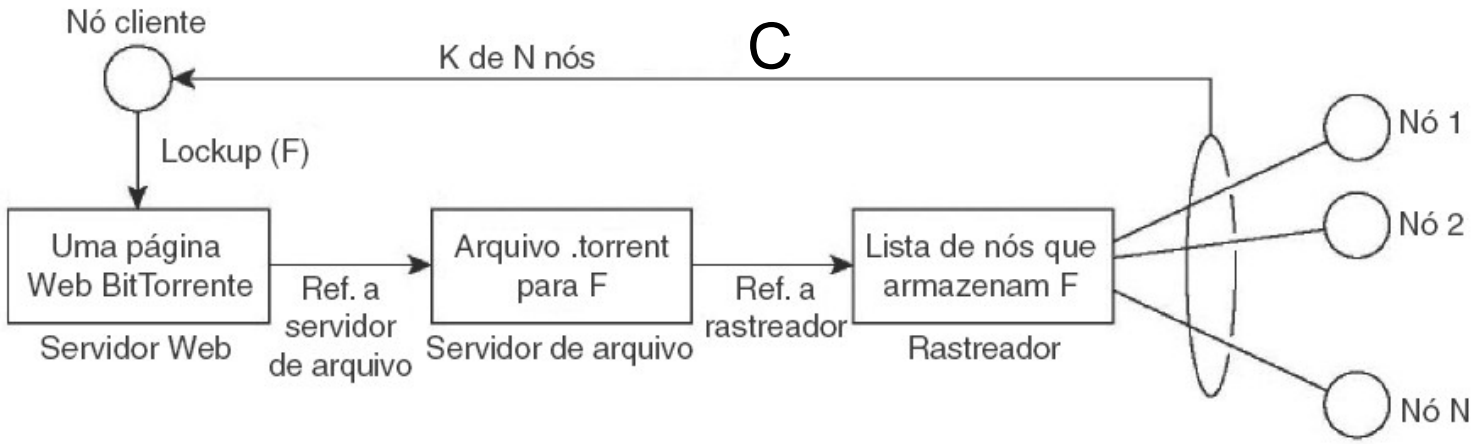


Figura 2.14 Funcionamento principal do BitTorrent [adaptado com permissão de Pouwelse et al. (2004)].

Arquiteturas *versus* Middleware

Cada middleware possui um estilo arquitetônico específico, com o objetivo de simplificar o projeto de aplicações

- Como fazer com que um middleware genérico se adapte a aplicação?
  - Middleware "inchado"
  - Interceptores podem ser usados para interromper o fluxo de execução para que uma procedimento específico da aplicação possa ser executado

Arquiteturas *versus* Middleware

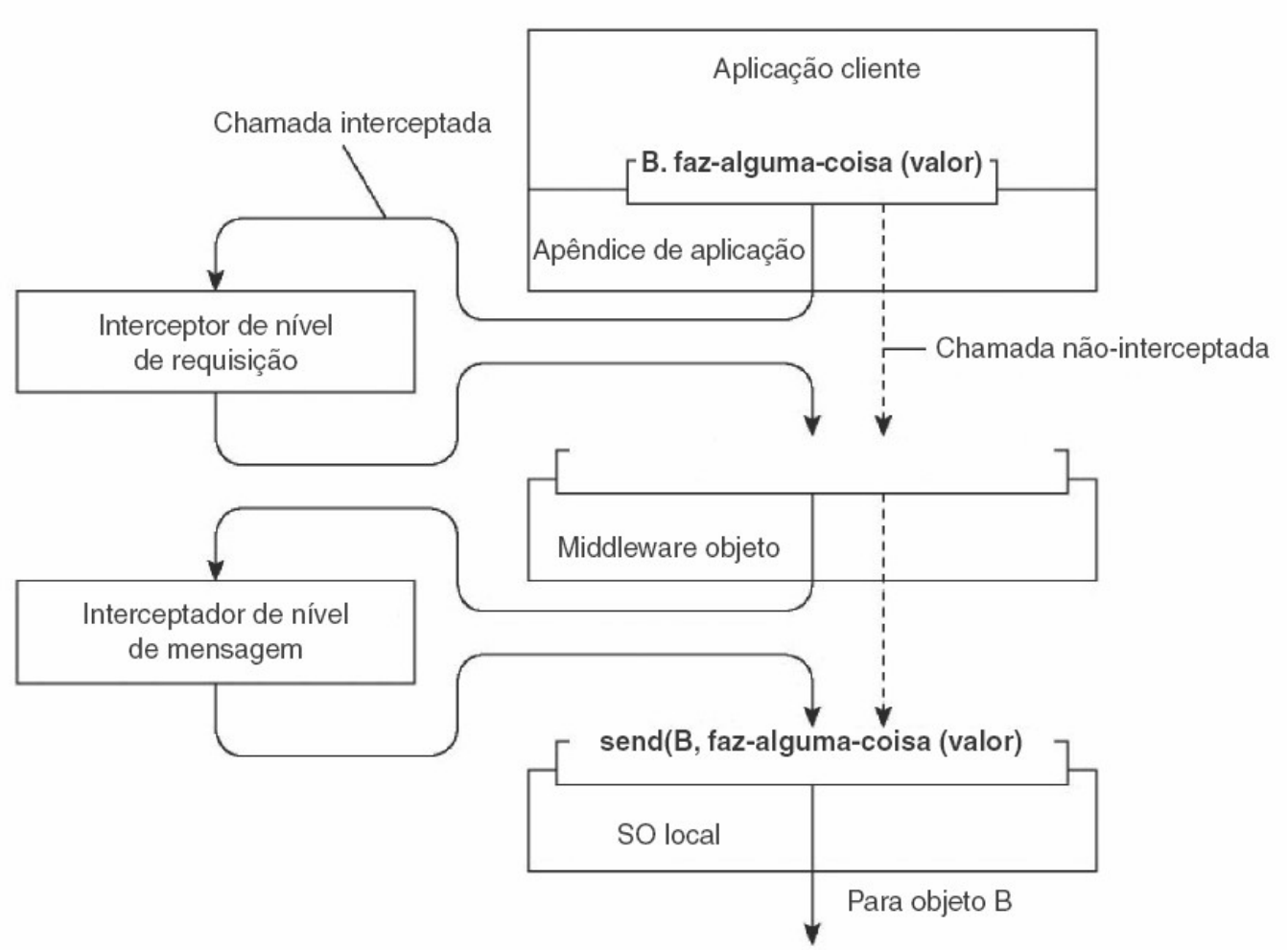


Figura 2.15 Utilização de interceptadores para manipular invocações de objeto remoto.

Autogerenciamento em SDs

Sistemas distribuídos devem ser capazes de reagir a mudanças em seu ambiente

- Ex: Troca dinâmica de políticas de alocação de recursos; Troca de codificação voz/vídeo para se adequar a condições de congestionamento na rede

Desafio: Deixar que o comportamento reativo ocorra sem intervenção humana!

Autogerenciamento em SDs

Idéia: Através de observações do comportamento do SD, componentes de estimativa de medições e de análise coletam dados e realimentam o sistema, modificando parâmetros controláveis.

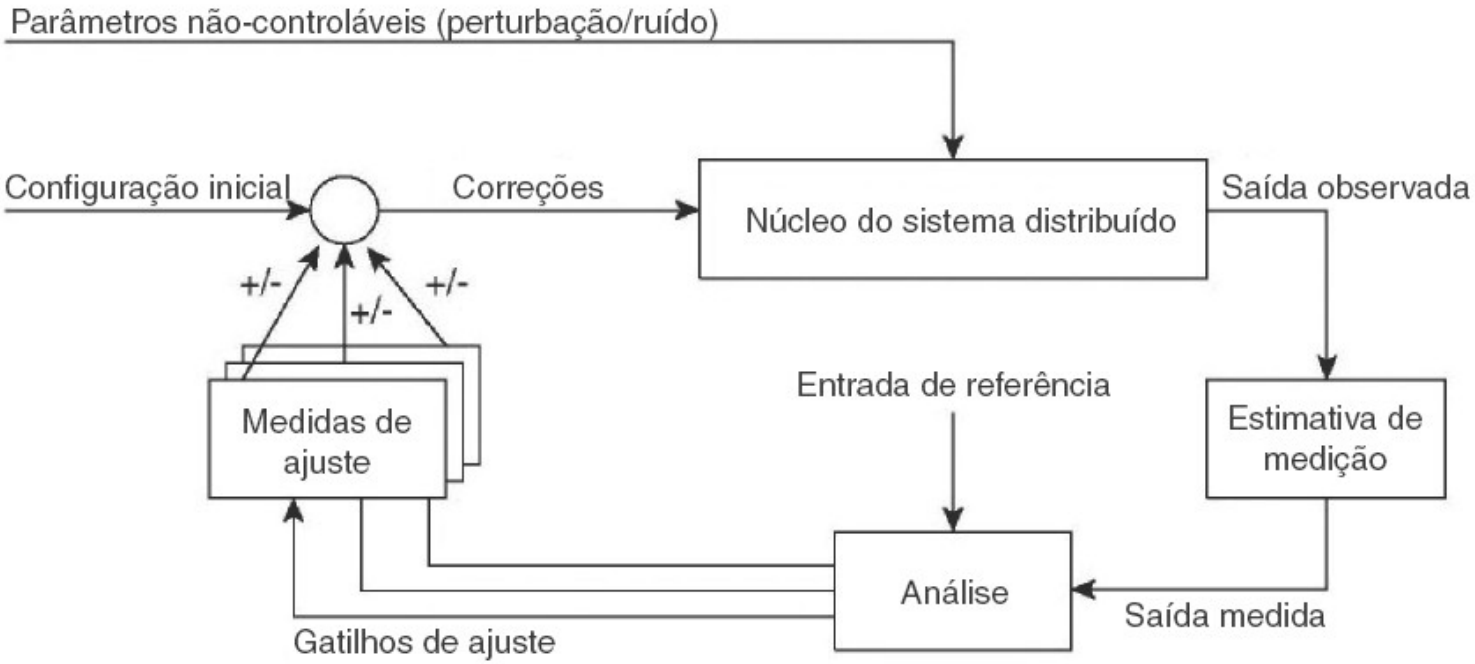


Figura 2.16 Organização lógica de um sistema de realimentação de controle.

Autogerenciamento em SDs

Alguns Exemplos:

- Astrolabe: ferramenta para observação de Sds. Resultados podem ser usados para alimentar um componente de análise para possíveis ações corretivas

Resumo

- SDs podem ser organizados de diferentes maneiras.
  - Software: Estilos arquitetônicos
  - Físico: Arquitetura de sistemas

- O projeto de sistemas autogerenciadores visa a adaptação do SD ao ambiente em que está sendo executado, obtendo um maior desempenho do mesmo