

Sistemas Distribuídos

Aula 7

Roteiro

- Arquitetura de sistemas
- Arquitetura de sistemas distribuídos
- Arquitetura Cliente/Servidor
- Sistemas de dois níveis



Construindo Sistemas

- Como construir um sistema de informação de grande porte?

- ex. g++, Linux, SIGA, Gmail?

- Modelar

- Projetar

- Implementar

- Testar

➔ **Legal, mas como?**

- Diferentes técnicas/abordagens para cada etapa

- ex. implementar em C++ ou Python?

- cada abordagem tem vantagens/desvantagens

- não há *bala de prata*!



Projetando Sistemas

- Como projetar (*design*) sistemas?
 - como organizar um grande sistema?

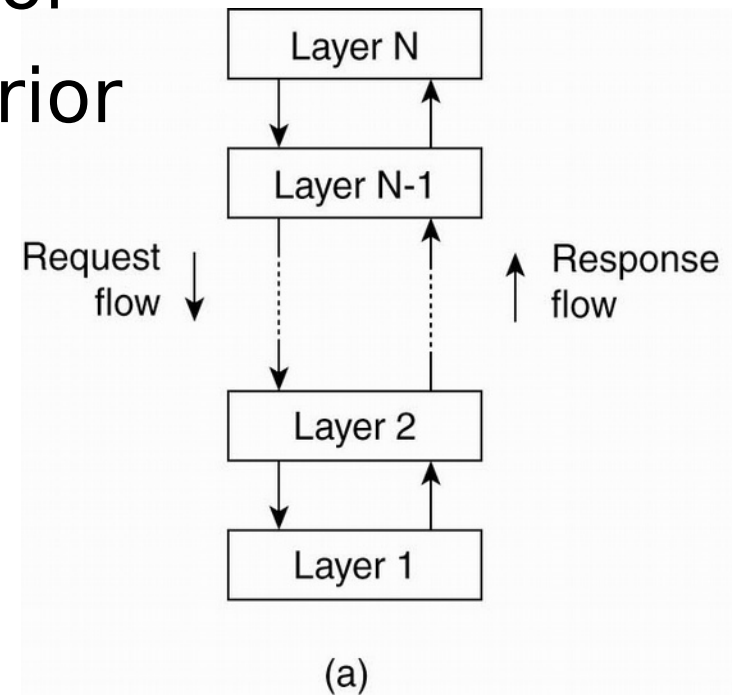
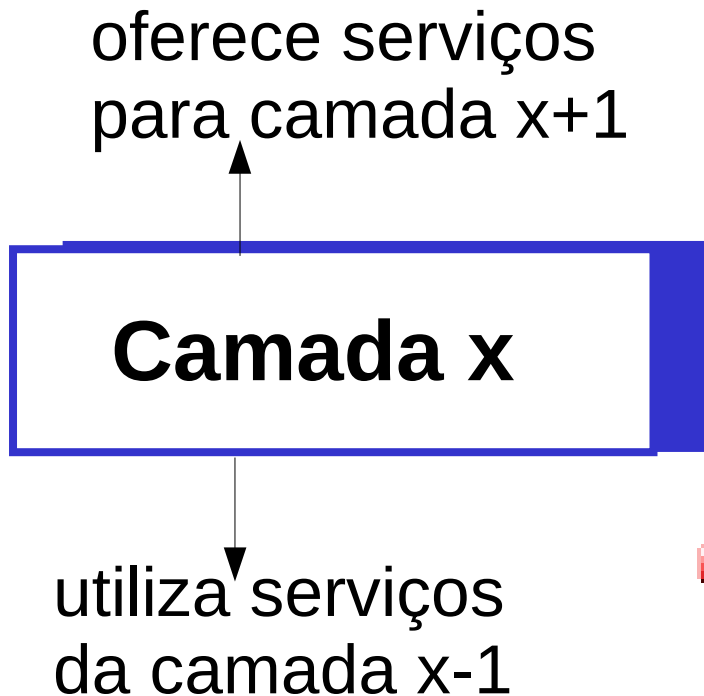
- Adotar uma *arquitetura de sistemas*
 - forma de organizar um sistema
 - dividir em componentes/módulos de software
 - módulos com interfaces bem definidas
- Diferentes arquiteturas desenvolvidas ao longo do tempo, com prática e teoria
 - como dividir, como conectar, como trocar informação → define uma arquitetura de sistema

Arquitetura de Sistemas

- Quatro grandes abordagens
 - baseada em camadas (mais antiga)
 - baseada em objetos
 - baseada em eventos
 - centrada em dados (mais nova)
- Cada qual com vantagens/desvantagens
- Não necessariamente ortogonais
 - grandes sistemas reais misturam as abordagens

Arquitetura em Camadas

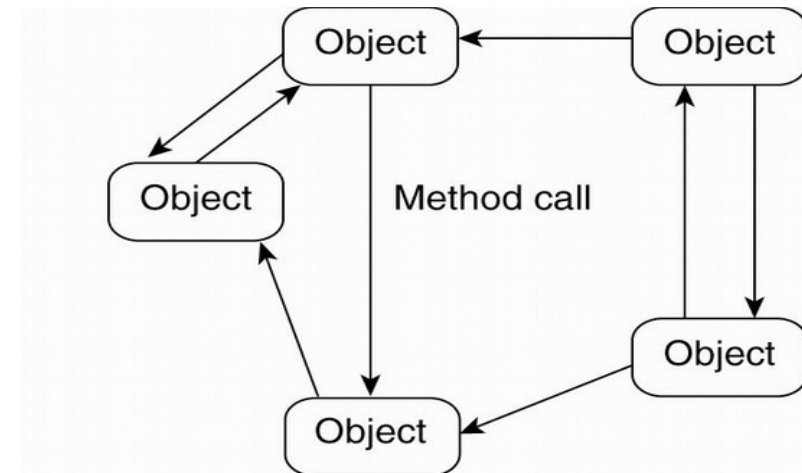
- Componentes organizados “verticalmente”
 - facilita organização e modificação
- Camada implementa um serviço (*função*)
 - utiliza serviço da camada inferior
 - oferece serviço a camada superior



- Mais clássica, muitos exemplos
 - TCP/IP, SO, etc

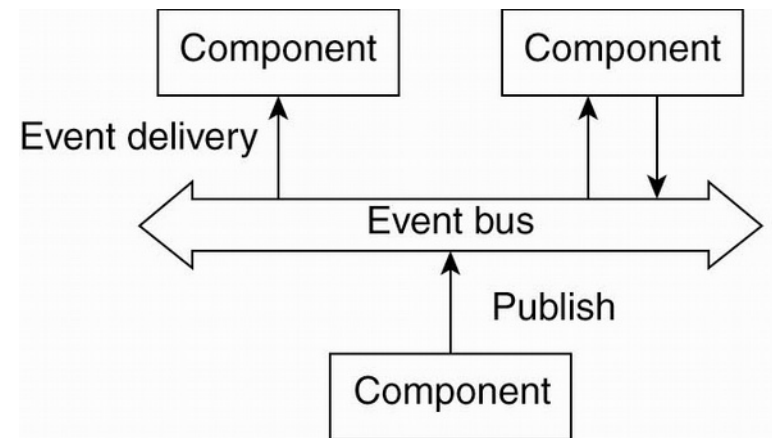
Arquitetura em Objetos

- Usam conceitos de orientação objeto
 - herança, polimorfismo, encapsulamento
- Componentes organizados como um grafo
 - maior flexibilidade, maior eficiência
- Objetos implementam serviços (*funções*)
 - utilizam serviços de outros objetos
 - chamam objetos necessários
- Exemplos
 - alguns SGBDs



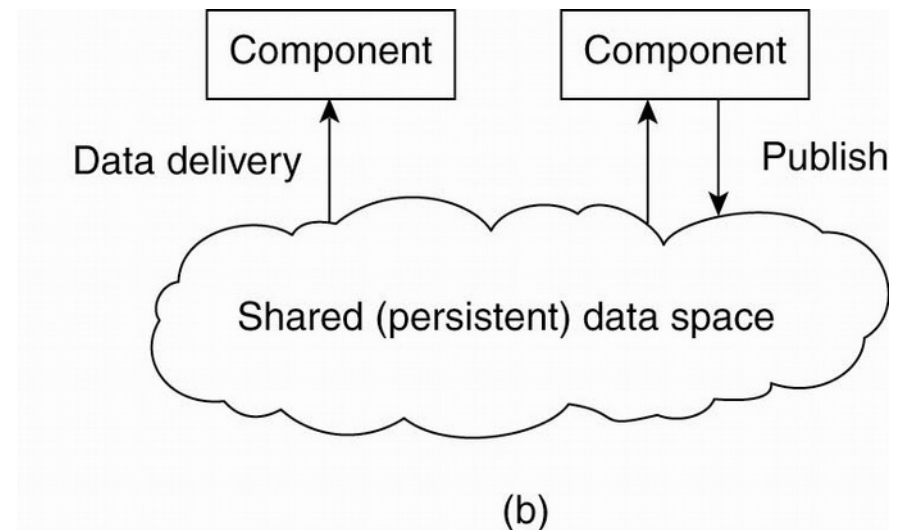
Arquitetura em Eventos

- Componentes organizados em um “barramento” (também chamado de *broker*)
 - maior simplicidade
- Componentes enviam/recebem eventos
 - eventos tem identidade e informação, processados apenas onde necessário
 - utilizam serviço de outros componentes
 - barramento não tem memória (de longo prazo)
- Exemplos
 - Sistemas Publish/Subscribe
 - Engines de jogos



Arquitetura Centrada em Dados

- Componentes organizados em estrela
- Centro da estrela é um grande repositório de dados compartilhado
 - comunicação indireta, através repositório
 - repositório registra (armazena) toda informação
 - repositório é um outro sistema
- Exemplos
 - Google File System
 - Hadoop (Map/Reduce)
 - Content Centric Networks





Sistemas Distribuídos

- Qual é mesmo a definição?

A collection of independent computers that appears to its users as a single coherent system.

- Como projetar sistemas distribuídos?
 - arquiteturas anteriores servem para qualquer sistema (incluindo distribuídos)
- Novamente, adotando uma *arquitetura de sistemas distribuído*
 - forma de organizar o sistema distribuído

Arquitetura de Sistemas Distribuído

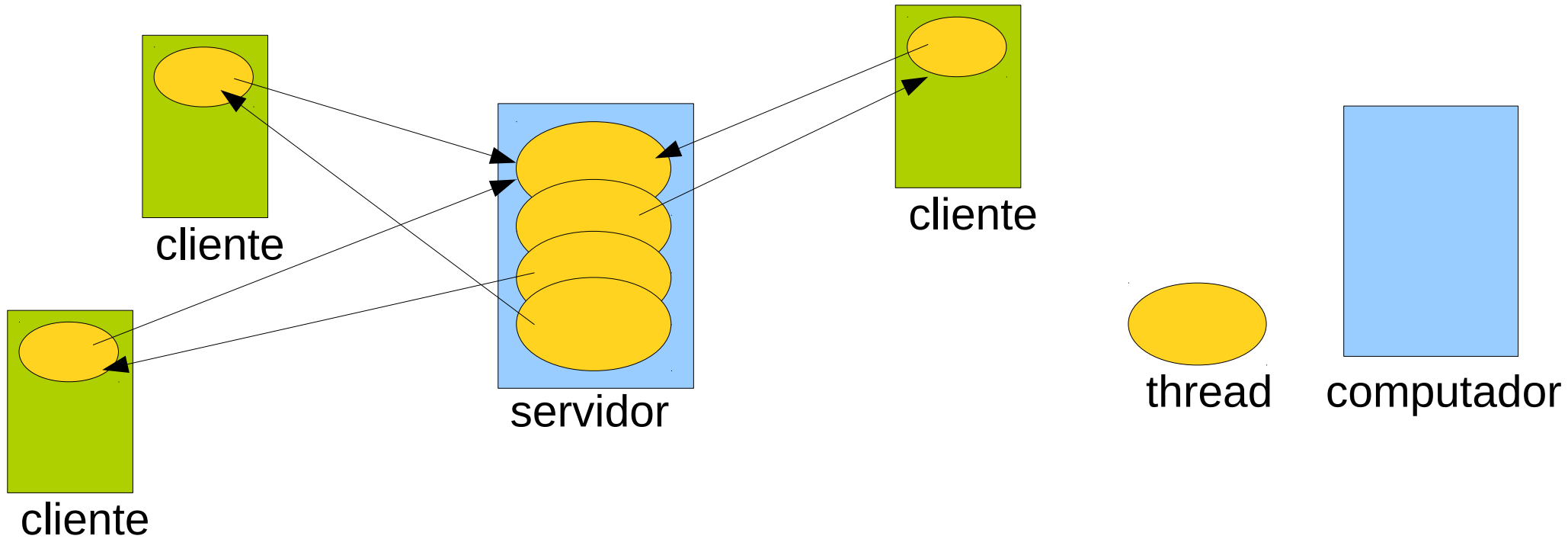
- Duas grandes abordagens
 - cliente/servidor: clássica e mais usada
 - P2P (peer-to-peer): mais nova e mais sofisticada
- Cada qual com vantagens e desvantagens
- Não necessariamente ortogonais
 - sistemas reais misturam as duas abordagens

Definições não são cartesianas!

Cliente/Servidor

- Cada componente (processo ou thread) assume um de dois papéis distintos: *servidor* ou *cliente*
- **Servidor:** oferece um serviço
 - aguarda conexão de um cliente
 - recebe e processa pedido do cliente
 - retorna resultado
- **Cliente:** demanda um serviço
 - se conecta ao servidor
 - envia pedido
 - aguarda resposta
- Papel cliente/servidor são bem distintos

Exemplo Cliente/Servidor



- Web: HTTP e todos os aplicativos sobre HTTP (Web, YouTube, Facebook, Netflix, etc)
- Email
- DNS



Aspectos Centrais a C/S

- Quais são os aspectos centrais da arquitetura cliente/servidor?

- Como encontrar o servidor?

- processo cliente precisa conhecer endereço

- Como organizar pedidos e respostas?

- um ou múltiplos pedidos por conexão?

- Como dimensionar um servidor?

- atender quantos clientes? demanda variável?

- Como lidar com falhas?

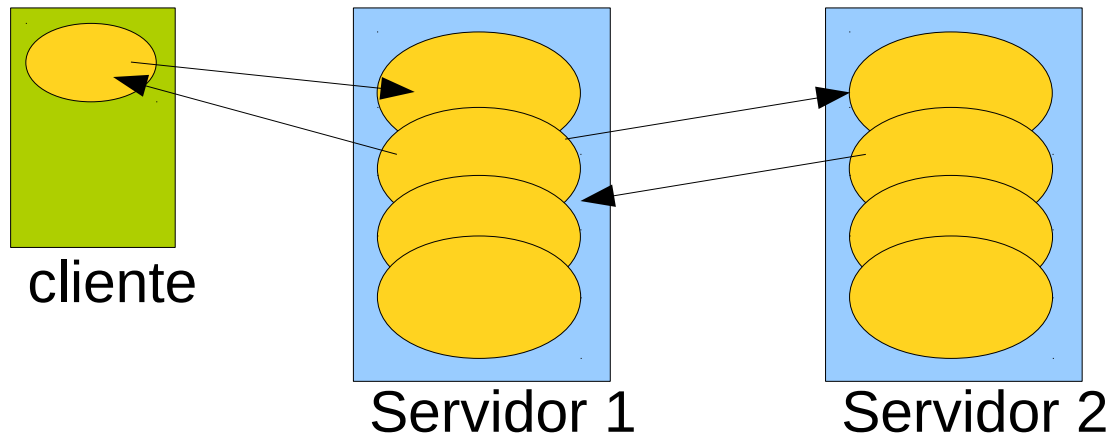
- redundância com múltiplos servidores idênticos?

Iremos discutir esses problemas!

Servidor que é Cliente



- Um servidor pode também fazer papel de cliente?
- *Claro! Servidor é meu, e projeto como quiser*



- Servidor 1 contacta Servidor 2 (se passando como cliente) para exercer sua função
- Sistema de dois níveis (*two-tier system*)
- Ex: servidor web, autenticação, banco de dados