# DISTRIBUTED SYSTEMS ARCHITECTURES

Ian Sommerville, 8° edição – Capítulo 12

Aula de Luiz Eduardo Guarino de Vasconcelos

# Objetivos

- Explicar as vantagens e desvantagens das arquiteturas de sistemas distribuídos
- Descrever diferentes abordagens para o desenvolvimento de arquiteturas cliente-servidor
- Descrever o conceito de CORBA, P2P e SOA como novos modelos de computação distribuída

# Tópicos abordados

- Arquiteturas Multiprocessor
- Arquiteturas Client-server
- Arquitetura de objetos distribuídos
- Computação inter-organizacional

# Sistemas Distribuídos

- Virtualmente, hoje em dia, todos sistemas com base em grandes computadores são sistemas distribuídos
- Processamento de informação é distribuído para vários computadores em vez de ficar confinado a uma única máquina
- Engenharia de sistemas distribuídos possui uma grande importância

# Tipos de sistemas

- Sistemas pessoais – não são distribuídos e são projetados para serem executados em um computador pessoal ou em uma estação de trabalho
- Sistemas embutidos – são executados em um único processador ou em um grupo integrado de processadores
- Sistemas distribuídos – o software de sistema executa em um grupo de processadores fracamente integrados, que cooperam entre si conectados por uma rede

# Características de Sistemas Distribuídos

- Compartilhamento de recursos / Resource sharing
  - Sharing of hardware and software resources.
- Abertura / Openness
  - Use of equipment and software from different vendors.
- Concorrência / Concurrency
  - Concurrent processing to enhance performance.
- Escalabilidade / Scalability
  - Increased throughput by adding new resources.
- Tolerância a falhas / Fault tolerance
  - The ability to continue in operation after a fault has occurred.

# Desvantagens dos Sistemas Distribuídos

- Complexidade / Complexity
  - Typically, distributed systems are more complex than centralised systems.
- Segurança / Security
  - More susceptible to external attack.
- Facilidade de Gerenciamento / Manageability
  - More effort (esforço) required for system management.
- Imprevisibilidade / Unpredictability
  - Unpredictable responses depending on the system organisation and network load.

# Arquiteturas de Sistemas Distribuídos

- Arquiteturas cliente-servidor
  - Conjunto de serviços distribuídos que podem ser solicitados pelos clientes. Servidores e clientes são tratados de maneiras distintas
- Arquiteturas de objetos distribuídos
  - Nenhuma distinção entre clientes e servidores.
  - Qualquer objeto no sistema podem fornecer e utilizar serviços de outros objetos

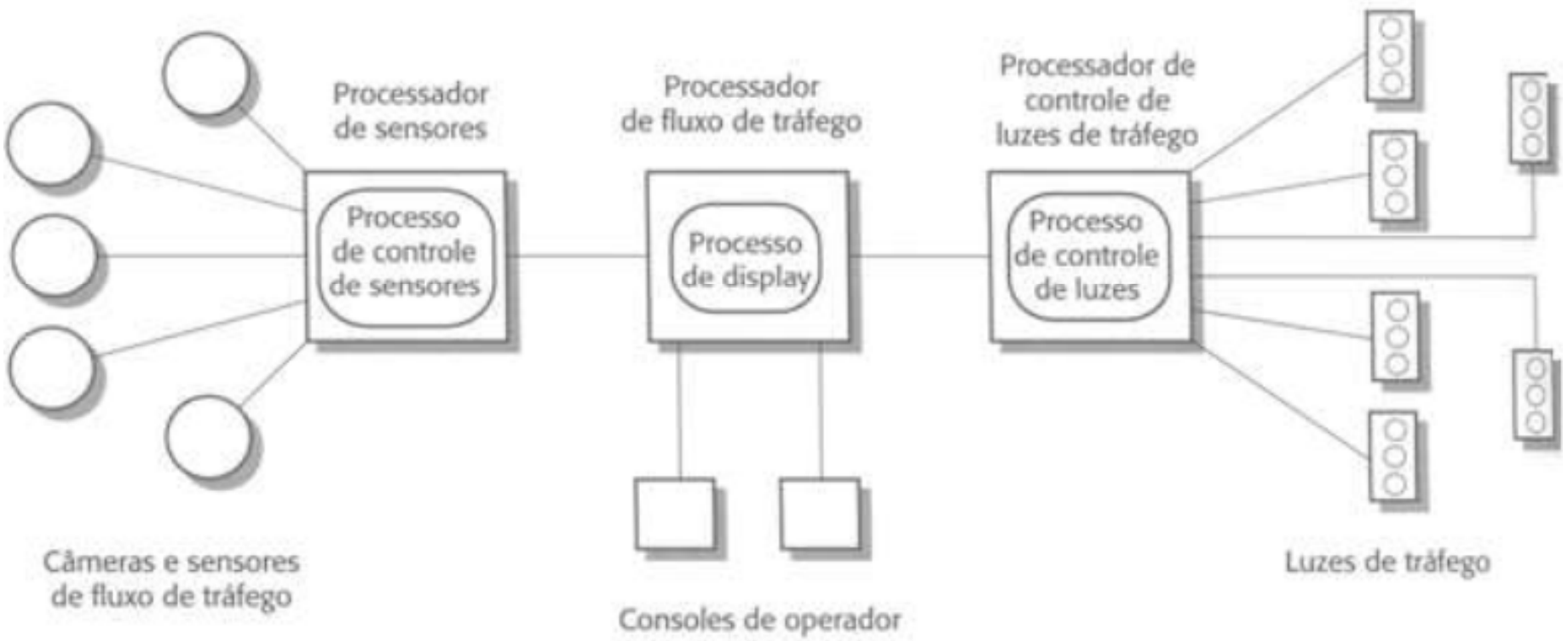
# Middleware

- Software que gerencia e suporta os diferentes componentes de um sistema distribuído. Essencialmente, ele é o intermediário dos diferentes componentes distribuídos do sistema
- Middeware é usualmente um software de propósito geral, adquirido no mercado, em vez de ser escrito especialmente por desenvolvedores de aplicação
- Exemplos
  - Monitores de processamento de transações
  - Conversores de dados
  - Controladores de comunicação

# Arquiteturas Multiprocessor

- Modelo mais simples de sistema distribuído
- Sistema composto de múltiplos processos que podem (mas não precisam) executar em diferentes processadores
- Modelo de arquitetura de muitos grandes sistemas de tempo real
- Distribuição de processos para os processadores pode ser predeterminada ou pode ficar sob o controle de um escalonador

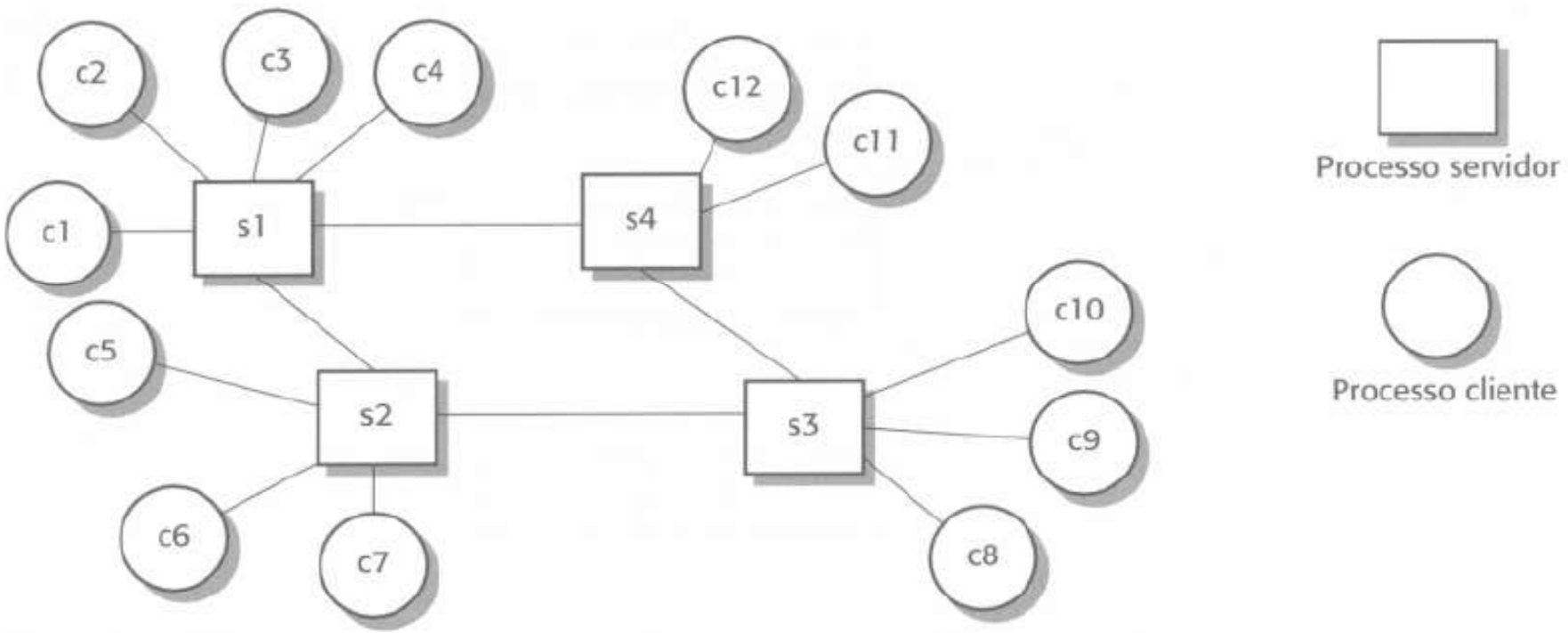
# Sistema de controle de tráfego multiprocessado



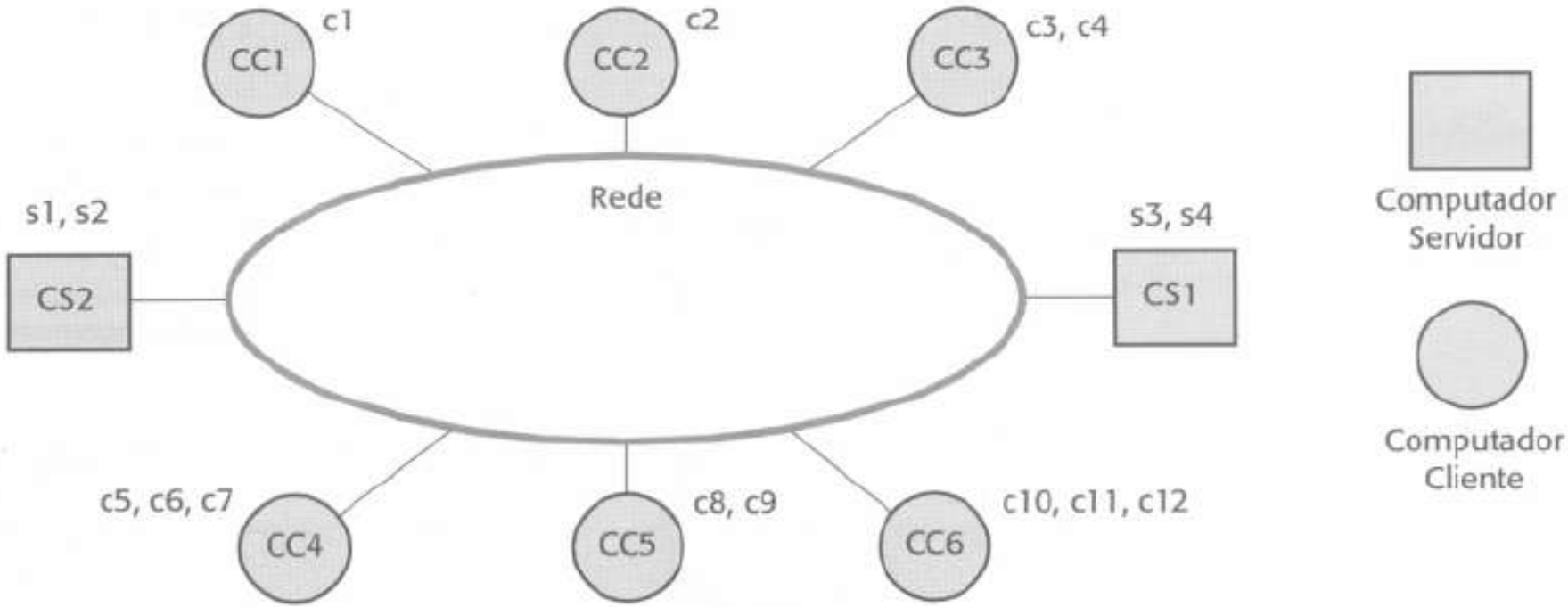
# Arquiteturas Client-server

- A aplicação é modelada como um conjunto de serviços que são fornecidos pelos servidores e um conjunto de clientes que utilizam esses serviços
- Clientes conhecem os servidores disponíveis mas os servidores não precisam conhecer os clientes
- Clientes e servidores são processos lógicos
- O mapeamento de processadores a processos não é necessariamente 1:1

# Um sistema client-server



# Computadores em uma rede Client-Server



# Arquitetura de aplicação em camadas

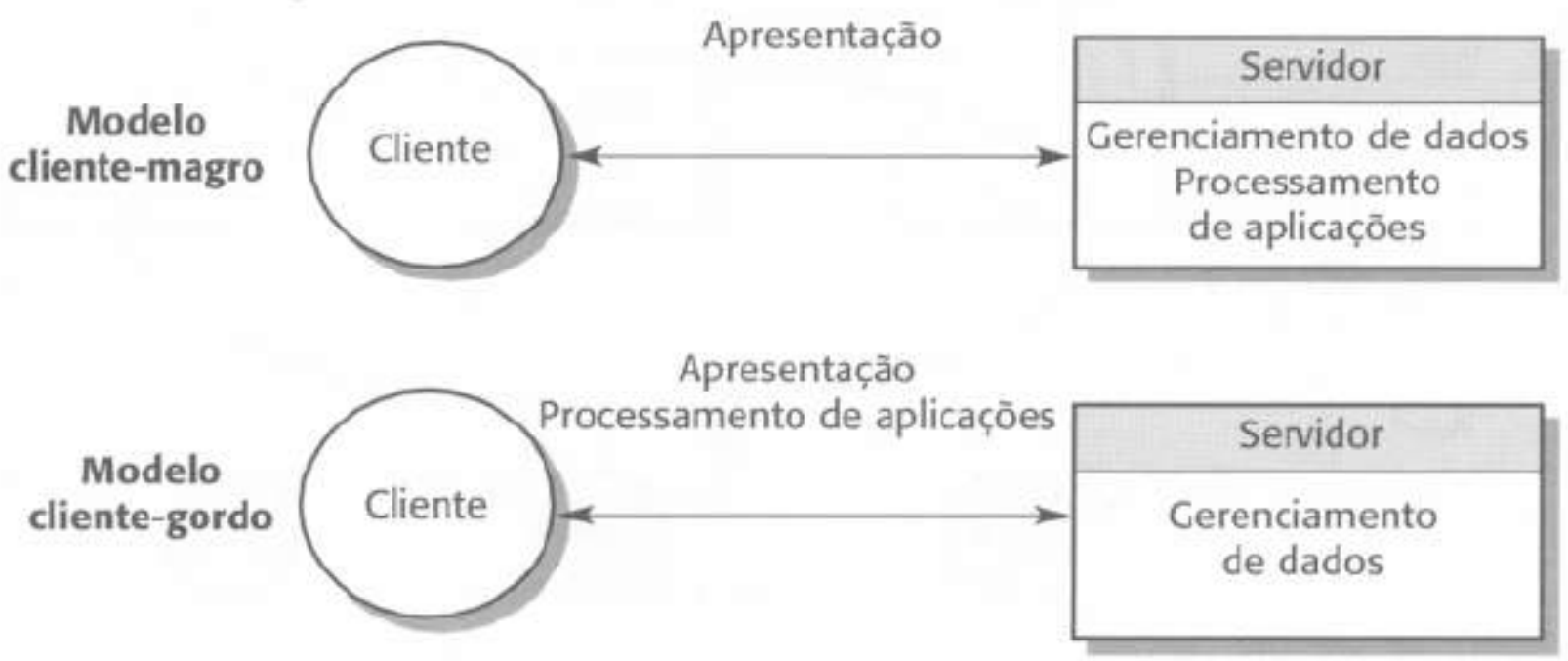
- Camada de apresentação
  - Se ocupa de apresentar informações aos usuários e todas interações com eles
- Camada de processamento de aplicação
  - Se ocupa de implementar a lógica da aplicação (ex. em um sistema bancário, funções como abrir conta, fechar conta etc)
- Camada de gerenciamento de dados
  - Se ocupa de gerenciar o sistema de banco de dados

# Camadas de aplicação

# Cliente magro (thin) e gordo (fat)

- Modelo cliente-magro
  - Todo o processamento da aplicação e gerenciamento de dados é realizado no servidor. O cliente é responsável apenas por executar o software de apresentação
- Modelo cliente-gordo
  - O servidor é responsável somente pelo gerenciamento de dados. O software no cliente implementa a lógica da aplicação e as interações com o usuário do sistema

# Thin and fat clients



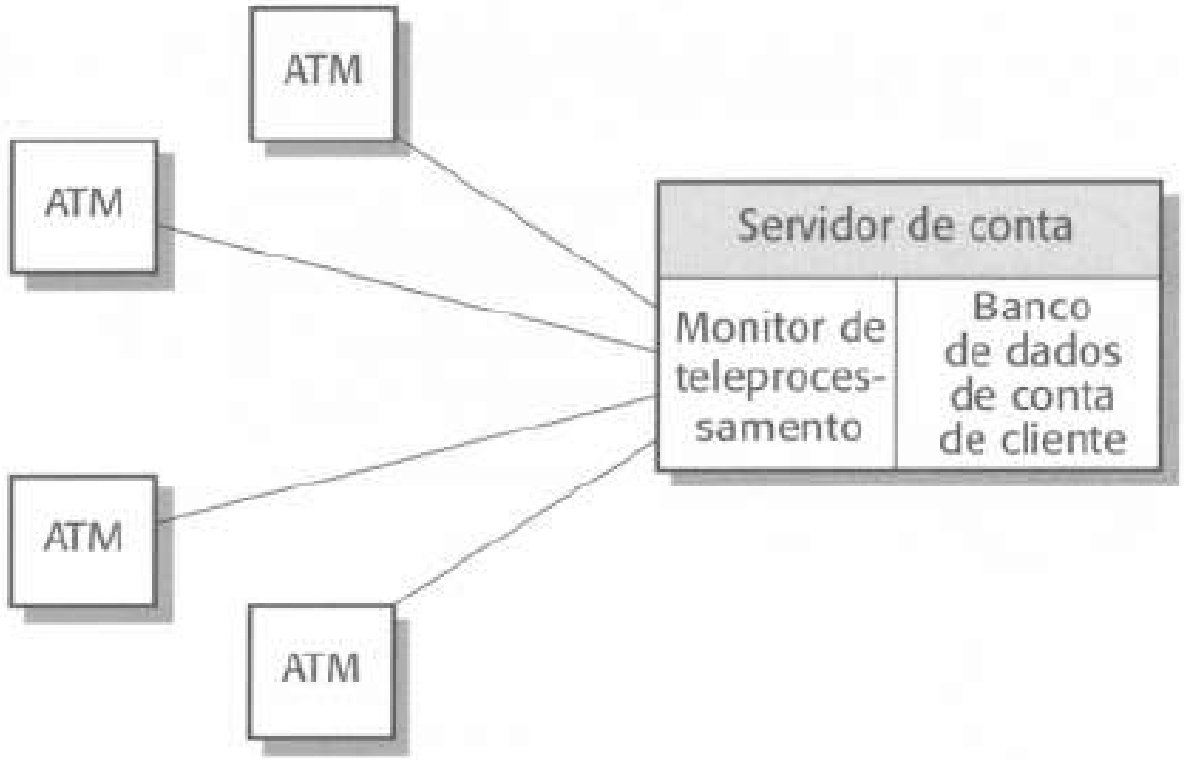
# Thin client model

- Utilizado quando sistemas legados são migrados para arquiteturas cliente servidor
  - O sistema legado atua como um servidor com a interface gráfica implementada nos clientes
- A principal desvantagem é que ele atribui uma grande carga de processamento ao servidor e à rede

# Fat client model

- Distribui o processamento lógico da aplicação e a apresentação para o cliente
- Mais adequado para novos sistemas cliente-servidor onde os clientes possuem uma grande capacidade de processamento
- Mais complexo que o modelo cliente magro especialmente para o gerenciamento. Novas versões da aplicação devem ser instaladas em todos os clientes

# Um sistema ATM client-server

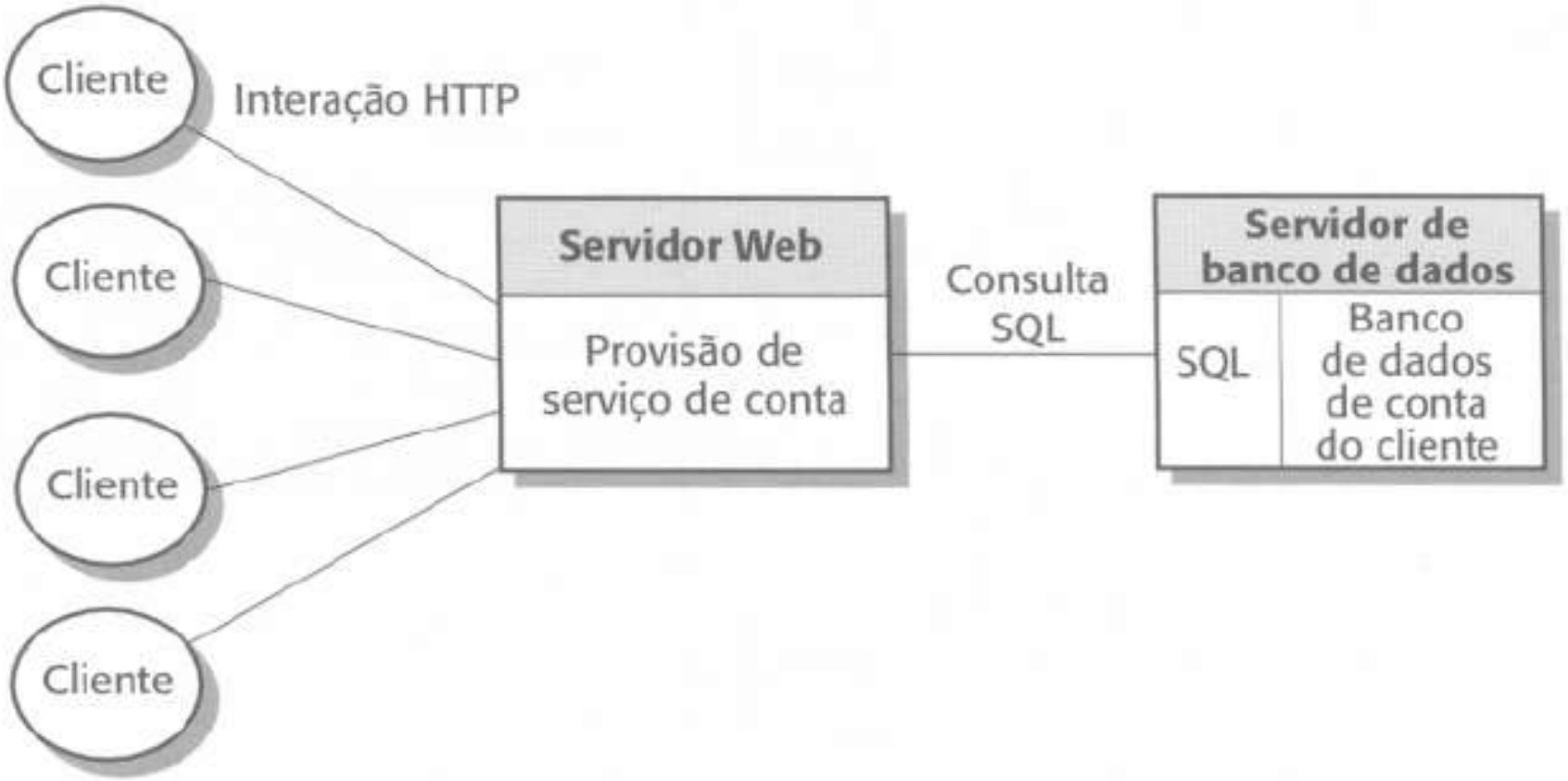


# Three-tier architectures

- Em uma arquitetura de três camadas, cada camada da arquitetura da aplicação pode executar em um processador separado
- Permite melhor performance que a abordagem cliente magro e maior simplicidade de gerenciamento que a abordagem cliente gordo
- Uma arquitetura mais escalável – à medida que a demanda aumenta, novos servidores podem ser adicionados

# Uma arquitetura client-server 3-tier

# Sistema bancário na Internet



# Uso de arquiteturas client-server

| Arquitetura | Aplicações |
|---|---|
| Arquitetura C/S de duas camadas, com clientes-magros | Aplicações de sistemas legados ou de processamento de aplicações e o gerenciamento de dados é impraticável.<br>Aplicações de computação intensiva, como compiladores com pouco ou nenhum gerenciamento de dados.<br>Aplicações de uso intensivo de dados (*browsing* e consultas) com pouco ou nenhum processamento de aplicativos. |
| Arquitetura C/S com clientes-gordos | Aplicações em que o processamento de aplicativos é fornecido por COTS, sistemas comerciais de prateleira (por exemplo, Microsoft Excel), no cliente.<br>Aplicações em que é exigido o processamento de dados de computação intensiva (por exemplo, visualização de dados).<br>Aplicações com funcionalidade relativamente estável para o usuário final, utilizadas em um ambiente de gerenciamento de sistemas bem-estabelecido. |
| Arquitetura C/S com três ou múltiplas camadas | Aplicações em grande escala, com centenas ou milhares de clientes.<br>Aplicações em que os dados e as aplicações são voláteis.<br>Aplicações em que os dados de múltiplas fontes são integrados. |

# Arquiteturas de objetos distribuídos

- Não há distinção entre clientes e servidores
- Cada entidade distribuída é um objeto que fornece serviços a outros objetos e utiliza serviços de outros objetos
- Comunicação entre objetos é realizada por meio de um middleware denominado requisitor de objetos (barramento de software)
- Mais complexo de ser projetado do que sistemas cliente-servidor

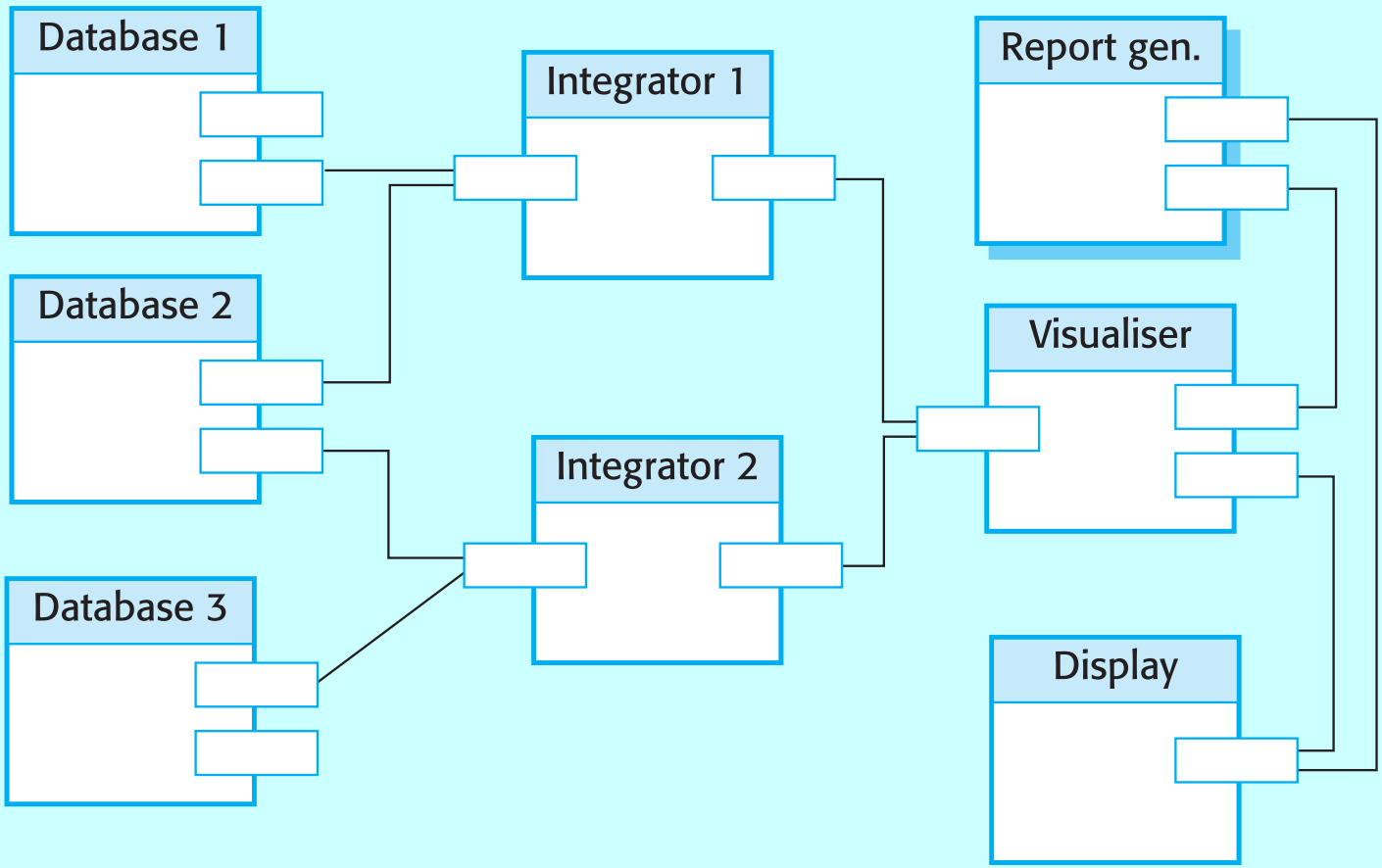
# Uso de arquiteturas de objetos distribuídos

- Cluster (HA, HS, HPC)
- Grid Computing
- Cloud Computing

# Vantagens da arquitetura de objetos distribuídos

- Permite ao projetista do sistema adiar decisões sobre onde ou como os serviços devem ser fornecidos
- É uma arquitetura de sistemas muito aberta que permite que novos recursos sejam adicionados, conforme necessário
- O sistema é flexível e escalável
- É possível reconfigurar o sistema dinamicamente com objetos que migrem pela rede, conforme necessário

# Um sistema de data mining



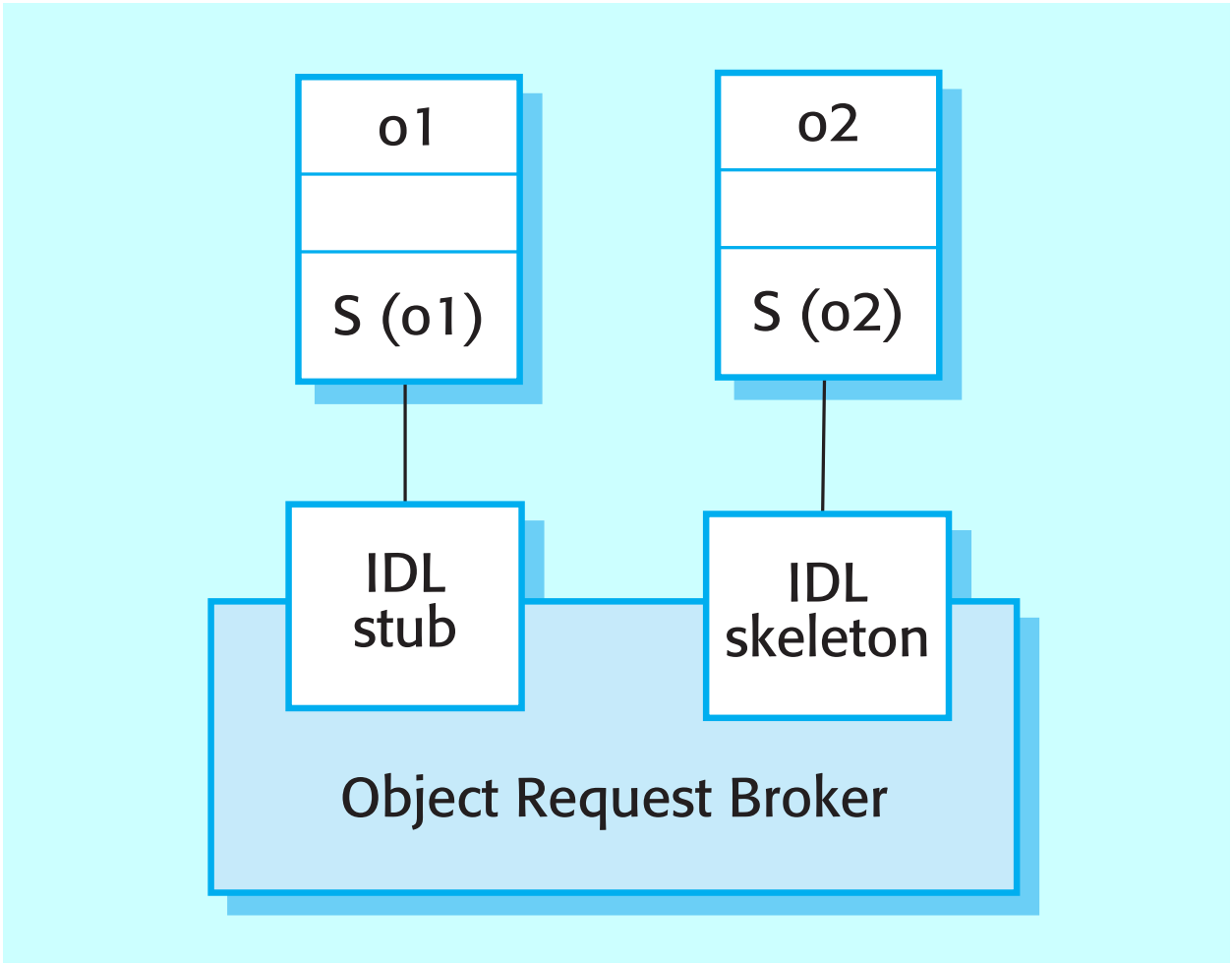
# Sistema de Data mining

- O modelo lógico do sistema não é o de provisão de serviços, em que há serviços distintos de gerenciamento de dados
- Permite que o número de bancos de dados acessados seja aumentado, sem interferir no sistema
- Permite que novos tipos de relacionamentos sejam selecionados pela adição de novos objetos integradores

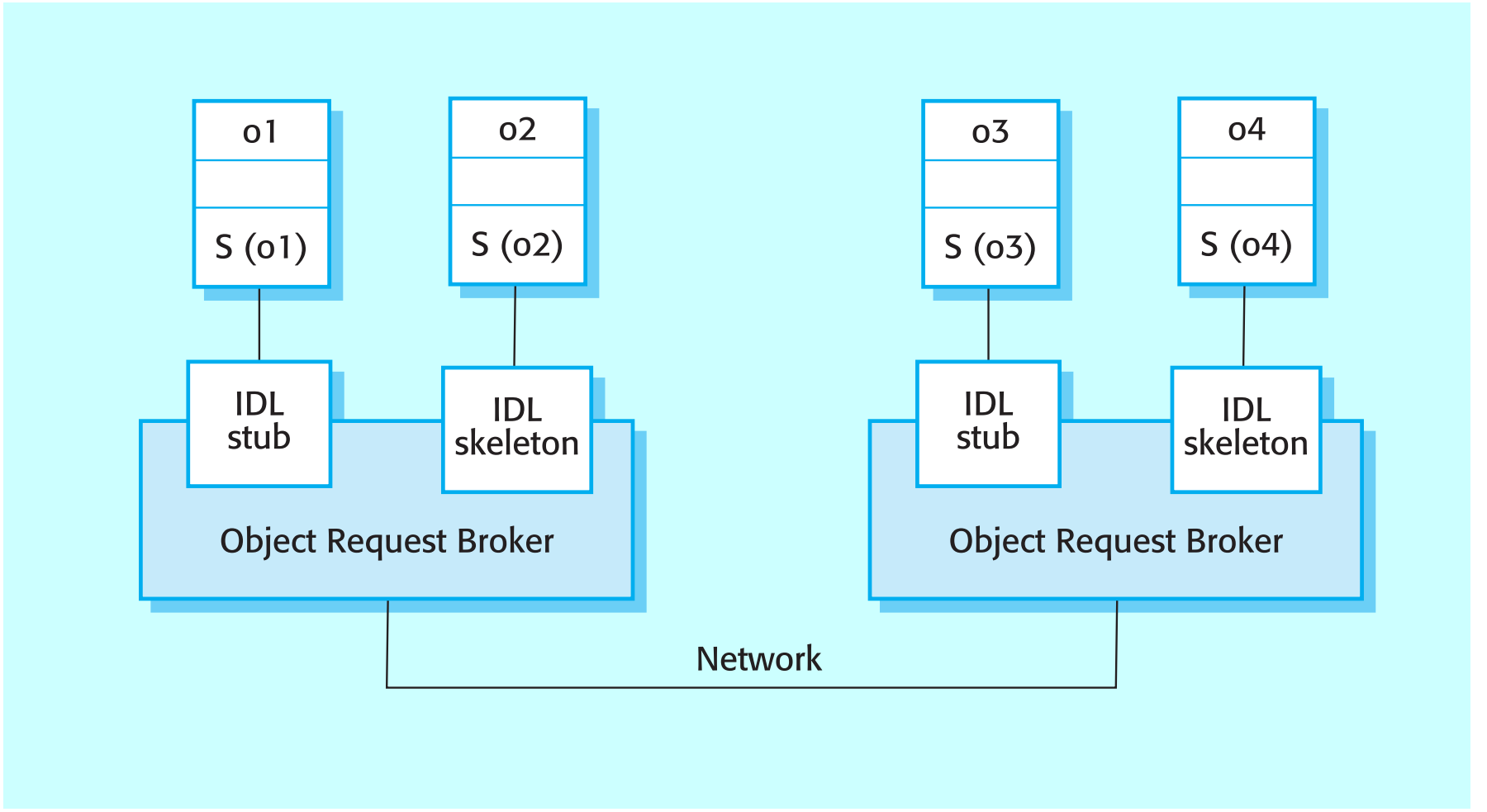
# CORBA

- CORBA é um padrão internacional para um Requisitor de objeto – middleware que gerencia a comunicação entre objetos distribuídos
- Diversas implementações do CORBA estão disponíveis
- DCOM é uma abordagem alternativa para requisitores de objetos desenvolvida pela Microsoft
- CORBA foi definido pelo OMG (Object Management Group)

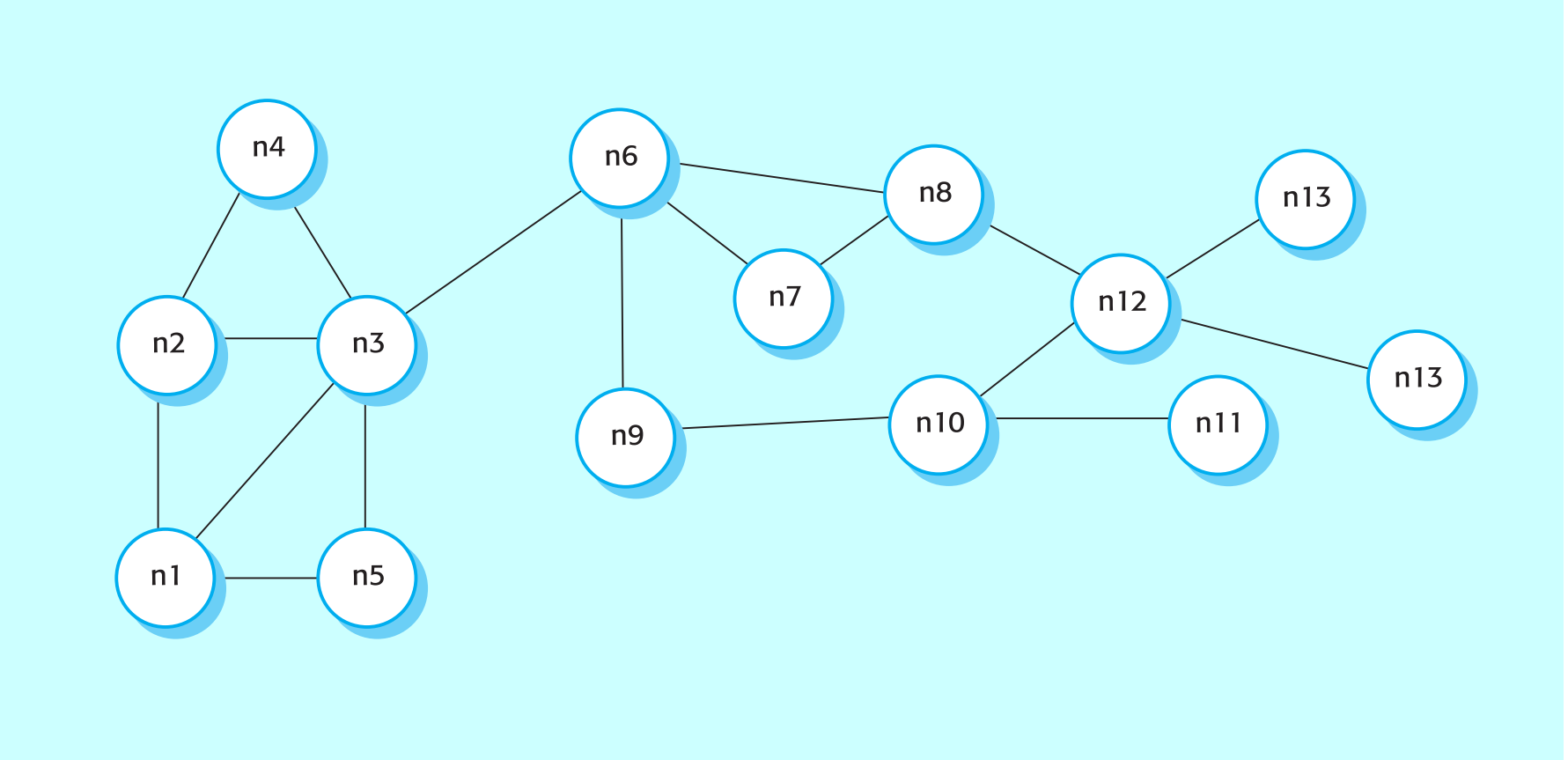
# Comunicação de objetos baseado no ORB



# Inter-ORB communications



# Peer-to-peer architectures (Decentralised)



# Service-oriented architectures (SOA)

- *"Uma ação ou desempenho oferecido de um grupo para um outro. Embora o processo possa estar ligado a um produto físico, o desempenho é essencialmente intangível e não resulta normalmente em propriedade de algum dos fatores de produção".*
- O fornecimento dos serviços é, portanto, independente da aplicação que usa o serviço.
- Um *web service é uma abordagem padronizada para tornar um* componente reusável disponível e acessível através da rede

# Web services

# Vantagens do uso de serviços

- Independência de Provider
- Interoperabilidade.
- Redução do tempo de desenvolvimento (redução do downtime).
- Construção de novos serviços baseados em composition.
- Paga pelo uso de serviços
- Smaller, aplicações mais compactas.
- Aplicações reativas e adaptativas

# Services standards

- Services são baseados em padrões XML
- Key standards
  - SOAP - Simple Object Access Protocol;
  - WSDL - Web Services Description Language;
  - UDDI - Universal Description, Discovery and Integration.

# Services examples

- Hotel
- Aeronaves
- Aluguel de carro
- Amazon, SAP
- Serviços Bancários
- CEP, Clima, SERASA
- etc

# Pontos-chave

- Sistemas distribuídos suportam compartilhamento de recursos, abertura, concorrência, escalabilidade, tolerância a falhas e transparência
- Arquiteturas Client-server envolvem serviços fornecidos por servidores para processos cliente
- UI sempre funciona no cliente e gerenciamento de dados no servidor compartilhado. A funcionalidade da aplicação pode ser feita no cliente ou no servidor.
- Arquitetura de objetos distribuídos não tem distinção entre clientes e servidores

# Pontos-chave

- Sistemas de objetos distribuídos requerem middleware para comunicação entre objetos, adição e remoção.
- CORBA, P2P e SOA são arquiteturas de objetos distribuídos
- Web Services é a tecnologia mais nova nesta área