

### Modelos Físicos

- Modelo físico básico.  
Conjunto de computadores interligados por uma rede trocando mensagens entre si.
- Sistemas distribuídos primitivos – Década de 70.  
10 a 100 computadores ligados por uma rede.
- Sistemas distribuídos adaptados para a internet – Década de 90.  
Sistemas projetados para serem conectados à internet.
- Sistemas distribuídos contemporâneos
  - Computação móvel  
Possibilidade de realizar computação em movimento.
  - Computação ubíqua  
Elementos de computação onipresente em todos os ambientes.
  - Computação em nuvem (clusters)  
Possibilidade de armazenar tudo na nuvem.
  - Aumento da heterogeneidade  
Hardware, aplicações entre outros interagindo.
- Sistemas Distribuídos de Sistemas.

Sistema composto de diversos subsistemas.

**Ex:** Sistema de monitoramento ambiental. Diversos sistemas atuando em conjunto para a gestão ambiental.

### Modelos de Arquitetura – Elementos arquitetônicos

Componentes que iniciam e recebem comunicação.

- Elementos de comunicação
  - Processos;  
Em um nível mais baixo pode-se ter processos enviando e recebendo mensagens.
  - Objetos, componentes e web services.  
Em alto nível, objetos enviam e recebem invocações de métodos. Componentes sendo objetos mais complexos.
  - Web services que são elementos em execução fazendo invocação de serviços web.
- Paradigmas de comunicação
  - Comunicação entre processos – sockets, multicast.

- Pode-se construir sistemas distribuídos com sockets ou multicast.
- Invocação remota  
Usada em implementações de sistemas de objetos distribuídos. Usam-se mecanismos para fazer invocação de um método e obter-se resposta.
    - Protocolo de requisição-resposta. Ex: HTTP.  
Na web. Envia uma requisição e recebe uma resposta.
    - Chamada de procedimento remoto (RPC).  
Possibilidade de fazer a invocação de uma função implementada em um servidor remoto e receber apenas o resultado.
    - Invocação de método remoto (RMI).  
Objetos fazem solicitação de função implementadas em outros objetos remotamente.
  - Comunicação indireta  
A comunicação não requisita ao servidor saber quem ele é e qual sua porta. Nesse tipo de comunicação o cliente não precisa saber quem exatamente é o destinatário da mensagem, uma camada de abstração recebe e entrega ao destinatário correto.

### Comunicação Indireta

- Comunicação em grupo;  
A mensagem é enviada para um grupo.
- Sistemas publicar-assinar (Um para muitos);  
A ideia é que um publique e quem assina recebe. Ex: Site que notifica coisas de acordo com assinatura. Um publica e vários recebem.
- Filas de mensagem (Um para um);  
A mensagem é enviada para uma fila que intermedia a entrega de mensagens.
- Espaços de tupla (persistente).  
Ex: Javaspaces.

As mensagens são guardadas e preservadas em meios persistentes para serem usadas futuramente. A ideia é organizar como uma tabela de duas colunas (atributo e valor), que são consumidas por elementos de comunicação.

- Memória compartilhada distribuída.

Une a memória de todos sistemas distribuídos para que os elementos de comunicação leiam e escrevam nela.

localmente evitando requisições ao servidor.

- Código móvel. Ex: Applets; Código hospedado em uma máquina movido para a máquina no cliente para ser executado localmente.
- Agentes móveis. Ex: worms e web crawlers. Semelhante ao código móvel, o código migra de máquina em máquina.

### Elementos arquitetônicos

- Posicionamento

Forma de posicionamento de informações para garantir acesso às informações.

- Mapeamento do serviço em vários servidores. Ex: NIS;  
Pode-se hospedar serviços em diversos servidores espalhados. NIS: onde é armazenado informações de usuários Linux.

#### PEDIR EXPLICAÇÃO

- Uso de cache;  
Informações acessadas em períodos de tempo curtos podem ser armazenadas

### Padrões arquitetônicos

Estruturas disponíveis para essa arquitetura:

- Camadas lógicas
  - Plataforma. ex: Intel x86/Windows;  
Conceito de plataforma: hardware + sistema operacional. A quantidade grande de plataformas traz uma heterogeneidade muito grande.
  - Middleware: mascara a heterogeneidade;  
Os middlewares são camadas de software para mascarar as heterogeneidades. Assim as aplicações são

## Aula 02 – Modelos de Sistema

abstraídas para serem executadas em qualquer plataforma.

- Arquitetura de camadas físicas

As camadas físicas são decompostas em 3 funcionalidades.

- Decomposição de funcionalidade de aplicações
  - Apresentação; Parte da aplicação que exibe a informação para o usuário.
  - Lógica de negócio; Onde está a regra de negócio, onde são executadas as funções para que a aplicação funcione.
  - Persistência; Onde são armazenados os dados processados pela regra de negócio.

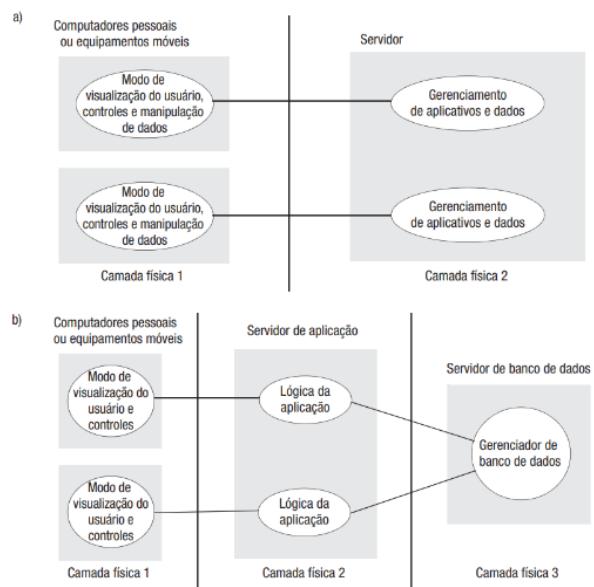


Figura 1. Arquitetura de duas e de três camadas físicas.

1. **Modelo tradicional:** divisão em duas camadas físicas, de um lado apenas a camada de regras de negócio e persistência e do outro apenas apresentação.

2. **Modelo mais recente:** Separado em 3 camadas, a camada de persistência (onde há o gerenciamento dos dados), a camada de regra de negócios (com a lógica da aplicação hospedada no servidor de aplicação), a camada de apresentação (dados apresentados para o usuário).

### A tecnologia AJAX

Maneira de apresentar informações para o usuário e atualizar em tempo real apenas pequenas partes.

- O javascript – Linguagem de programação independente de navegador e plataforma. Permite programar e executar tanto a interface do usuário como a lógica da aplicação no contexto da janela do navegador.
- O AJAX, baseado em javascript, permite atualizações seletivas de dados e a reação do frontend a novos dados.
- Largamente utilizado nas aplicações web atuais. Ex: Google Maps.

### Funcionamento

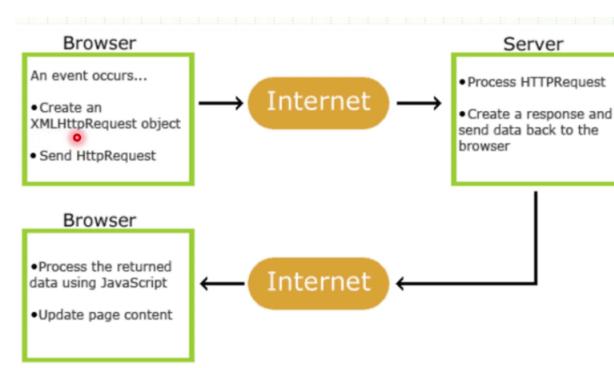


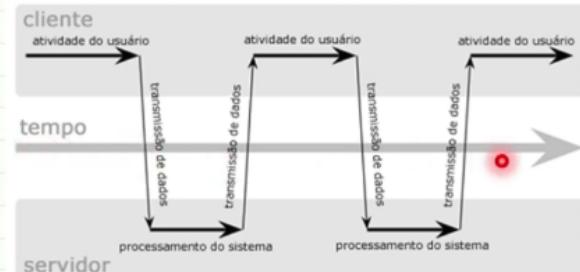
Figura 2. Funcionamento do AJAX.

Quando ocorre um evento, é criado um objeto xml http request por onde é enviada uma requisição ao servidor. No servidor, a requisição

chega e é criada uma resposta a ser enviada ao navegador. A resposta é recebida e o dado é atualizado na página, sem interromper o uso do site para o usuário.

### Clássico X AJAX

modelo de aplicação web clássico (síncrono)



modelo de aplicação web Ajax (assíncrono)

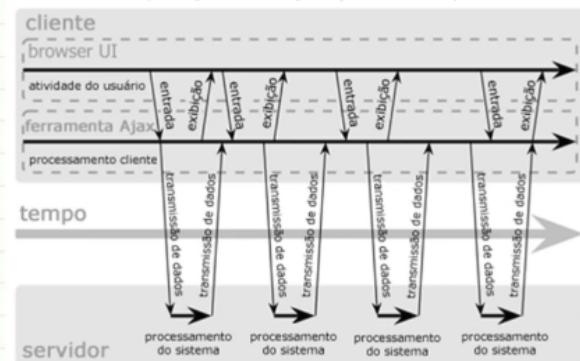


Figura 3. Diferença entre modelos de implementação web.

### O objeto XMLHttpRequest

Usado na comunicação com o servidor;

Criado pela Microsoft e padronizado pelo W3C;

```
var xmlhttp;
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=funcaoTrataRespostaServidor;
xmlhttp.open("GET", "nome_recurso_servidor",true);
xmlhttp.send();
```

Figura 3. Exemplo de criação do objeto XMLHttpRequest.

Método open: tipo do método, nome do recurso, síncrona/assíncrona.

Trabalho: Criar uma pequena aplicação usando Ajax.

Bom tutorial: ???

## Soluções de Middleware

Principais categorias	Subcategoria	Exemplos de sistemas
Objetos distribuídos (Capítulos 5, 8)	Padrão	RM-ODP
	Plataforma	CORBA
	Plataforma	Java RMI
Componentes distribuídos (Capítulo 8)	Componentes leves	Fractal
	Componentes leves	OpenCOM
	Servidores de aplicação	SUN EJB
	Servidores de aplicação	CORBA Component Model
	Servidores de aplicação	JBoss
Sistemas publicar-assinar (Capítulo 6)	–	CORBA Event Service
	–	Scribe
	–	JMS
Filas de mensagem (Capítulo 6)	–	Websphere MQ
	–	JMS
Serviços web (Capítulo 9)	Serviços web	Apache Axis
	Serviços de grade	The Globus Toolkit
Peer-to-peer (Capítulo 10)	Sobreposições de roteamento	Pastry
	Sobreposições de roteamento	Tapestry
	Específico da aplicação	Squirrel
	Específico da aplicação	OceanStore
	Específico da aplicação	Ivy
	Específico da aplicação	Gnutella

Figura 3. Categorias de middleware.

## Modelos fundamentais – Modelo de interação

Como se dá a interação entre os elementos que compõem os sistemas distribuídos?

- Um SD consiste de muitos processos interagindo de maneira complexa.

Processos interagindo de maneira complexa.

- Dificuldade de descrever todos os estados de um algoritmo distribuído.

Em função dessa complexidade de comunicação, existe uma

dificuldade de determinar o estado de algoritmos distribuídos, pois são várias máquinas trabalhando em estados diferentes.

- Desempenho da comunicação.

- Latência;

É importante ter uma baixa latência: tempo curto entre a requisição e uma resposta.

- Largura de banda;

Quantidade de dados enviados por vez. Maior largura de banda = maior vazão., melhor desempenho de comunicação

- Jitter - variação no atraso de entrega de dados.

atrapalha principalmente aplicações de sistemas multimídia distribuídos. é a variação da latência no envio de mensagens, variação no atraso de entrega da mensagem. Essa variação deve ser a menor possível para garantir constância.

- Relógios de computador e eventos de temporização.

Há sempre uma defasagem entre uma máquina e outra porque não existe relógio global que sincronize.

**Solução:** Garantir a existência de um relógio e garantir que esse relógio sincronize os elementos que compõem o sistema distribuído. Contudo, sempre há uma diferença entre o tempo das máquinas.

### PEDIR EXEMPLO

#### Modelo de interação - Variantes

- Síncronos

O tempo é fundamental para um bom funcionamento. Os relógios precisam estar sincronizados o máximo possível.

- Tem limites de tempo conhecido para executar cada tarefa;
- Tempo limite para recepção de uma mensagem;
- O relógio de cada processador tem um desvio máximo do tempo real conhecido;

- Assíncronos

Os limites de tempo não são previamente estabelecidos e não são tão importantes.

- Nada do síncrono;

- Ex: Internet.

- Ordenação de eventos - Em alguns casos é importante a ordem dos eventos.

#### Ordenação de eventos - Email

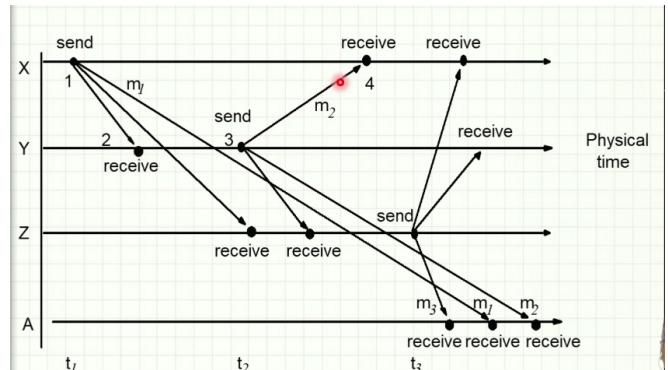


Figura 4. Exemplo de ordenação de eventos.

As mensagens chegam de forma desordenada, assim uma solução seria enviar junto com a mensagem a hora de envio para ordenação. Contudo, se os relógios das máquinas estiverem errado, irá ordenar errado.

Lamport usa enumeração de mensagens para ordenar os eventos.

#### Modelo de falhas

- Falhas por omissão de processo - O processo entra em colapso ("deu pau");

O processo trava e não consegue mais executar a operação, deve ser excluído do sistema distribuído.

- **Falhas por omissão na comunicação** - Perda de mensagens na comunicação; Um determinado perdeu comunicação com a rede, assim ocorrem falhas no processo.
- **Falhas arbitrárias** - ex: Mensagens corrompidas, duplicadas, ausentes;
- **Falhas de temporização** - Aplicada a SD síncronos; Quando o limite máximo de tempo de envio ou recepção é excedido.
- **Mascaramento de falhas** - Uso de técnicas para ocultar as falhas: réplicas, checksum, retransmissão;  
**réplicas:** tratam a omissão, criam clones de processos para caso algo falhe.  
**checksum:** trata mensagens corrompidas.  
**retransmissão:** reenvio de mensagem ao perceber que houve falha.
- **Confiabilidade da comunicação de um para um**
  - garantir validade e integridade.

**validade:** garantir que a mensagem saia de um nó e chegue ao destino.

**Integridade:** garantir que a mensagem chegue sem alteração.

### Modelo de segurança

- Processos e canais seguros; Usar técnicas para garantir seguranças dos canais e processos. Garantir validade, integridade, confiabilidade.
- Proteção de objetos: Direitos de acesso; Outra técnica para garantir segurança é ACL (direitos de acesso), onde são estipulados que acessos podem ser feitos a um objeto.

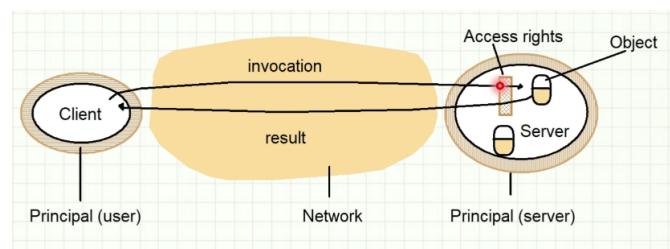


Figura 5. Exemplo de ACL.

Cliente envia uma requisição que passa por uma camada chamada "Access Rights" a qual verifica se o mesmo possui direito de acesso e caso possua, envia a resposta.

## O invasor

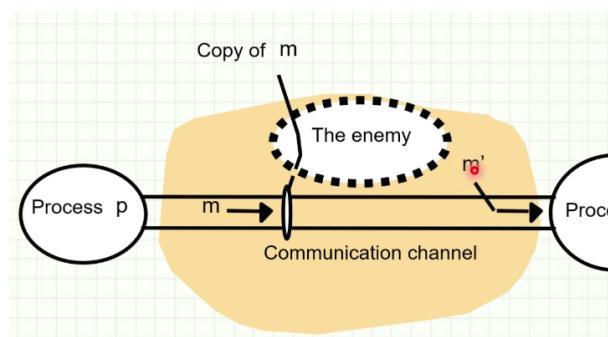


Figura 6. Ataque “men in the middle”.

Ataque no qual o indivíduo captura e faz cópias das mensagens, as intercepta.

## Modelo de segurança

- Ameaças aos processos. ex: IP spoofing;  
Quando é enviada uma mensagem com endereço IP de outra pessoa. Essas mensagens falsas devem ser detectadas e evitar que sejam processadas.
- Ameaça aos canais de comunicação;
  - Possibilidade de alterar, copiar ou injetar mensagens.
  - O uso de criptografia e autenticação;  
**Criptografia** garante a confidencialidade, pois evita que a mensagem seja interceptada e alterada.

**A autenticação** garante que a mensagem está sendo enviada por quem deve enviar.

- O uso de canais seguros. Ex: VPN;

**VPN:** redes virtuais privadas, onde os dados são criptografados no envio.

- Outras ameaças

- Negação de serviço; Envio grande de requisições com objetivo de parar o sistema, negando serviço a outras pessoas.
- Código móvel; Códigos que migram de uma máquina para outra.

- Ponderar a eficácia e o custo das técnicas de segurança em relação às ameaças.

Quanto mais segurança, mais recursos são requeridos, consumindo mais banda por exemplo. Deve-se avaliar bem quais técnicas usar para ser seguro e eficiente.