Arquiteturas – Parte I

Vanice Canuto Cunha

Instituto de Computação - IC

Estilos arquiteturais

Ideia básica

- Um estilo é formulado em termos de:
  - componentes (substituíveis) com interfaces bem definidas
  - a maneira como os componentes são conectados entre si
  - os dados trocados entre os componentes
  - como esses componentes e conectores são configurados conjuntamente em um sistema.

Conector

- Um mecanismo que serve para mediar a comunicação, coordenação ou cooperação entre componentes.

Estilos arquitetônicos

- A organização dos sistemas distribuídos é focada, principalmente, sobre nos componentes de software que constituem o sistema.
- Essas arquiteturas de software nos dizem como os diversos componentes de software devem ser organizados e como eles devem interagir.

Importante:

Em sistemas distribuídos é importante separar as aplicações das plataformas subjacentes, fornecendo uma camada **chamada middleware.**

Estilos arquitetônicos

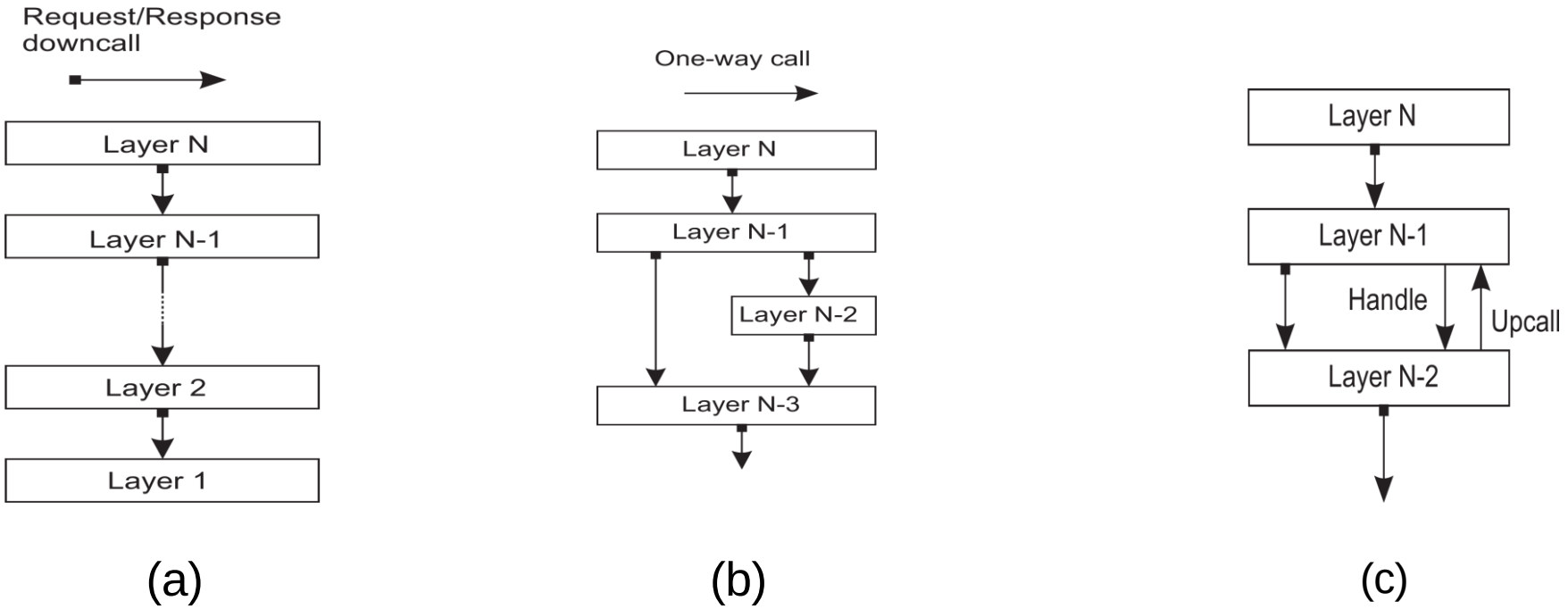
- A instância final de uma arquitetura de software também é chamada de **arquitetura de sistema**.

- **Arquitetura de software** - organização lógica de um sistema distribuído em componentes de software.

Estilos arquitetônicos

- Combinando componentes e conectores, podemos chegar a as seguintes classificações de estilos arquiteturais.
- Arquiteturas em camadas
- Arquiteturas orientadas a serviços
- Arquiteturas de publicação e assinatura

Arquitetura em camadas
Diferentes organizações em camadas



(a) (b) (c)

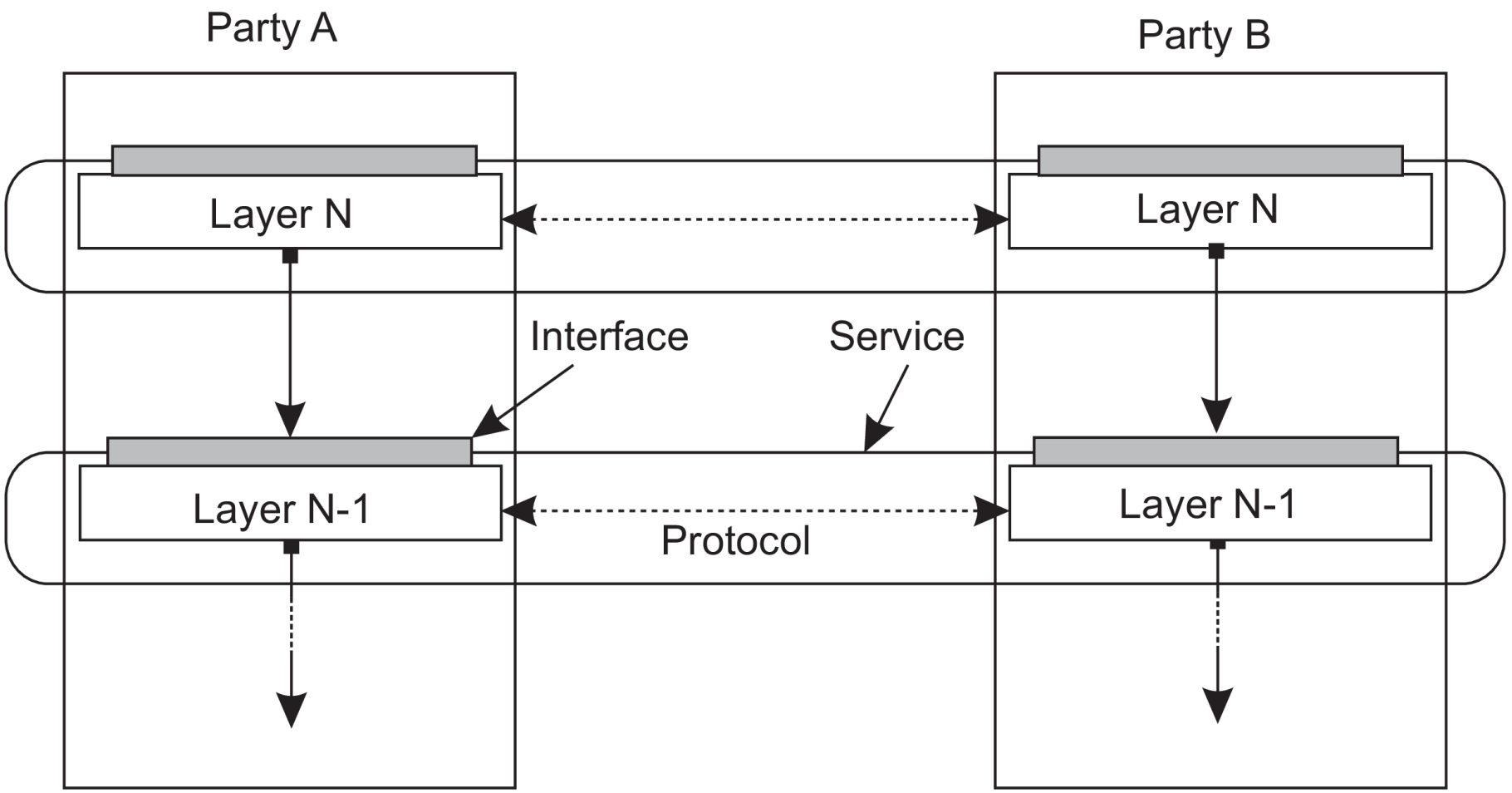
A ideia básica do estilo em camadas é simples: os componentes são organizados em camadas, onde um componente na camada Lj pode fazer uma chamada para baixo para um componente em uma camada inferior Li (com i < j) e geralmente espera uma resposta.

Protocolos de comunicação em camadas

Uma pilha de protocolos de comunicação em camadas, mostrando a diferença entre um serviço, sua interface e o protocolo que ele utiliza:

- **Serviço**: A funcionalidade oferecida pelo protocolo. Por exemplo, uma camada de transporte que oferece uma conexão confiável entre dois sistemas.
- **Interface**: O conjunto de operações e serviços que a camada oferece aos seus usuários. Por exemplo, a interface de uma camada de transporte pode incluir operações como conectar, enviar dados e desconectar.
- **Protocolo**: O conjunto de regras e formatos que define como a comunicação deve ocorrer. No caso da camada de transporte, o protocolo pode definir o formato das mensagens e como garantir que os dados sejam entregues corretamente.

Exemplo: protocolos de comunicação

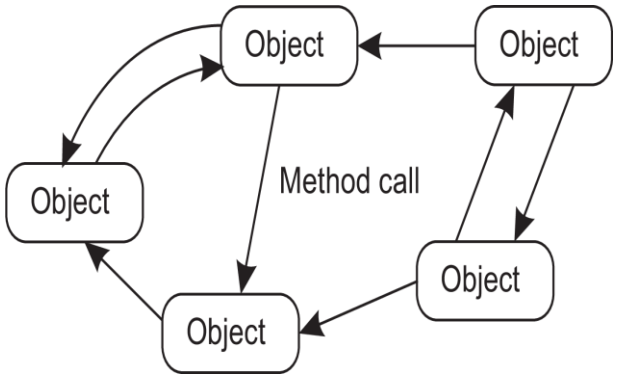


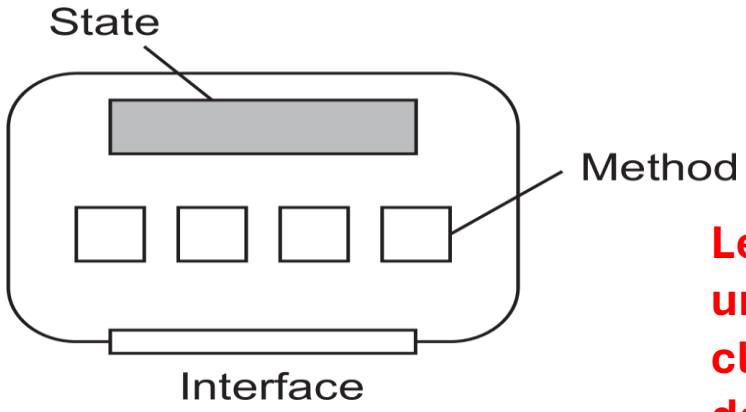
Arquitetura em camadas

Problema:

- Embora o estilo arquitetônico em camadas seja popular, uma das suas principais desvantagens é a forte dependência entre as diferentes camadas.

Estilo Arquitetônico Baseado em Objetos





Lembre-se: Um objeto é uma instância de uma classe que encapsula dados e comportamentos relacionados

Os componentes são objetos, conectados uns aos outros por meio de chamadas de procedimentos. Os objetos podem estar em máquinas diferentes; assim, as chamadas podem ser executadas através de uma rede.

Encapsulamento

Diz-se que os objetos encapsulam dados e oferecem métodos para esses dados sem revelar a implementação interna.

Estilo Arquitetônico de Microsserviços

- Pode-se argumentar que arquiteturas baseadas em objetos formam a base para encapsular serviços em unidades independentes.
- Ao **separar** claramente **vários serviços** de forma que possam operar de maneira independente, estamos pavimentando o caminho para **arquiteturas orientadas a serviços**, geralmente abreviadas como SOAs.

Microsserviços

- Muitos pequenos programas independentes podem ser facilmente compostos para formar programas maiores.
- O essencial é que cada microsserviço execute como um processo (de rede) separado. A implementação de um **microsserviço** pode ser na forma de um objeto remoto, mas isso não é uma exigência.
- Tamanho do microsserviços- ?? - não há um consenso comum sobre qual deve ser o tamanho de tal serviço.

Importante:

Um microsserviço represente verdadeiramente um serviço separado e independente.

Arquiteturas de Publicação e Assinatura (Publish and subscribe)

- Arquitetura na qual as dependências entre processos sejam o mais soltas possível.
- Uma grande classe de sistemas distribuídos adotou uma arquitetura em que há uma forte separação entre processamento e coordenação.
- A ideia é ver um sistema como uma coleção de processos que operam de forma autônoma.
- Neste modelo, a coordenação abrange a comunicação e a cooperação entre processos.

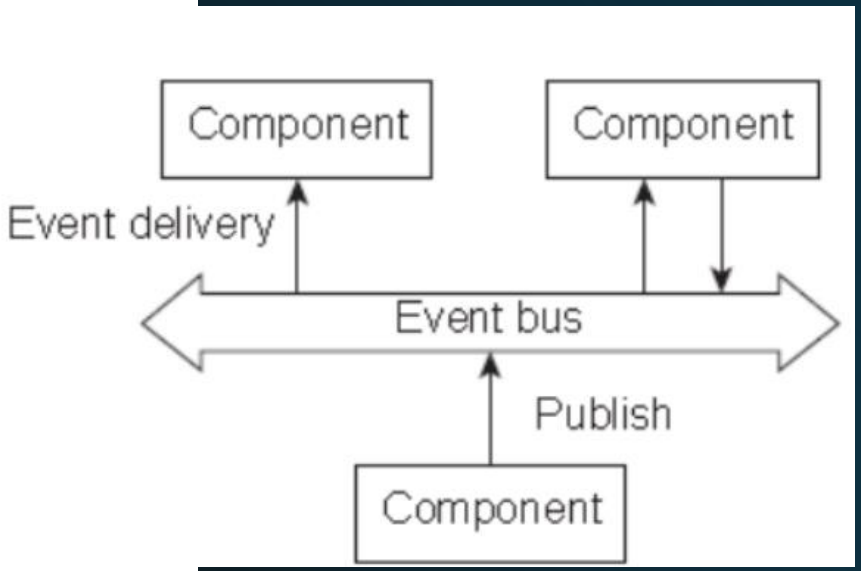
Arquiteturas de Publicação e Assinatura

- A combinação de sistemas referencialmente desacoplados e temporalmente acoplados forma o grupo de modelos para coordenação baseada em eventos.
- Em sistemas desacoplados, os processos não se conhecem explicitamente.
- A única coisa que um processo pode fazer é publicar uma notificação descrevendo a ocorrência de um evento (por exemplo, que deseja coordenar atividades ou que acabou de produzir alguns resultados interessantes).

Arquiteturas de Publicação e Assinatura

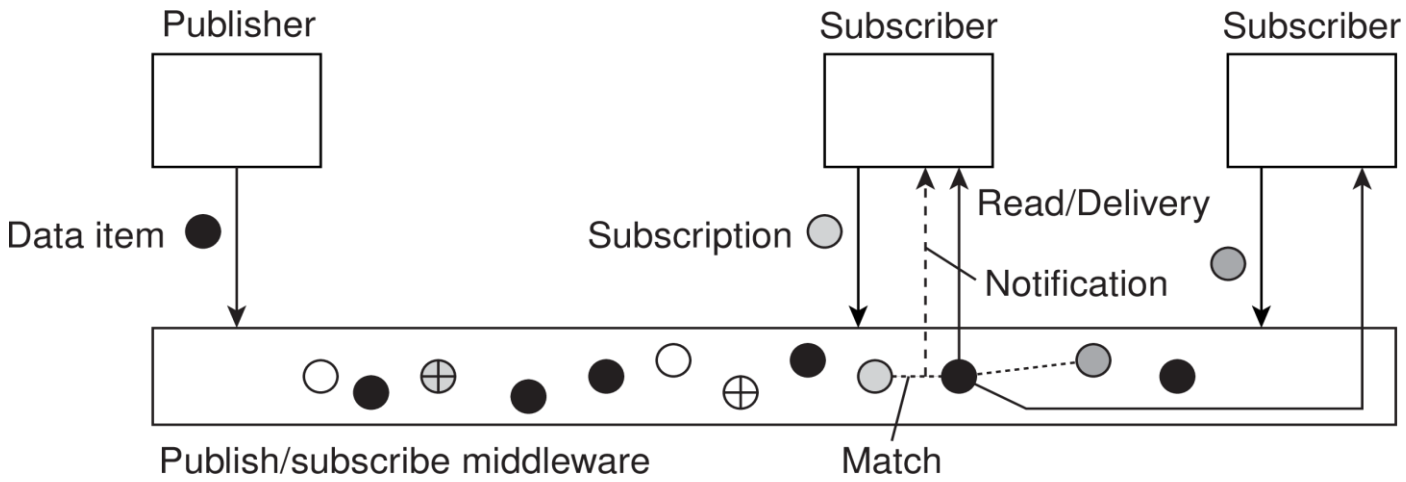
- Em um modelo ideal de coordenação baseada em eventos, uma notificação publicada será entregue exatamente aos processos que se inscreveram nela.
- Padrões comuns incluem o uso de barramentos de eventos (*event buses*) para facilitar a troca de eventos entre componentes.

Exemplos: Kafka, RabbitMQ e outros sistemas de mensagens.



Questão: como combinar eventos?

Assinaturas baseadas em conteúdo podem facilmente enfrentar sérios problemas de escalabilidade (por quê?).



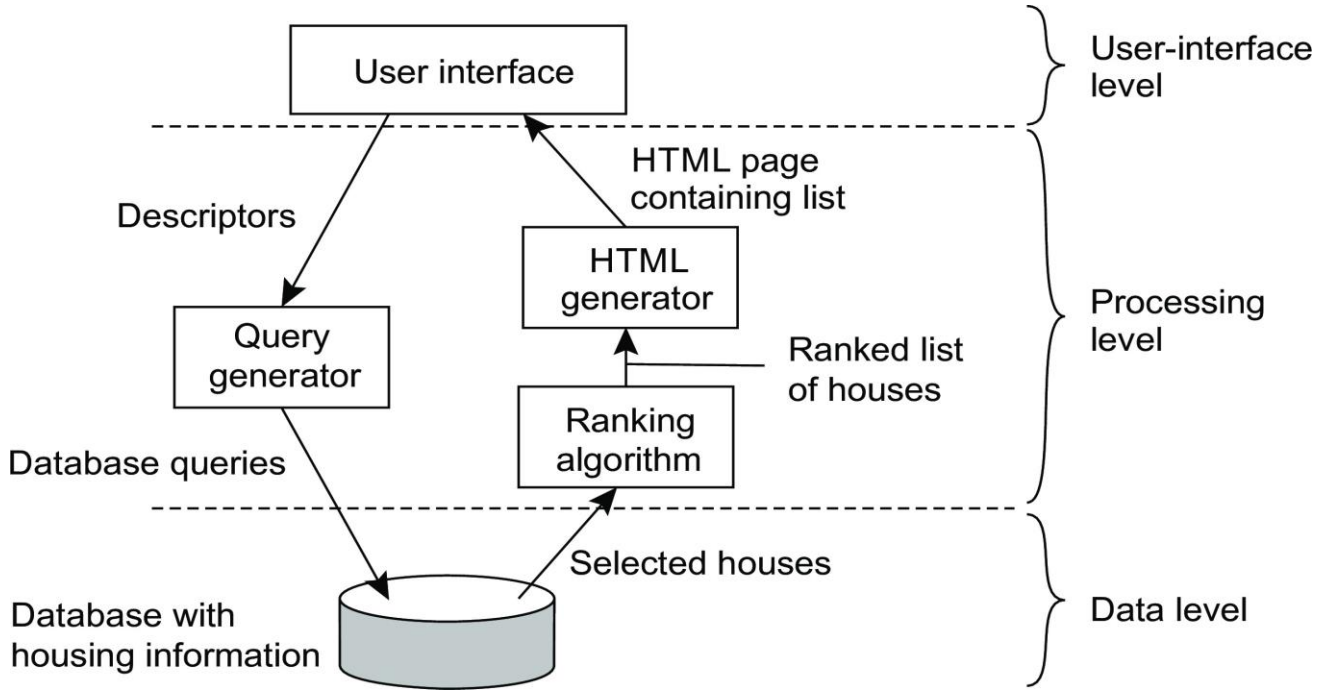
Assinaturas baseadas em conteúdo podem facilmente enfrentar sérios problemas de escalabilidade (por quê?).

Camadas de Aplicação

Vamos agora direcionar nossa atenção para a organização lógica das aplicações em camadas. Considerando que uma grande classe de aplicações distribuídas é voltada para oferecer aos usuários ou aplicações acesso a bancos de dados, muitos especialistas defendem uma distinção entre três níveis lógicos, essencialmente seguindo um estilo arquitetônico em camadas:

- **O nível de interface da aplicação**
- **O nível de processamento**
- **O nível de dados**

Exemplo: um mecanismo de busca simples



Nível de Interface da Aplicação

- **Descrição**: Este nível lida com a interação do usuário. No caso do mecanismo de busca, isso incluiria a interface do usuário onde as consultas de pesquisa são inseridas e os resultados são exibidos.
- **Exemplo**: Uma página web com uma caixa de pesquisa onde o usuário digita termos e clica em "Buscar".

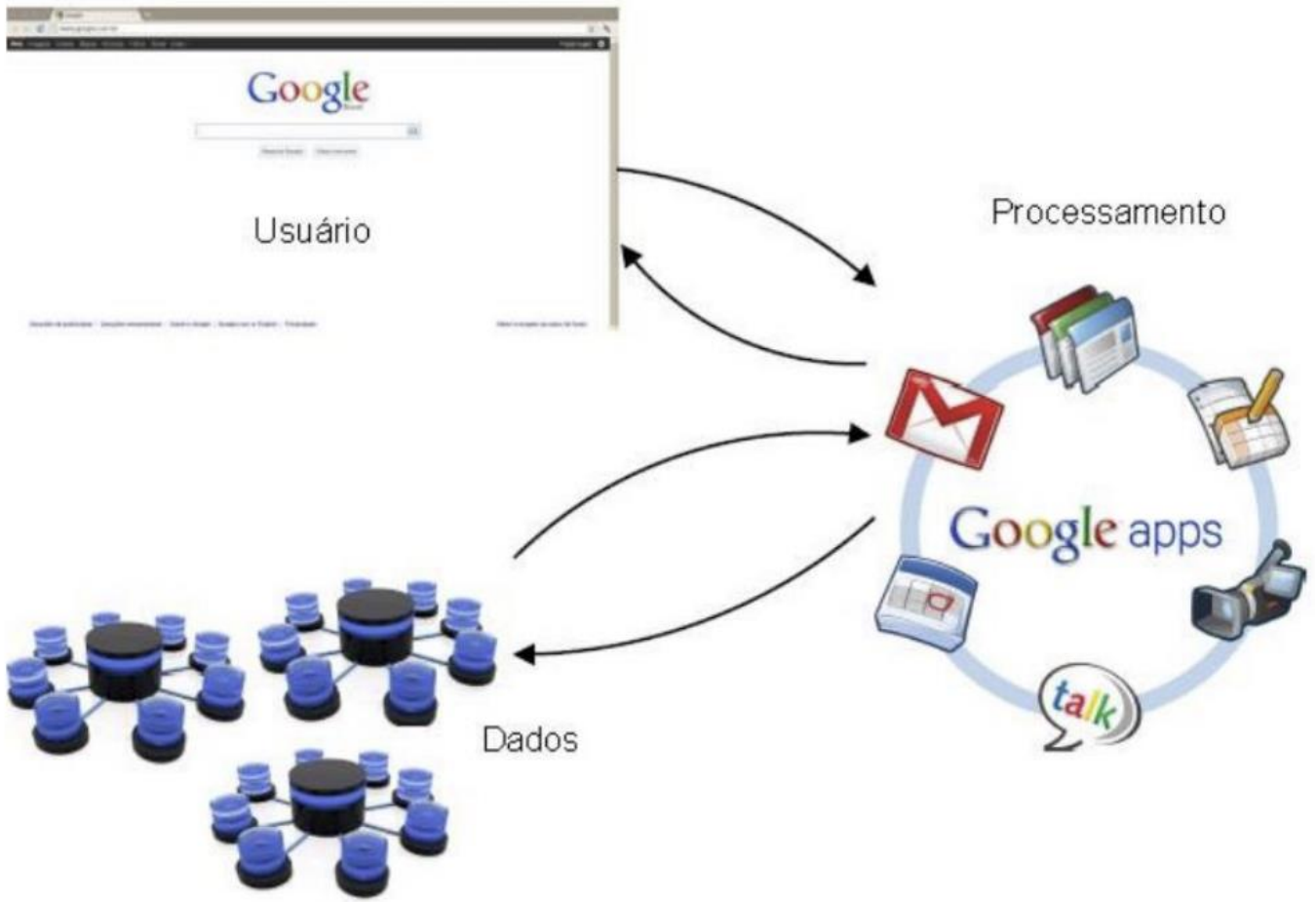
Nível de Processamento

- Descrição: Este nível é responsável pela lógica de processamento das solicitações. Ele recebe a consulta de pesquisa do nível de interface e executa a lógica necessária para encontrar e organizar os resultados.
- Exemplo: O código do servidor que processa a consulta, realiza a busca no banco de dados e classifica os resultados.

Nível de Dados

- Descrição: Este nível gerencia o armazenamento e a recuperação de dados. No mecanismo de busca, ele é responsável por armazenar os documentos e índices que serão pesquisados.
- Exemplo: Um banco de dados ou um índice invertido que armazena o conteúdo dos documentos e permite uma busca rápida.

Exemplo: Google



Arquiteturas de Sistemas

Vamos agora analisar como muitos sistemas distribuídos são realmente organizados, considerando onde os componentes de software são colocados.

- Arquiteturas Centralizadas - Cliente-Servidor
- Arquiteturas Descentralizadas
- Arquiteturas de Sistemas Híbridos

Arquiteturas de Sistemas Centralizados

Modelo Básico Cliente-Servidor

- **Características:**
- Existem processos que oferecem serviços (servidores).
- Existem processos que utilizam serviços (clientes).
- Clientes e servidores podem estar em máquinas diferentes.
- Clientes seguem o modelo de solicitação/resposta ao utilizar serviços.

Arquiteturas de Sistemas Centralizados

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

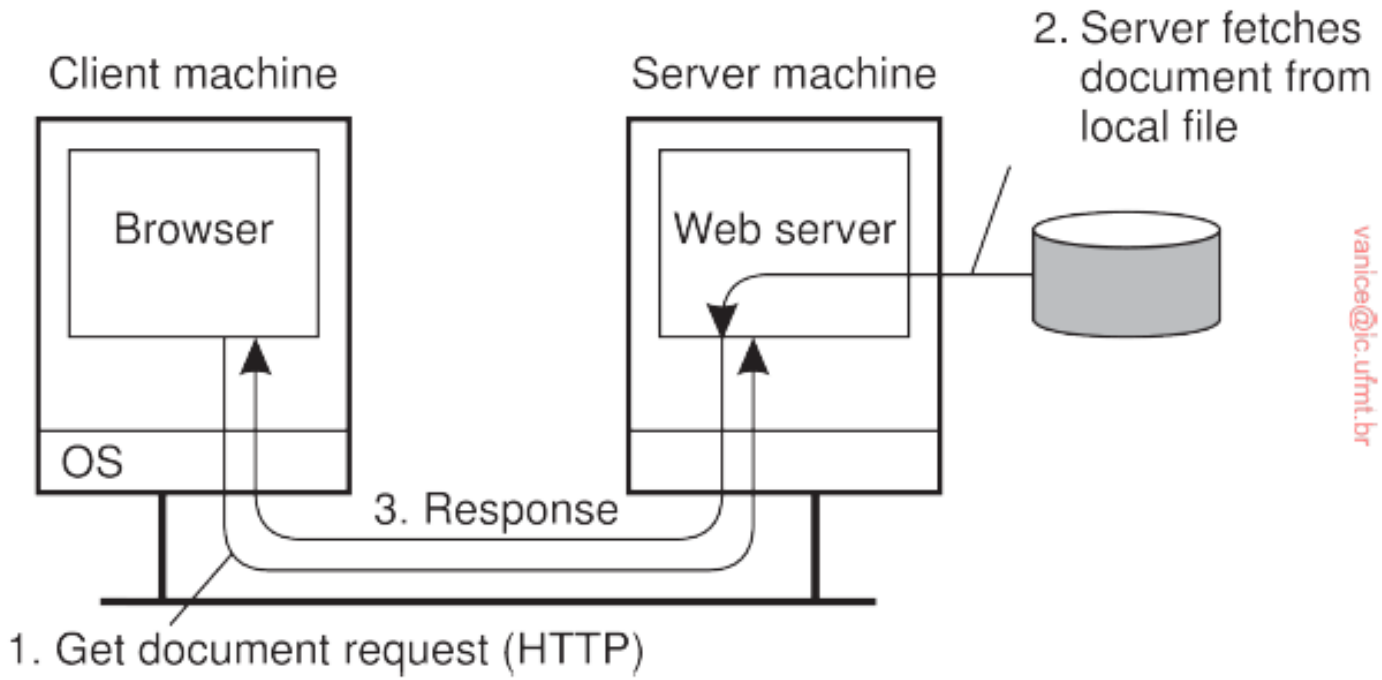
Configuração de Duas Camadas

- Descrição: O sistema é dividido em cliente e servidor (C/S), onde o cliente **faz solicitações** e o servidor **fornece serviços**.

Exemplos:

- Cliente/Servidor de Banco de Dados: Um aplicativo de banco de dados que opera em um cliente que envia consultas a um servidor de banco de dados. O cliente (**software de frontend**) solicita dados e o servidor (**software de backend**) fornece e processa esses dados.
- Aplicação de E-mail: O cliente de e-mail (como Outlook) se comunica com um servidor de e-mail (como Exchange) para enviar e receber mensagens.

Exemplo: Arquitetura cliente/servidor



Comunicação Cliente-Servidor

Através de um protocolo simples sem conexão:

- Utilizado quando a rede é razoavelmente confiável, como a maioria das LANs;
- Vantagem: eficiência;
- Desvantagem: dificuldades de tornar a aplicação resistente a falhas de transmissão;
- Se o cliente não recebe resposta, ele não tem como detectar se a mensagem original foi perdida ou se isto ocorreu com a resposta;
- Para requisições como ”retorne o saldo da minha conta” isto não gera problemas mas para mensagens do tipo ”transfira 10.000,00 da minha conta” o reenvio da requisição seria desastroso;

Através de protocolo confiável orientado a conexão:

- Desvantagem: desempenho relativamente baixo;
- Funciona muito bem em WANs;
- O cliente pode solicitar uma nova requisição;
- Exemplo: praticamente todas as aplicações na Internet utilizam conexões TCP;

Arquiteturas de Sistemas Centralizados Multi-camadas

A distinção entre os três níveis lógicos sugere várias possibilidades para a distribuição física de uma aplicação C/S;

A organização mais simples requer apenas dois tipos de máquinas:

1 - Cliente: que executa os programas que compreendem a interface com o usuário;

2 - Servidor: que implementa as camadas de processamento e de dados. Esta organização é conhecida como Cliente/Servidor em duas camadas; No entanto, diversas outras arquiteturas podem ser utilizadas

Organizações Cliente-Servidor em uma Arquitetura de Duas Camadas.