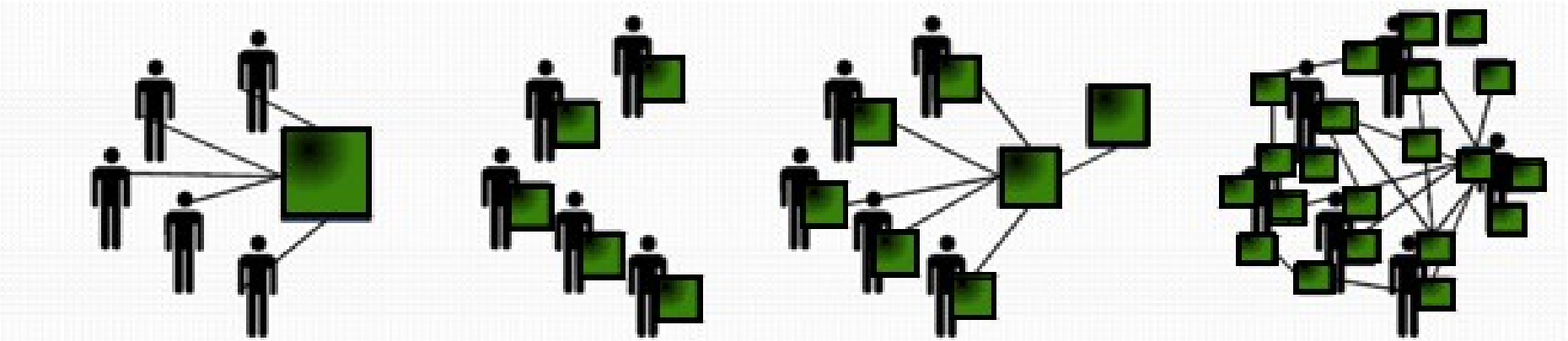
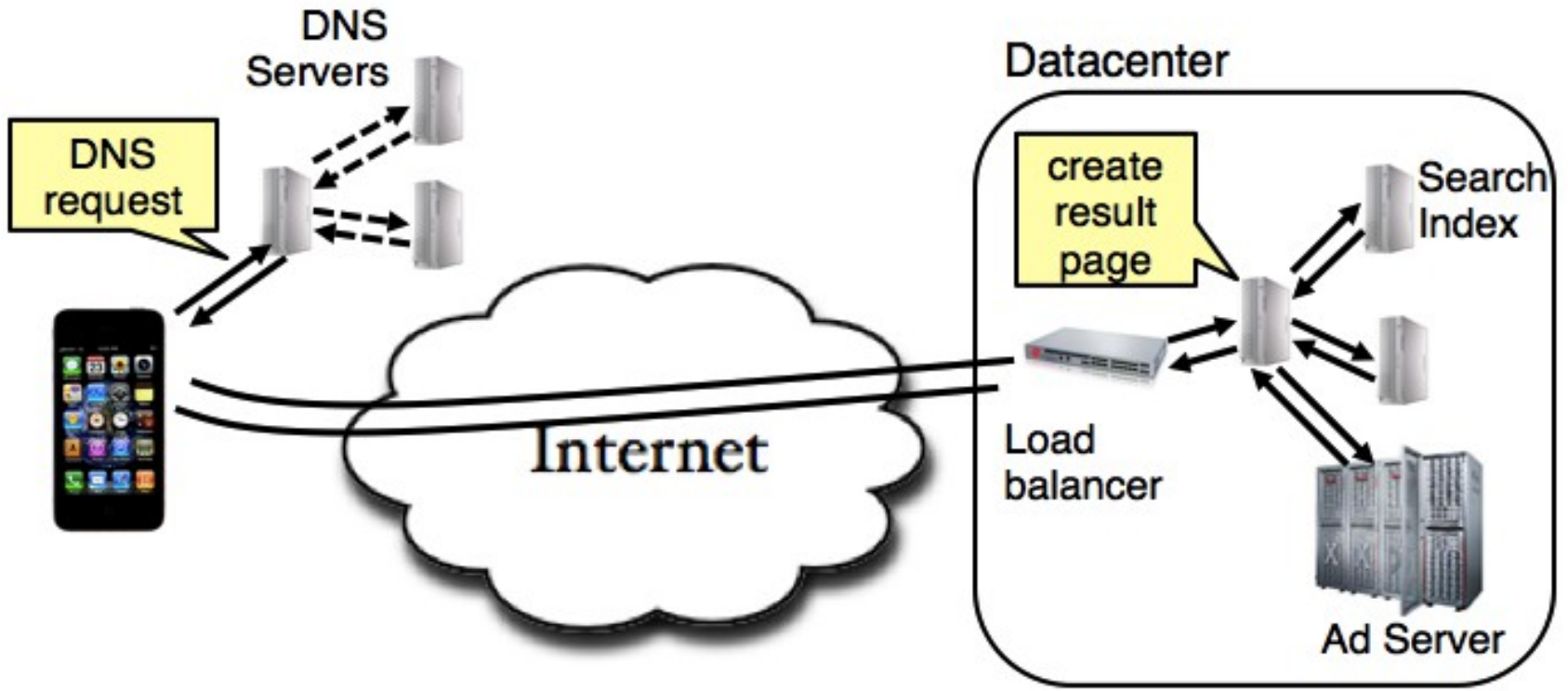
# Disciplina
# Sistemas de Computação

Aula 15



# Próximas aulas

- Os diversos níveis de um Sistema Computacional:
  - Organização e Arquitetura
  - Sistema Operacional
  - Tarefas distribuídas



# Definição de SD

- "Um sistema distribuído é uma coleção de computadores autônomos conectados por uma rede e equipados com um sistema de software distribuído." [Coulouris]

- "Um sistema distribuído é uma coleção de computadores independentes que aparenta ao usuário ser um computador único." [Tanenbaum]

# Outra definição de SD

- "Você sabe que tem um sistema distribuído quando a falha de um computador do qual você nunca ouviu falar faz com que você pare completamente de trabalhar." [Leslie Lamport]

# Avanços tecnológicos

- Invenção de redes de computadores  de alta velocidade (anos 70):
  - Rede local (Local Area Network - LAN)
  - Rede global (Wide Area Network - WAN)
- Desenvolvimento de microprocessadores potentes (anos 80).

# Estado da arte

- É relativamente fácil agrupar um grande número de CPUs, conectando-as por uma rede de alta velocidade.
- O software para sistemas distribuídos é completamente diferente do software para sistemas centralizados e está apenas começando a se desenvolver.

# Vantagens de SD sobre SC

- Melhor relação custo/benefício
- Capacidade de processamento além dos limites práticos de SC (velocidade da luz, aquecimento)
- Maior domínio de aplicações
- Maior confiabilidade e disponibilidade
- Crescimento gradativo da capacidade de processamento

# Vantagens de SD sobre PCs independentes

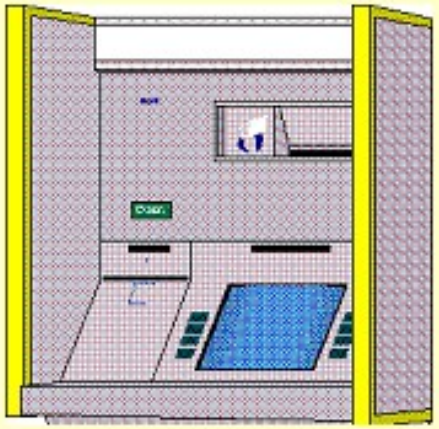
- Compartilhamento de dados comuns entre usuários
- Compartilhamento de recursos de hardware e software
- Comunicação entre pessoas
- Flexibilidade na distribuição de tarefas de acordo com as aplicações

# Desvantagens de SD

- Falta de software adequado
- Falhas e saturação da rede de comunicação podem eliminar as vantagens de SD
- Segurança pode ser comprometida:
  fácil acesso a dados e recursos reservados

# Qual é a Importância de SDs?

Distributed computer systems are critical for functioning of many organisations



**Banks** **Transport** **Telecommunications**

# Problemas!!!!

**No pain, no gain!**

# Desafios para os SDs

- Heterogeneidade
  - redes, hardware, S.O., linguagens e implementações de diferentes desenvolvedores
  - adoção de padrões: protocolos, representação dos dados
- Concorrência
  - execução simultânea de programas/processos
  - problemas no compartilhamento dos recursos e coerência
  - coordenação e controle de concorrência
- Inexistência de relógio global
  - falta da noção global única do tempo correto
  - sincronização, ordenação de eventos e estados globais
- Falhas independentes
  - rede, computadores e programas
  - detecção, mascaramento e recuperação de falhas

# Características Desejáveis de um SD

- Aberto
- Escalável
- Seguro
- Tolerante a falhas
- Transparente

# Características Desejáveis de um SD

- Aberto
  - poder ser estendido e reimplementado
    - especificar e documentar as principais interfaces de software dos componentes
    - definir um mecanismo de comunicação uniforme para acesso aos recursos compartilhados
- Escalável
  - permanecer eficiente com o aumento do número de recursos e usuários
    - controlar a perda de desempenho
    - evitar o esgotamento de recursos e os gargalos de desempenho

# Características Desejáveis de um SD (Cont.)

- Seguro
  - garantir confidencialidade, integridade e disponibilidade
    - autenticação, criptografia, robustez a ataques, etc.
- Tolerante a falhas
  - realizar a detecção, mascaramento e recuperação das falhas
- Transparente
  - ocultar a separação dos componentes
  - ocultar e transformar em anônimos os recursos irrelevantes para a execução de uma tarefa

# Transparência da Distribuição

- Consiste em ocultar o fato de que os processos e recursos estão fisicamente distribuídos por vários computadores

- Tipos:
  - Acesso
  - Localização
  - Migração
  - Replicação
  - Concorrência
  - Falha

# Transparência da Distribuição

- Acesso:
  - Ocultar diferenças em representação de dados, e o modo como os recursos podem ser acessados por usuários
  - Exemplo: representação de inteiros little endian, big endian
- Localização:
  - Usuários não podem dizer a localização física do recurso.
  - Exemplo: www.google.com (??????)

# Transparência da Distribuição

- Migração:
  - Recursos podem ser movimentados sem afetar o modo como podem ser acessados
  - Exemplo: Mudança de um servidor WEB
- Relocação (ou mobilidade):
  - Recursos podem ser relocados enquanto estão sendo acessados
  - Exemplo: uso móvel de laptops (redes wireless)

# Transparência da Distribuição

- Replicação:
  - Ocultar o fato de que existem várias cópias de um recurso
  - Exemplo: aumentar a disponibilidade ou melhorar o desempenho
- Concorrência:
  - Ocutar o fato que 2 ou mais usuários esteja acessando um recurso no mesmo instante
  - Consistência?
- Falha:
  - Ocultar do usuário que um recurso deixou de funcionar bem e que o sistema se recuperou da falha

# Transparência é sempre requerida?

- Bela meta no projeto e na implementação de sistemas distribuídos, mas deve ser considerada em conjunto com outras questões, como desempenho e facilidade de compreensão.

# Escalabilidade dos SDs: mais uma característica desejável

- Três Dimensões:
  - Tamanho: Facilidade em adicionar mais usuários e recursos ao sistema
  - Geográfico: Usuários e recursos podem estar longes uns dos outros
  - Administrativo: Facilidade de gerenciamento, mesmo que abranja muitas organizações administrativas diferentes

# Ciladas!!!

- Premissas que podem ser adotadas ao se desenvolver uma aplicação distribuída
  - A rede é confiável
  - A rede é segura
  - A rede é homogênea
  - A topologia não muda
  - A latência é zero
  - A largura de banda é infinita
  - O custo de transporte é zero
  - Há somente um administrador

# Tipos de SDs

- Classificação relacionada com a função principal do sistema
  - Computação Distribuída
  - Sistemas de Informação distribuídos
  - Sistemas Pervasivos

# Tipos de SDs: Computação Distribuída

- Obj: Oferecer computação de alto desempenho
  - Cluster versus Grade
  - Simulação de fenômenos físicos
  - Teste de novos protocos, aplicações distribuídas (PlanetLab, p.ex.)

# Tipos de SDs: Sistemas de Informação

- Sistemas de processamento de transações
  - Base de dados distribuída
- Integração de aplicações empresariais
- Internet
  - Rede Heterogênea (pela própria definição)
  - Serviços: email, www, VoIP, tranferência de arquivos

# Tipos de SDs: Sistemas Pervasivos

- Equipamentos costumam ser caracterizados por seu pequeno tamanho, pela alimentação por bateria, por sua mobilidade e por terem somente uma conexão sem fio
  - Redes de Sensores
  - Redes Celulares

# Arquiteturas de SDs

- Arquiteturas de Sistemas
  - Arquiteturas Centralizadas
  - Arquiteturas Descentralizadas
  - Arquiteturas  Híbridas

# 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?

# Arquiteturas de Sistemas

- 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

- Arquiteturas Centralizadas
  Cliente-Servidor
  - Vídeo sob demanda
  - Terminais bancários

- Arquiteturas Descentralizadas
  - Peer-to-peer (P2P): Chord

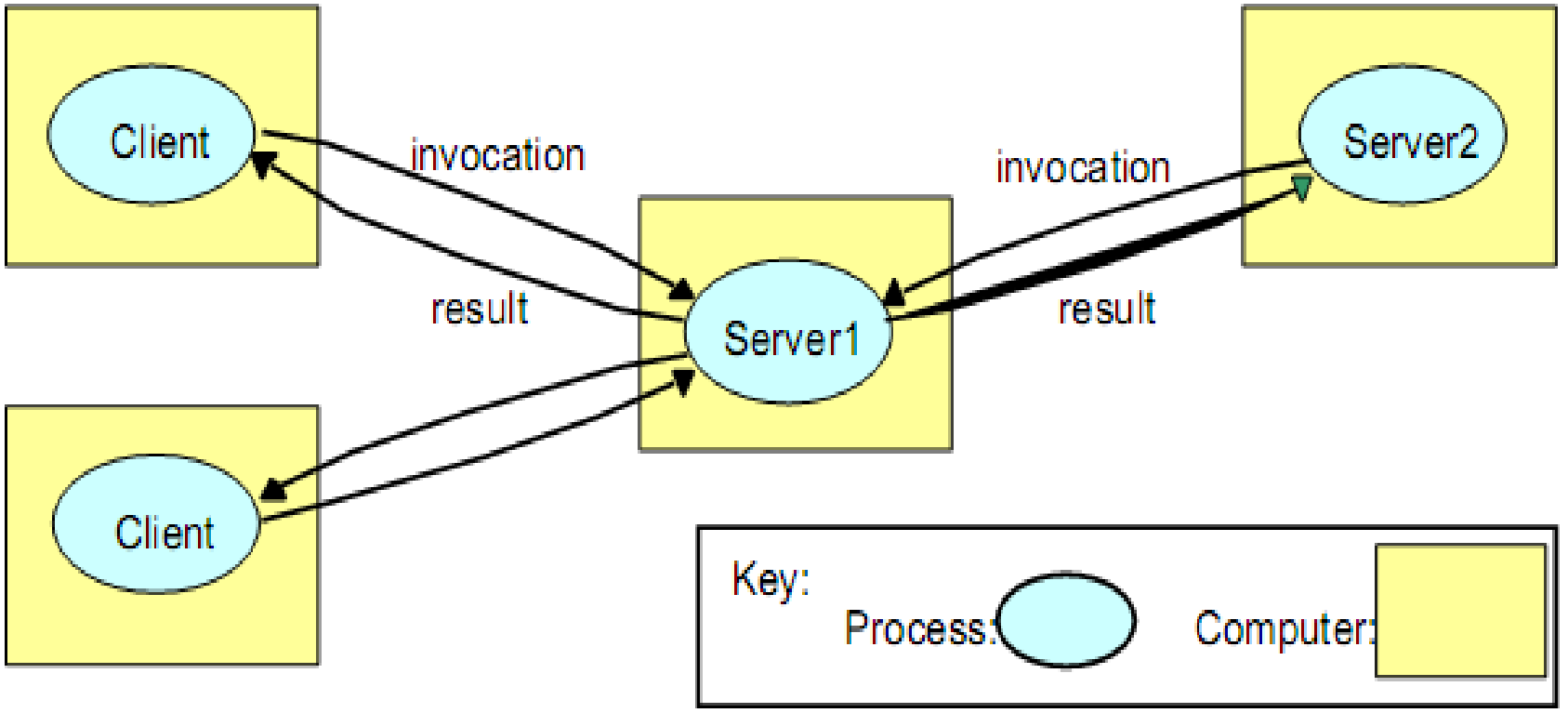
- Arquiteturas Híbridas
  - Peer-to-peer (P2P):
    BitTorrent, PPLive

# Arquiteturas Centralizadas

## 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 (Cont.)



# Arquiteturas Centralizadas (Cont.)

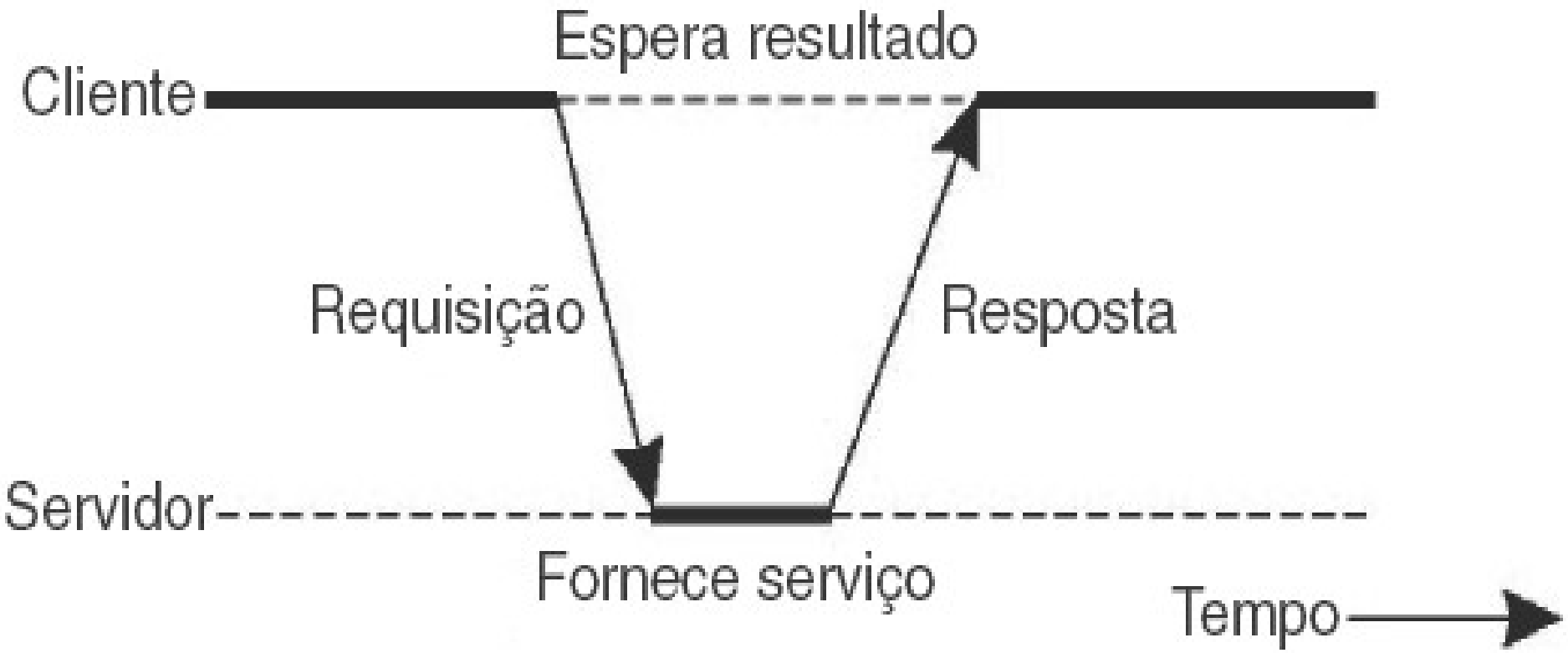



Figura 2.3 Interação geral entre um cliente e um servidor.

# Arquiteturas Centralizadas (Cont.)

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

# Arquiteturas Centralizadas (Cont.)

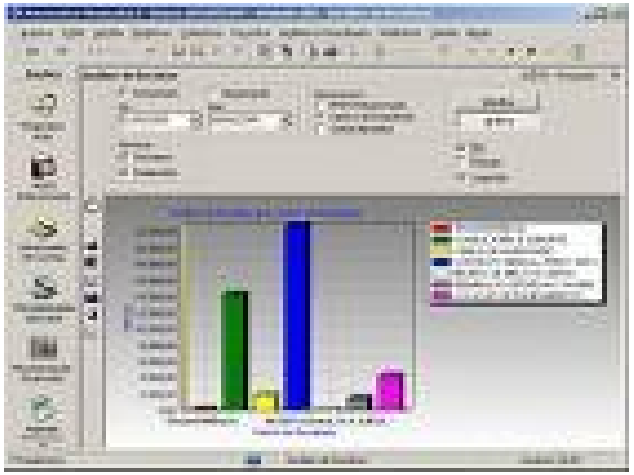
- Camadas de Aplicação (estilo arquitetônico)
  - Considerando aplicações cliente-servidor que visam dar suporte ao acesso de usuários a banco de dados:
    - Nível de interface
    - Nível de processamento
    - Nível de dados

# Arquiteturas Centralizadas (Cont.)

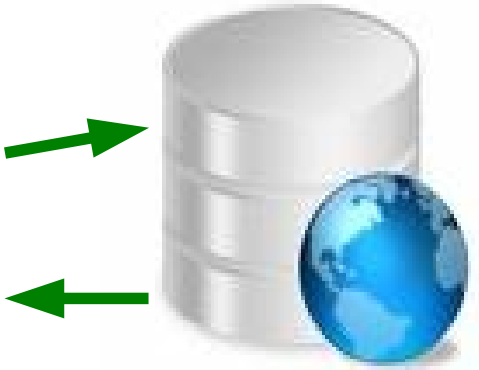
- Camadas de Aplicação
  - Exemplo: Google

# Arquiteturas Centralizadas (Cont.)

- Camadas de Aplicação
  - Exemplo: Suporte à decisão (corretora de valores)





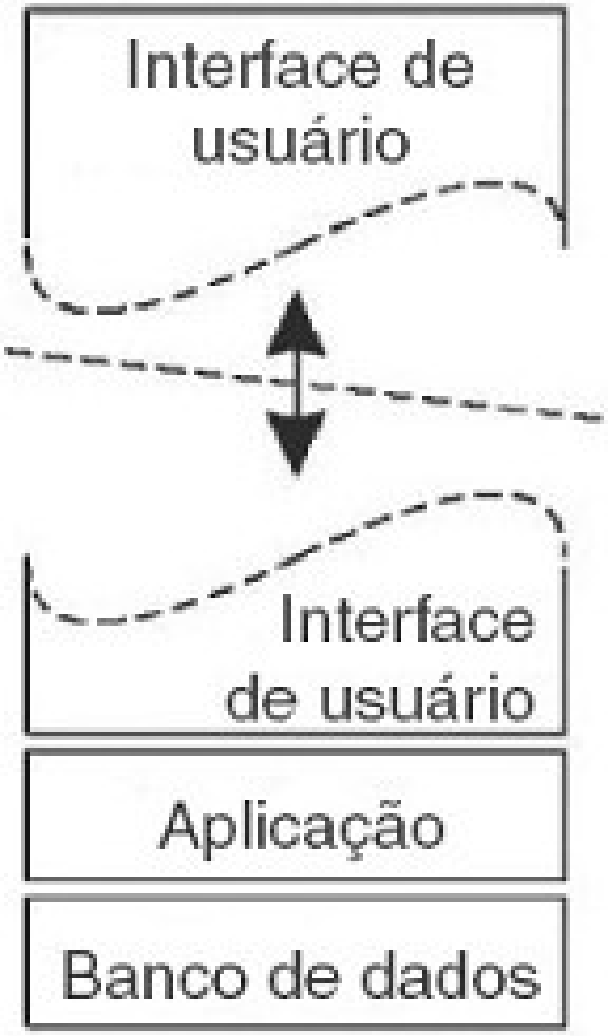


# Arquiteturas Centralizadas (Cont.)

- 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

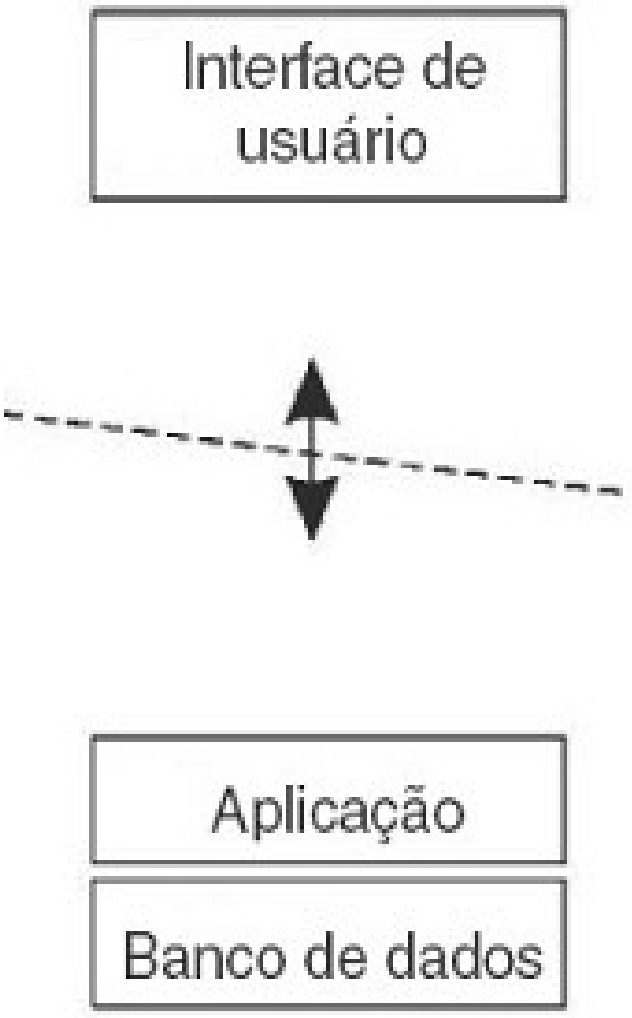
# Arquiteturas Centralizadas (Cont.)

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



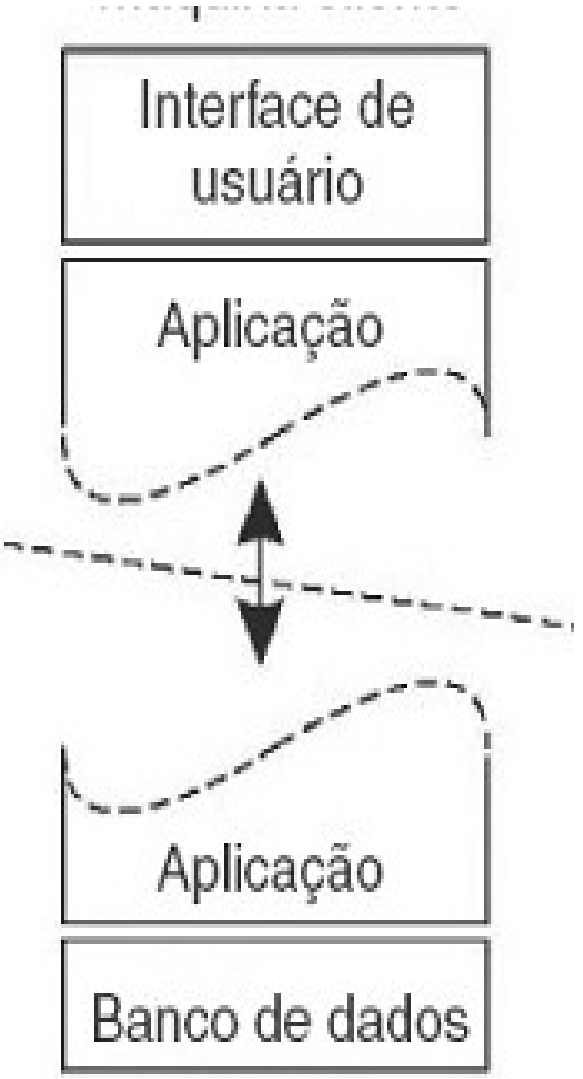
# Arquiteturas Centralizadas (Cont.)

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



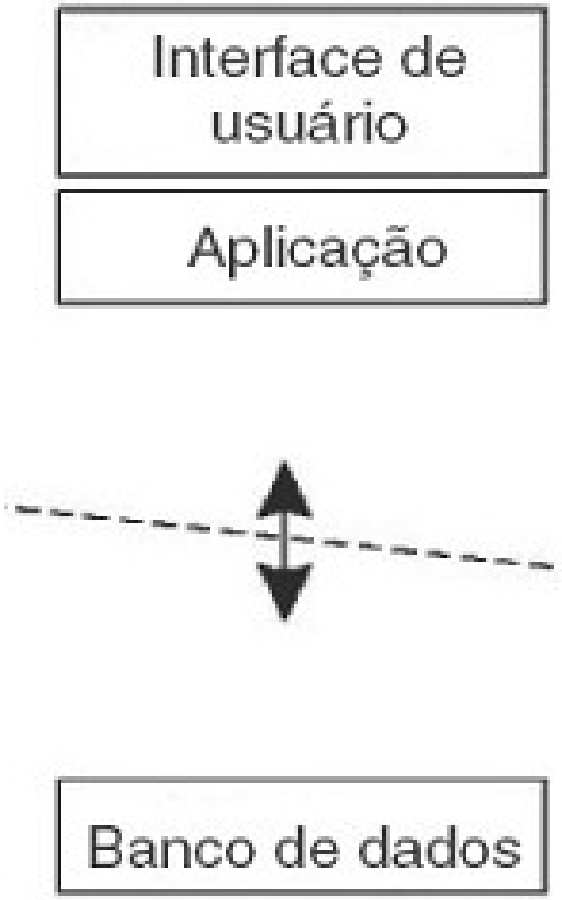
# Arquiteturas Centralizadas (Cont.)

- 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



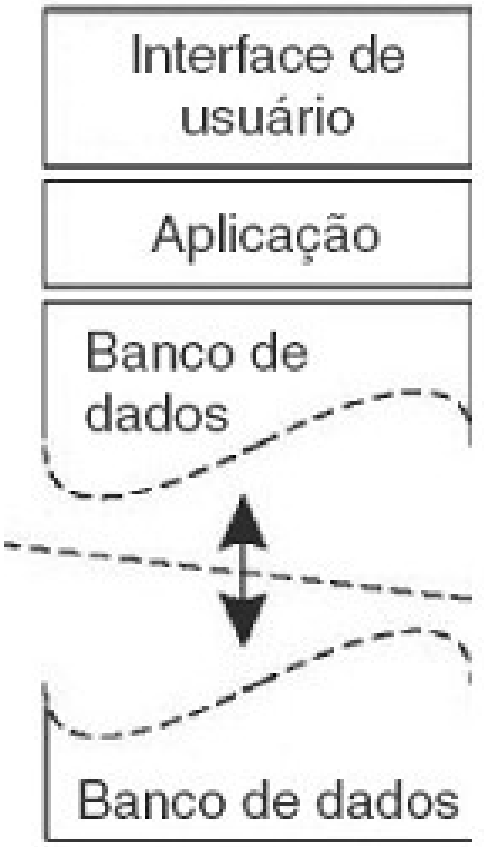
# Arquiteturas Centralizadas (Cont.)

- 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 (Cont.)

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



# Arquiteturas Centralizadas (Cont.)

## Arquiteturas de três divisões físicas

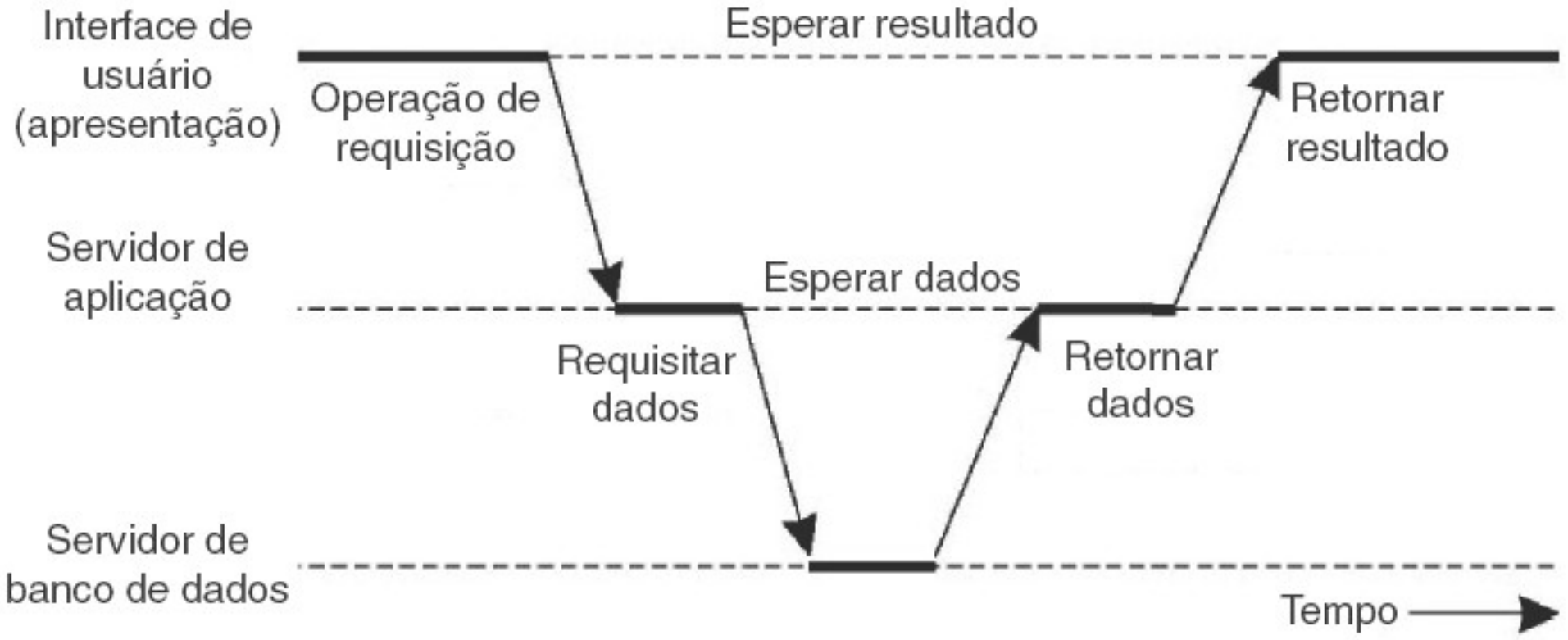



Figura 2.6 Exemplo de um servidor que age como cliente.

# Arquiteturas Descentralizadas

- 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(Cont)

- Sistemas P2P: algumas questões!
  - Como organizar os peers em uma rede de sobreposição (overlay)?
  - Como difundir o conteúdo?
  - Como incentivar os peers a colaborarem?

# Arquiteturas Descentralizadas(Cont)

- 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(Cont)

- 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 >= k
  - LOOKUP(k)

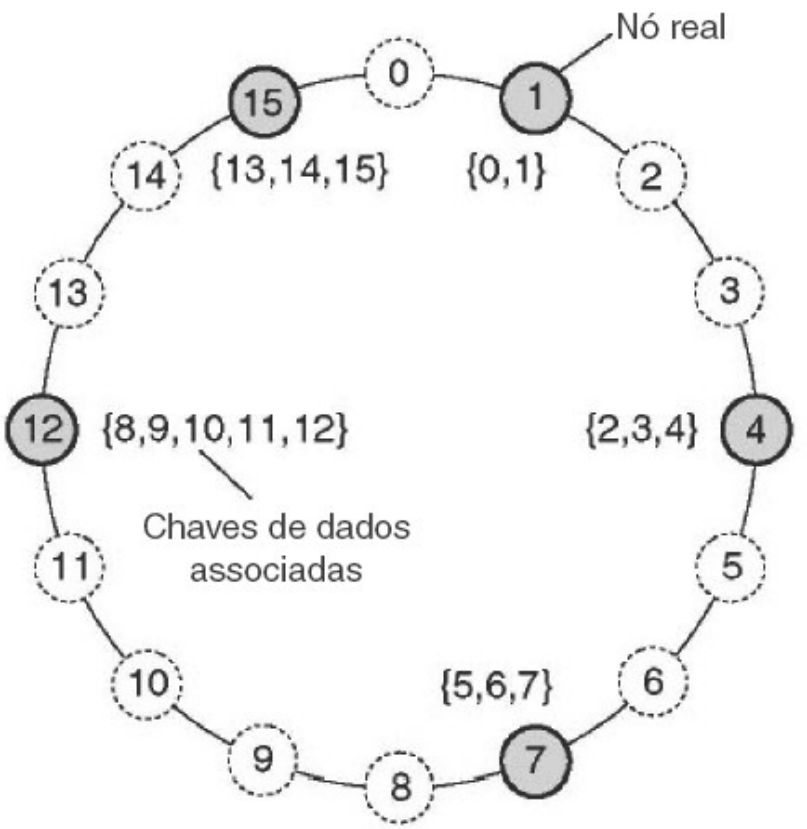
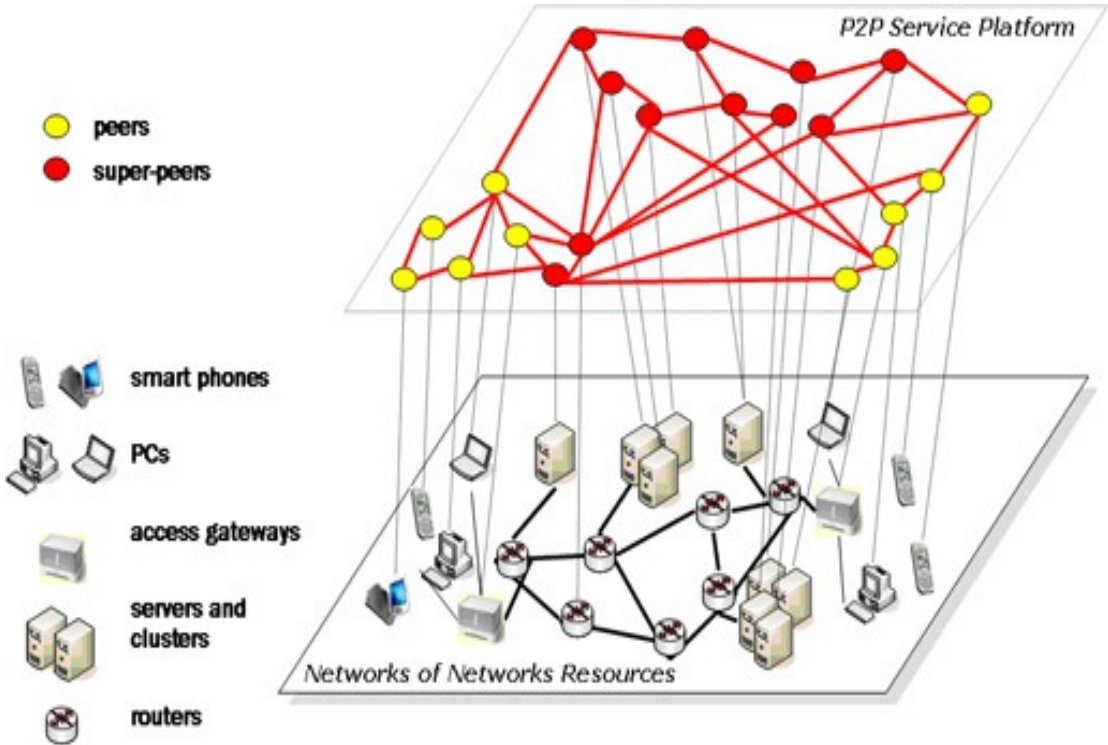



Figura 2.7 Mapeamento de itens de dados para nós em Chord.

# Arquiteturas Descentralizadas(Cont)

- 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(Cont)

- 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(Cont)

- 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(Cont)

- Arquiteturas P2P não-estruturadas

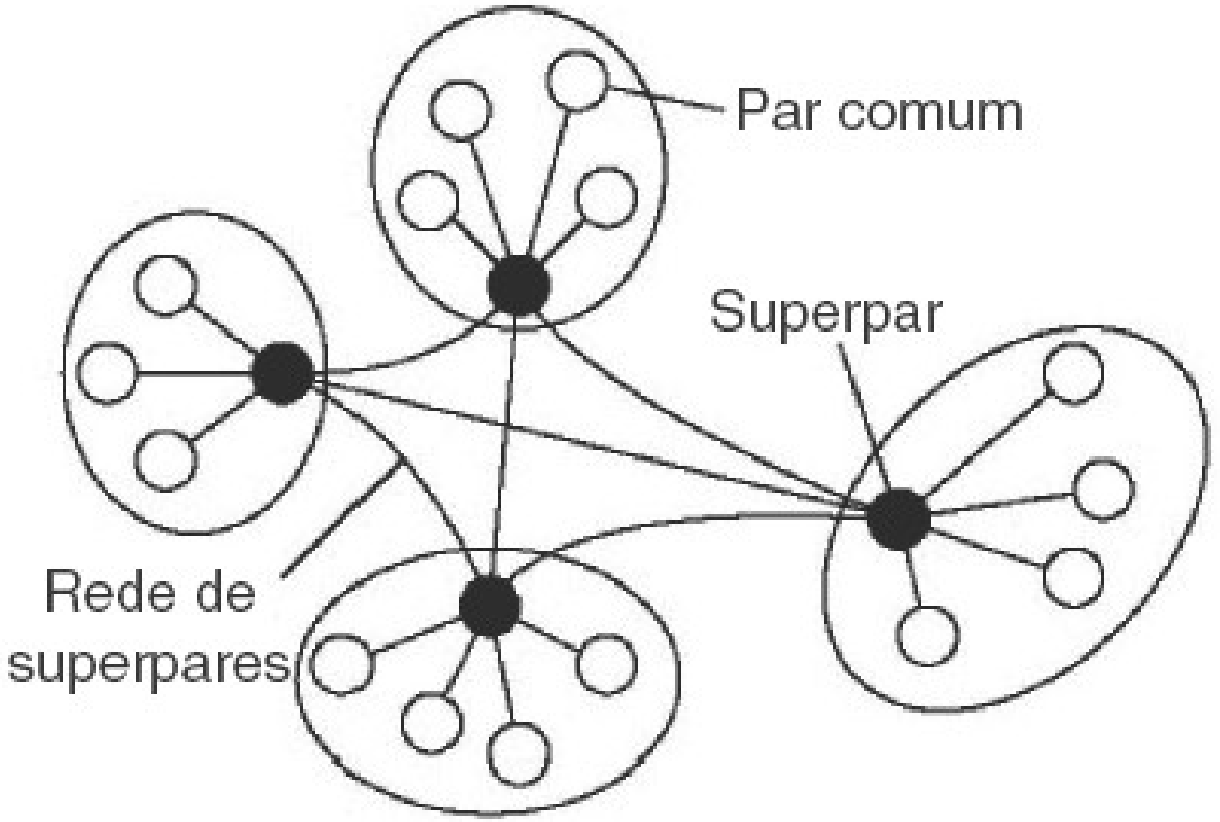



Figura 2.12 Organização hierárquica de nós em uma rede de superpares.

# Arquiteturas Híbridas

- BitTorrent(Cohenm, 2003)