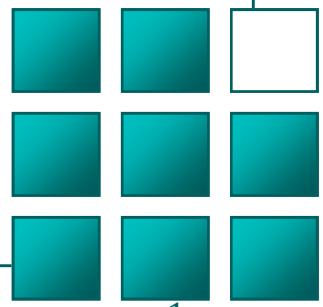
Programa de Pós-graduação em Informática

Tópicos em Sistemas de Computação – Computação em Nuvem

Aletéia Patrícia Favacho de Araújo

Aula 1 – Introdução



O que é um Sistema Distribuído?

- Não existe definição única!
- Segundo Enslow:
  - Multiplicidade de recursos (físicos e lógicos);
  - Distribuição física (componentes físicos e lógicos);
  - SO para unificar e integrar;
  - Autonomia cooperativa.

O que é um Sistema Distribuído?

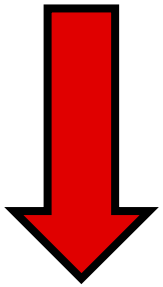
- Segundo Müllender:
  - Múltiplos elementos de processamento;
  - Existência de hardware de interconexão;
  - Elementos de processamento falham independentemente;
  - Existência de estado compartilhado.

O que é um Sistema Distribuído?

- Segundo Coulouris:
  - Um Sistema Distribuído consiste de uma coleção de computadores autônomos, interligados através de uma rede de computadores e equipados com softwares distribuídos.
- Segundo Tanenbaum:
  - Um Sistema Distribuído é uma coleção de computadores independentes que, para o usuário, aparentam ser um sistema centralizado.

O que é um Sistema Distribuído?

- Conceito chave



TRANSPARÊNCIA

O uso de múltiplos processadores deve ser invisível ao usuário

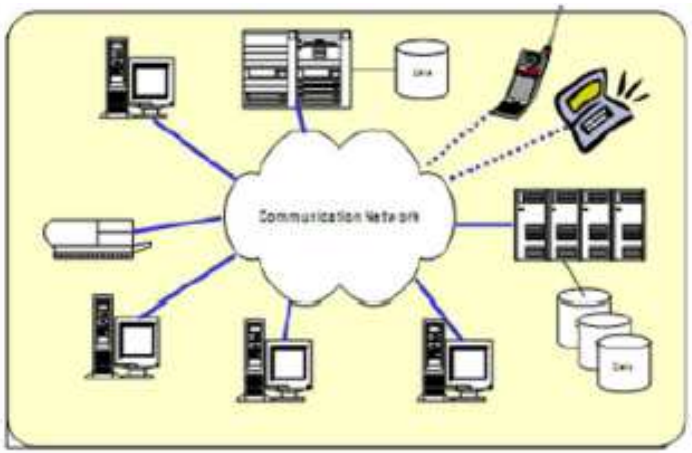
O que é um Sistema Distribuído?

Received: by jumbo.dec.com (5.54.3/4.7.34)
id AA09105; Thu, 28 May 87 12:23:29 PDT
Date: Thu, 28 May 87 12:23:29 PDT
From: lamport (Leslie Lamport)
To: src-t
Subject: distribution
There has been considerable debate over the years about what constitutes a distributed system. It would appear that the following definition has been adopted at SRC:

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

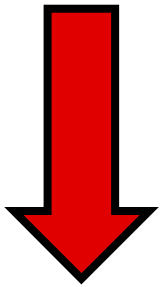
O que é um Sistema Distribuído?

- Um sistema distribuído é uma coleção de computadores independentes ligados por uma rede, que do ponto de vista do usuário do sistema se comportam como um único computador.



Motivação p/ Sistemas Distribuídos

- Microprocessadores
- Redes Locais de Alta Velocidade (LANS)



Sistemas Distribuídos

Motivação p/ Sistemas Distribuídos

- Computadores isolados não atendiam a todas as necessidades;
- Baixa qualidade dos serviços dos sistemas centralizados + crescente exigência dos usuários;
- Garantia de melhor desempenho.

Sistemas Distribuídos *versus* Sistemas Centralizados

Vantagem	Descrição
Economia	Os microprocessadores oferecem um custo/benefício melhor que o dos *mainframes*.
Velocidade	Um sistema distribuído pode ter mais poder de processamento que um *mainframe*.
Distribuição Inerente	Algumas aplicações envolvem máquinas espacialmente separadas.
Confiabilidade	Se uma máquina falha, o sistema como um todo ainda pode prosseguir.
Escalabilidade	Pode-se adicionar poder de processamento em pequenas parcelas.

Vantagens dos sistemas distribuídos sobre os sistemas centralizados.

Sistemas Distribuídos *versus* Microcomputadores Isolados

Vantagem	Descrição
Compartilhamento de Dados	Um sistema distribuído permite que muitos usuários acessem um banco de dados comum.
Compartilhamento de Dispositivos	Um sistema distribuído possibilita que muitos usuários compartilhem periféricos caros.
Comunicação	A comunicação entre as pessoas pode ser bastante facilitada por um sistema distribuído.
Flexibilidade	Distribuir a carga de trabalho sobre as máquinas disponíveis é uma forma efetiva de diminuir custos.

Vantagens dos sistemas distribuídos sobre os microcomputadores isolados.

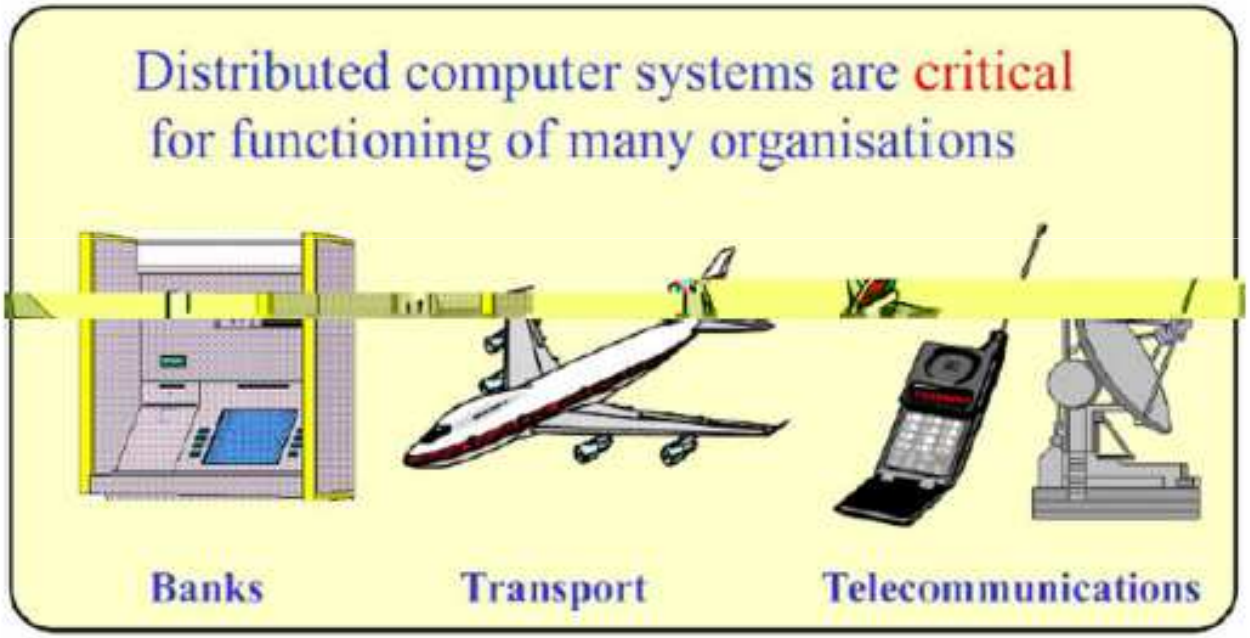
Desvantagens dos Sistemas Distribuídos

No pain, no gain!

Desvantagens dos Sistemas Distribuídos

Desvantagem	Descrição
Software	Implementar software eficiente distribuído/paralelo é mais complexo que software centralizado.
Rede	A rede pode saturar ou causar outros problemas.
Segurança	O acesso fácil também se aplica aos dados secretos.

Qual a Importância...



...o Mundo é distribuído!!!

Conceitos de Hardware

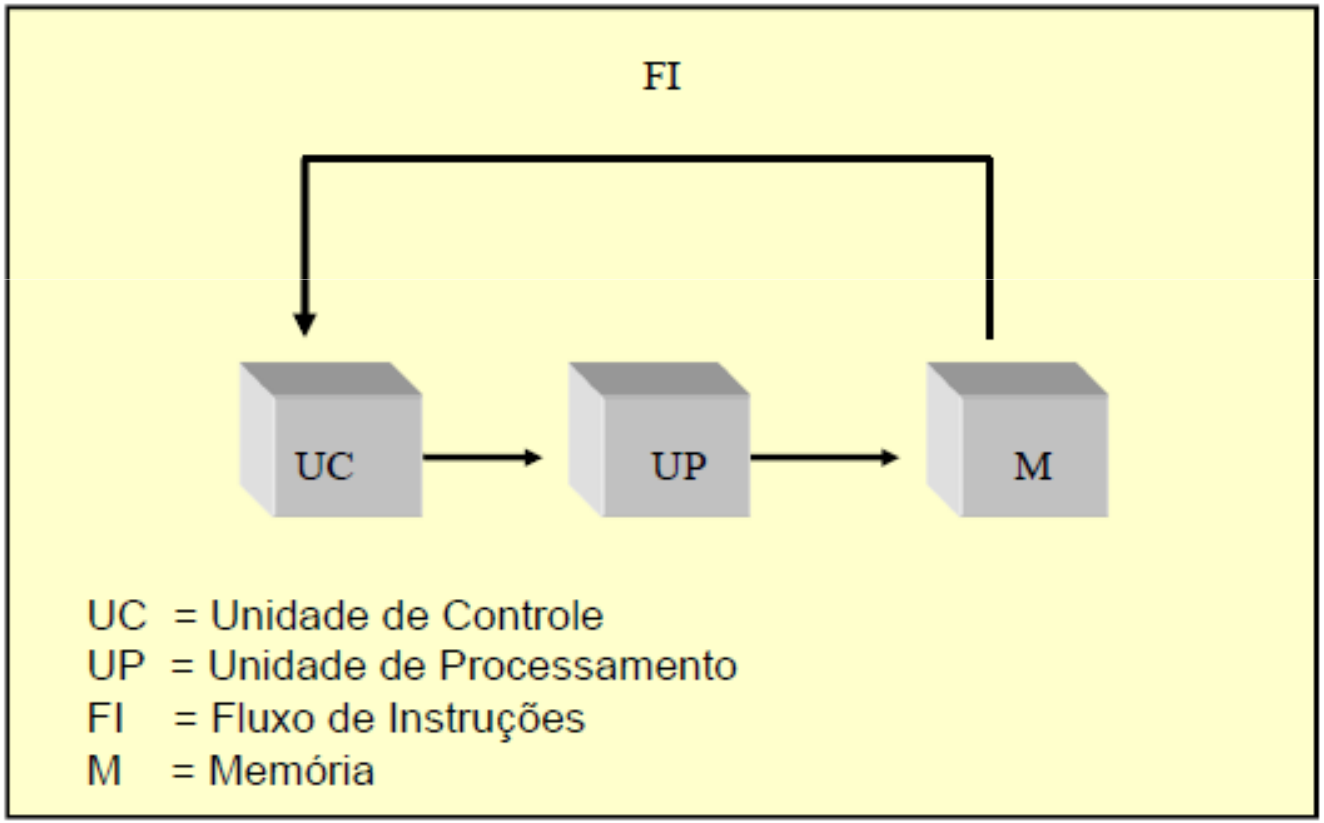
Classificação de *Flynn*

SISD	Computadores com somente um fluxo de instruções e um fluxo de dados
SIMD	Computadores com um fluxo de instruções e múltiplos fluxos de dados
MISD	Computadores com múltiplos fluxos de instruções e um único fluxo de dados
MIMD	Computadores com múltiplos fluxos de instruções e múltiplos fluxos de dados

Classificação de *Flynn - SISD*

- *Single Instruction Stream/Single Data Stream* (Fluxo único de instruções/Fluxo único de dados):
  - Corresponde ao tradicional modelo de von Neumann;
  - Um processador executa seqüencialmente um conjunto de instruções sobre um conjunto de dados.

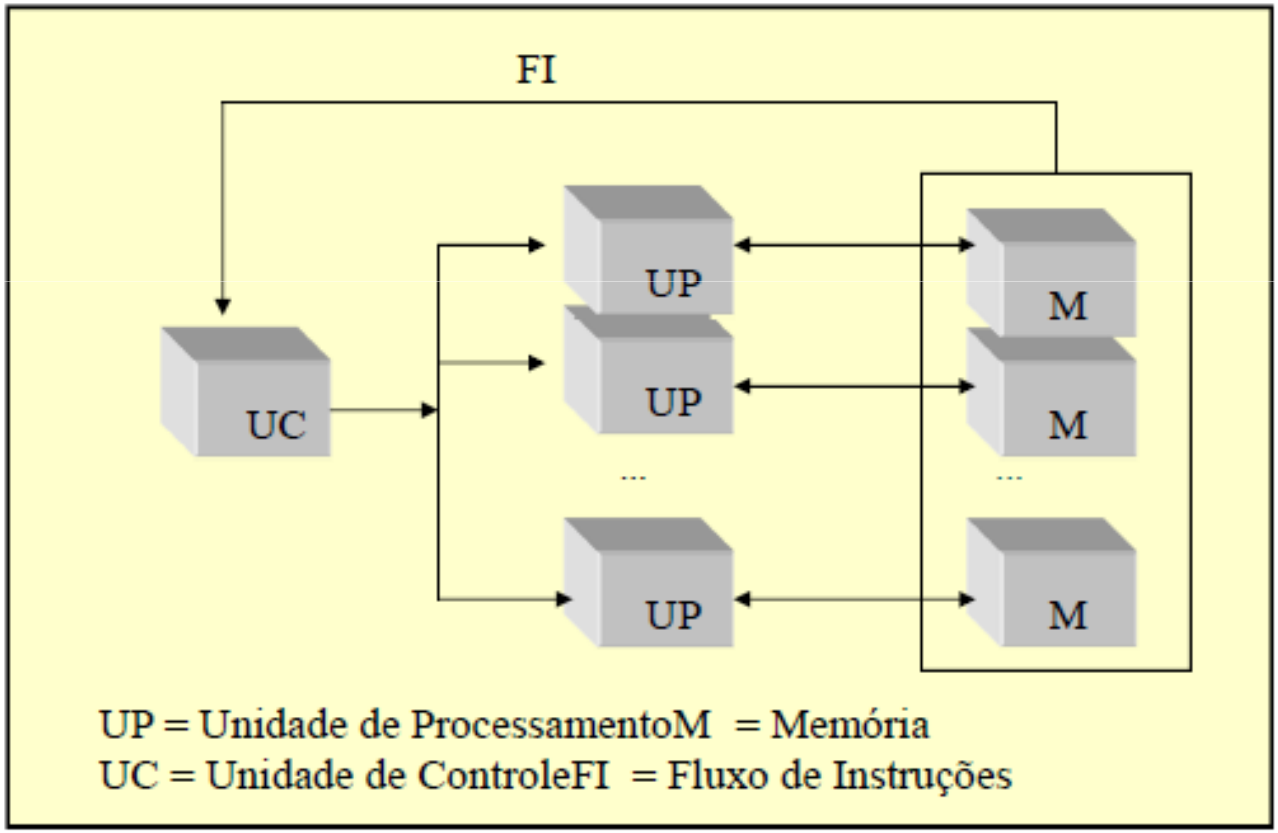
Classificação de *Flynn - SISD*



Classificação de *Flynn - SIMD*

- *Single Instruction Stream/Multiple Data Stream* (Fluxo único de instruções/Fluxo múltiplo de dados)
  - Envolve múltiplos processadores (escravos) sob o controle de uma única unidade de controle (mestre), executando simultaneamente a mesma instrução em diversos conjuntos de dados;
  - Arquiteturas SIMD são utilizadas, por exemplo, para manipulação de matrizes e processamento de imagens.

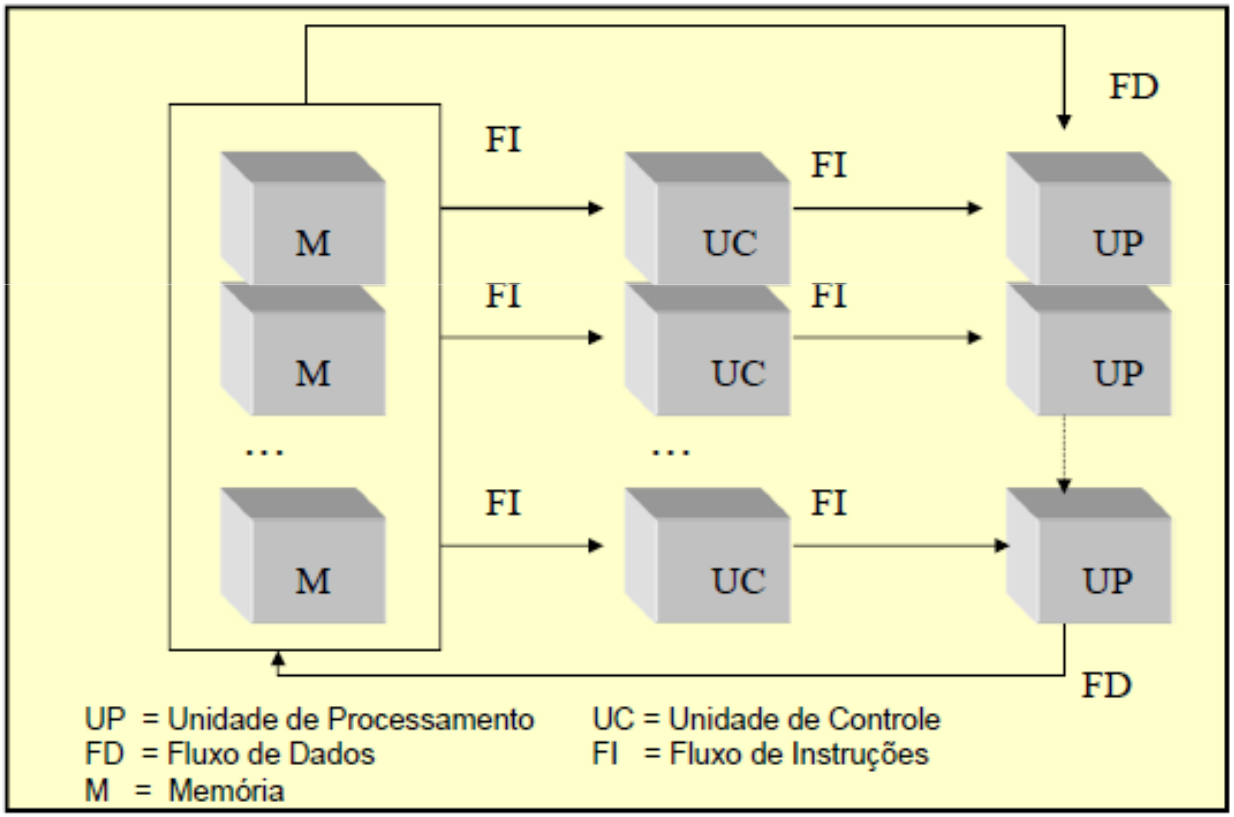
Classificação de *Flynn - SIMD*



Classificação de *Flynn - MISD*

- *Multiple Instruction Stream/Single Data Stream* (Fluxo múltiplo de instruções/Fluxo único de dados)
  - Envolve múltiplos processadores executando diferentes instruções em um único conjunto de dados;
  - Há um impasse na literatura, e alguns autores, geralmente, consideram que nenhuma arquitetura é classificada como MISD, isto é, não existem representantes desta categoria;
  - Por outro lado, há autores que classificam as máquinas que implementam *pipeline* de processador como sendo deste grupo.

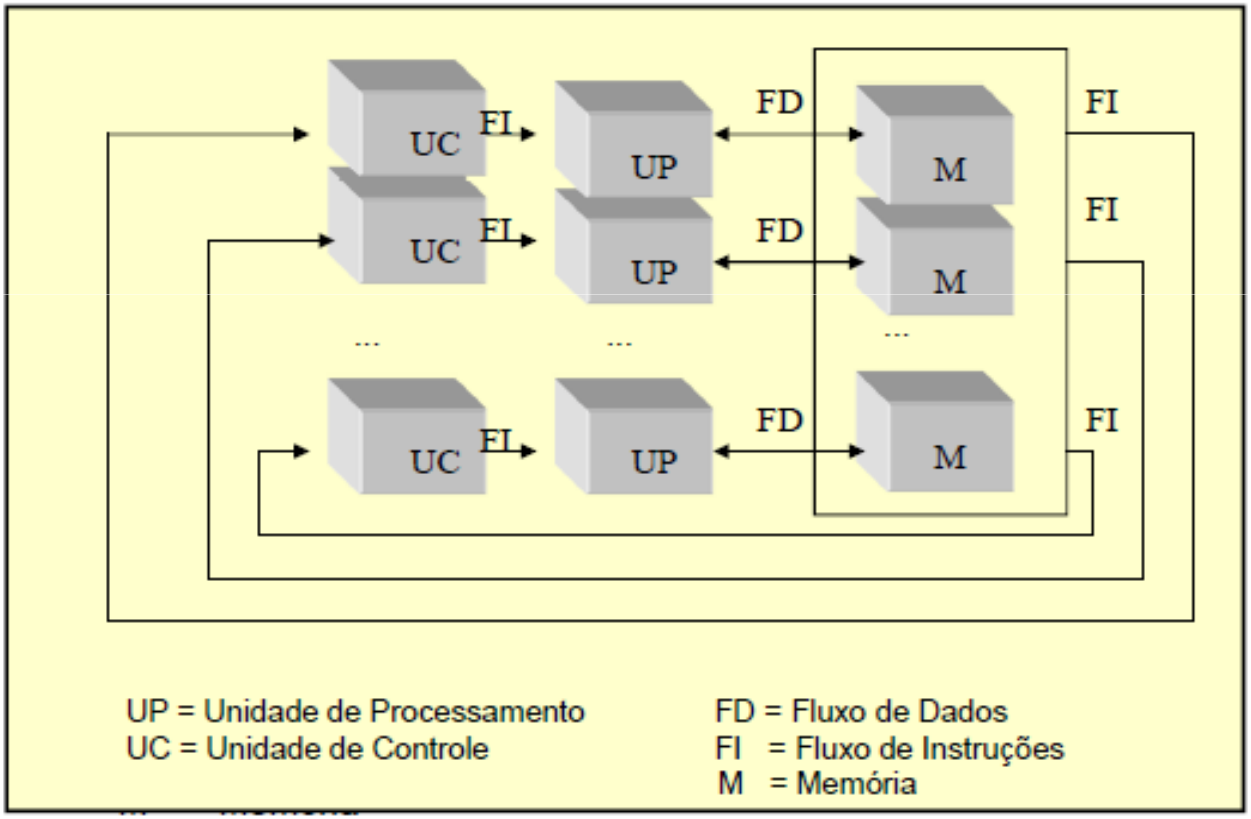
Classificação de *Flynn - MISD*



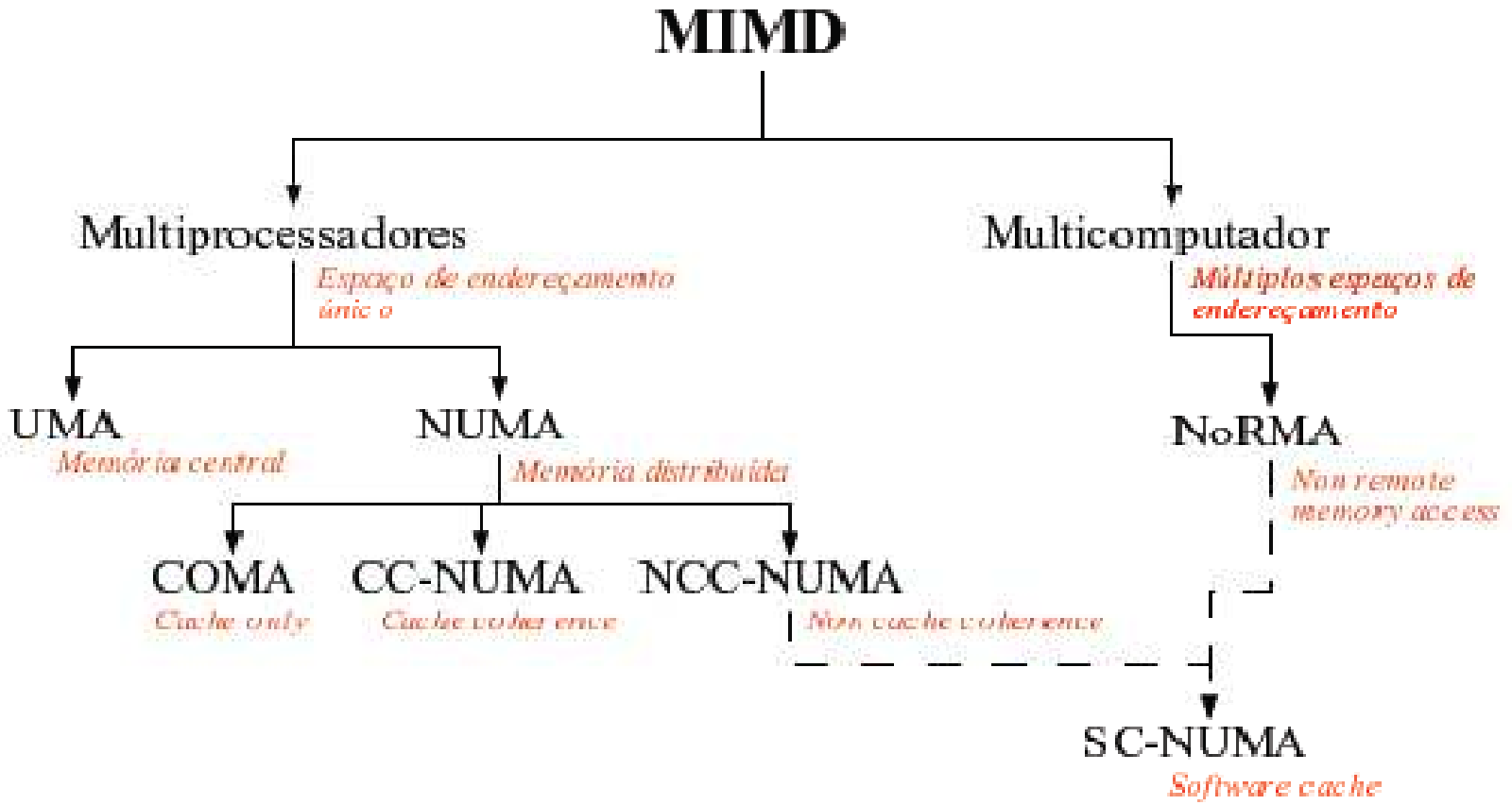
Classificação de *Flynn - MIMD*

- *Multiple Instruction Stream/Multiple Data Stream (Fluxo múltiplo de instruções/Fluxo* múltiplo de dados)
  - Envolve múltiplos processadores executando diferentes instruções em diferentes conjuntos de dados, de maneira independente.

Classificação de *Flynn - MIMD*



Classificação Quanto ao Acesso à Memória



Máquinas MIMD

- Pode-se dividir os computadores MIMD em:

Multiprocessadores	– Fortemente acoplados – Memória compartilhada
Multicomputadores	– Fracamente acoplados – Memória privada

Fortemente Acoplados *(Tightly Coupled)*

- Fortemente Acopladas (*Tightly Coupled*):
  - Nesta plataforma, os processadores compartilham a memória (único espaço de endereçamento) e são interligados por um barramento.
  - A comunicação entre os processos é através de leitura e escrita na memória.
  - O sistema fortemente acoplado mais comum é definido como Multiprocessamento Simétrico (SMP), onde não há hierarquia entre os processadores e cada processador executa uma cópia idêntica do SO que gerencia o compartilhamento dos recursos.

Fracamente Acoplados *(Loosely Coupled)*

- Fracamente Acoplados (*Loosely Coupled*):
  - Neste caso, os computadores estão interligados entre si, mas mantém sua “individualidade”.
  - Isso significa que cada computador tem seus próprios recursos de processador e memória local, e utilizam um sistema de comunicação para trocar mensagens com os demais computadores da rede.
  - Não há necessidade dos computadores envolvidos na rede possuírem configurações similares de hardware ou software podendo, cada computador, ter seu SO particular.

Fracamente Acoplados *(Loosely Coupled)*

- Fracamente Acoplados (*Loosely Coupled*):
  - Todavia, é necessário conhecer o protocolo de comunicação.
  - E como é esperado, a comunicação por meio de troca de mensagens reduz o desempenho do processamento global.
  - Assim, um sistema fracamente acoplado com N processadores, raramente, tem um desempenho N vezes maior que um processador.

Máquinas Paralelas

- Atualmente, sistemas com mil CPUs já estão disponíveis no mercado.
- TOP500, **Tianhe-2**, um supercomputador desenvolvido na Universidade Nacional da China, com uma performance de 33.86 petaflop/s (quadrilhões de cálculos por segundo) é o computador mais rápido do Mundo.
- Ele tem **3.120.000 núcleos**, e é uma máquina com sistema operacional Kylin Linux.
- Essa máquina já está há 2 anos e meio (cinco edições) no primeiro lugar do *rank* TOP 500.

TOP 500

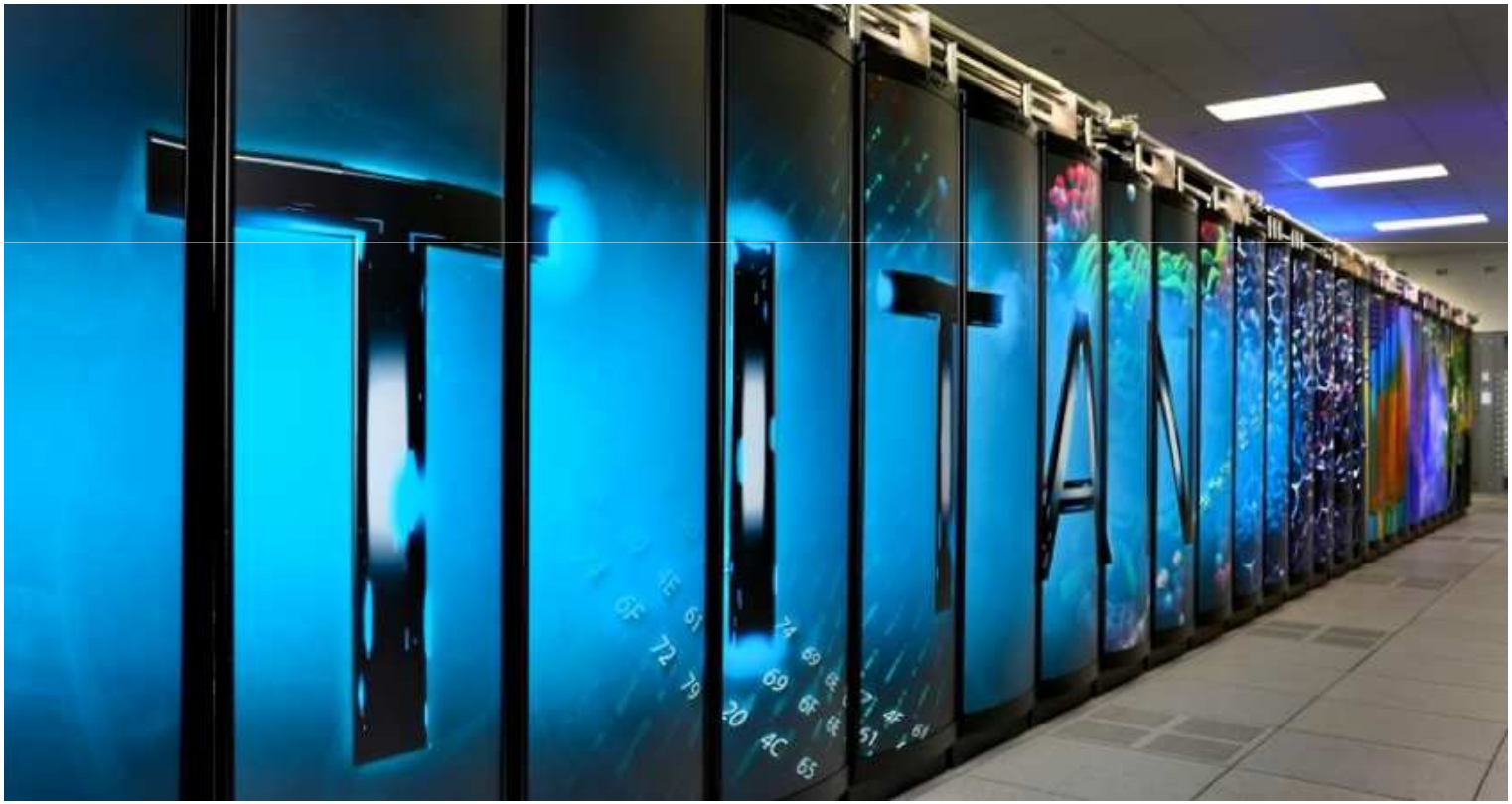
- O segundo lugar é o **Titan**, nos Estados Unidos, com **560.640 núcleos.** É uma máquina Cray Inc, com sistema operacional Cray Linux Environment. Ela entrou no TOP 500 em Novembro de 2012, em primeiro lugar. Em Junho de 2013 passou para segundo e está na mesma posição até hoje.
- O terceiro lugar é a máquina **Sequoia**, localizada nos Estados Unidos. Ela tem **1.572.864 núcleos**, e é uma máquina da IBM. Entrou no *rank* em novembro de 2011, em 17° lugar; Em junho de 2012 passou para primeiro; Em novembro de 2012 para segundo; Em junho de 2013 ficou em terceiro, e está até hoje.

Top 500

- **Tianhe-2**, o mais rápido do Mundo.

TOP 500

- **Titan**, o segundo mais rápido com 560.640 núcleos.



TOP 500

- **Segoia**, o terceiro mais rápido com 1.574.864 núcleos, localizado nos Estados Unidos

TOP 500

- **Computer K** no Japão é o quarto computador mais rápido do Mundo, ele tem **705.024 núcleos**

TOP 500 - Brasil

- No Brasil:
  - **Rank 146** – LNCC (Lab. Nacional de Computação Científica), 10.692 núcleos. Entrou na última listagem.
  - **Rank 165** – SENAI/CIMATEC, 17.200 núcleos (cluster).
    - Na listagem anterior (maio deste ano) estava no *rank* 124.
    - Na listagem de 2014/6 estava na posição 95.
    - SENAI/CIMATEC é o Campus Integrado de Manufatura e Tecnologia.

TOP 500 - Brasil

- No Brasil:
  - **Rank 178** – LNCC, 24.732 núcleos. Entrou na última listagem.
  - **Rank 208** – LNCC, 18.144 núcleos. Entrou na última listagem.
  - **Rank 285** (em 2012 era *rank* 68, em 2015/6 era o *rank 224*) – Petrobras, 17.408 núcleos (cluster).
  - **Rank 347** – INPE, até 2015/6 era o *rank* 281. Máquina com 31.104 núcleos (super computador).
  - A Petrobras perdeu uma máquina no TOP 500, na última listagem.

Realidade...

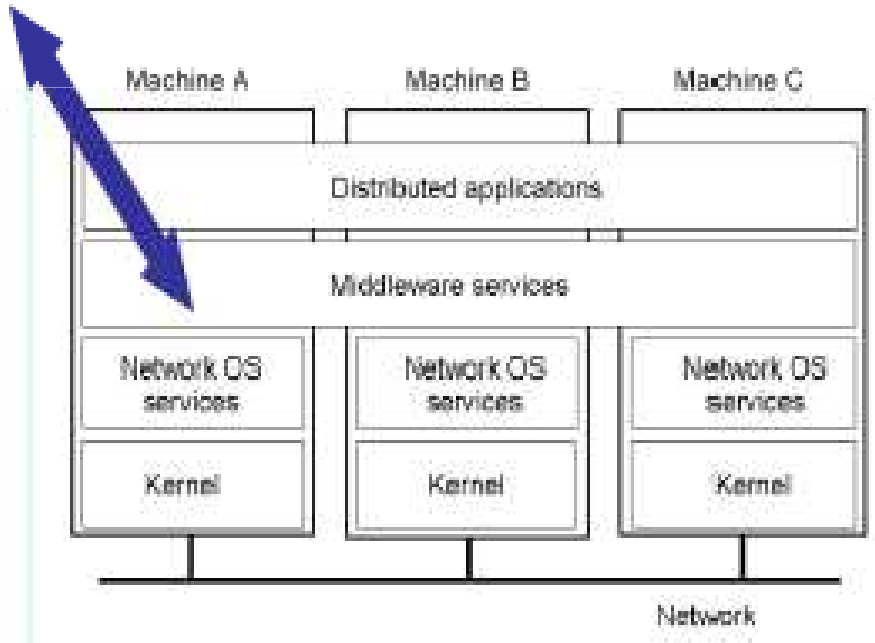
- Sistemas com milhões de CPUs já são realidade nesta década.
- Colocar um milhão de computadores em uma sala é fácil desde que se tenha bastante dinheiro e espaço suficiente.
- O espaço deixa de ser um problema se esses computadores estiverem distribuídos ao redor do mundo.

Programação em Sistemas Distribuídos

- A construção de aplicação distribuída sobre recursos de redes de SO pode ser árdua.
- Para facilitar, são usados *middlewares*...
- *Middleware* - serviços e abstrações que facilitam o desenvolvimento de aplicações distribuídas.
- Situado logicamente entre uma camada de nível mais alto, composta de usuários e aplicações, e uma camada adjacente (SOs e comunicação).

Middleware

- A camada de *middleware* se estende por várias máquinas e oferece a mesma interface a cada aplicação.



Middleware

- *Middleware* permite ao sistema distribuído conseguir uniformidade na presença de diferentes hardwares e SOs:
  - Oculta a distribuição: o fato que a aplicação está sendo executada em diferentes máquinas distribuídas geograficamente.
  - Oculta a heterogeneidade: diferentes sistemas operacionais, diferentes protocolos de comunicação.

Características Essenciais

- Heterogeneidade
- Compartilhamento de recurso
- Abertura
- Concorrência
- Escalabilidade
- Tolerância a falhas
- Transparência

Heterogeneidade

- Heterogeneidade
  - Os SDs são construídos sobre uma variedade de redes, SOs, plataformas de hardware e LPs.
  - Os protocolos de comunicação da Internet escondem as diferenças entre redes, os *middlewares* podem lidar com as demais diferenças.
- Escopo:
  - Hardware de computadores
  - Sistemas operacionais
  - Diferentes fornecedores
  - Linguagens de programação

Compartilhamento de Recursos

- É uma das características mais importantes para a funcionalidade de um SD;
- Recurso: termo usado para Hardware, Software e Dados;
- Nesse ponto, aparece o conceito de gerente de recursos que é usado para:
  - Nomeação
  - Mapeamento
  - Controle de concorrência
- Preocupações: Segurança, Manutenção da Consistência e Comunicação.

Abertura

- É a característica que determina com que facilidade um sistema pode ser expandido;
- O grau de abertura de um SD pode ser determinado medindo-se a possibilidade de acrescentar novos serviços ou recursos compartilhados sem causar problemas para os já existentes;
- Um sistema pode ser aberto em relação a uma característica e fechado em relação a outra;
- Por exemplo, aberto no nível de hardware, significa que pode-se mudar memória, discos e outros sem criar nenhum problema no sistema.

Abertura

- Por outro lado, um sistema aberto no nível de software, traduz a possibilidade de incluir novas características do sistema operacional, dos protocolos de comunicação e dos serviços de compartilhamento de recursos;
- Em resumo, um SD aberto é um sistema que oferece serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços;
  - Sistema Aberto: interfaces bem definidas.

Concorrência

- Concorrência é uma característica que existe sempre que se tem M processos disputando a utilização de N recursos (sendo M>N), em um mesmo intervalo de tempo;
- A concorrência pode ocorrer tanto em sistemas com um único processador, quanto em sistemas com múltiplos processadores;
- Dessa forma, concorrência é a atribuição de parcelas de tempos para diferentes processos.

Escalabilidade

- Escalabilidade é a característica que permite um SD operar efetivamente e eficientemente independente do seu tamanho;
- SD escalável:
  - Os softwares não devem necessitar de alterações quando a escala do sistema aumenta.
  - O hardware deve permitir a expansão do sistema.
- Um sistema é escalável se permanece eficiente quando há um aumento significativo no número de recursos e/ou no número de usuário.

Escalabilidade

- A escalabilidade abrange três dimensões:
  - **Tamanho**: adicionar mais usuários e recursos ao sistema.
  - **Geográfico:** até que ponto usuários e recursos podem estar distantes uns dos outros.
  - **Administrativo:** medida da dificuldade de gerenciar o sistema, especialmente quando envolve várias organizações.

Escalabilidade

- Técnicas para a Escalabilidade:
  - Comunicação Assíncrona: oculta a latência de comunicação.
  - Distribuição: particiona um componente em partes menores e espalha essas partes pelo sistema.
  - Replicação: aumenta a disponibilidade e permite balanceamento de carga (pode gerar problema de consistência).

Tolerância a Falhas

- Alto grau de disponibilidade aos seus usuários, mesmo na presença de falhas.
- Duas abordagens:
  - Redundância de Hardware;
  - Recuperação de Software.
- Um SD deve ser projetado de forma a mascarar falhas eventualmente ocorridas, isto é, escondê-las de seus usuários.

Transparência

- Essa é a característica mais importante dos sistemas distribuídos;
- Um sistema, apesar de distribuído, deve parecer ao usuário como uma única máquina, chamado de imagem única;
- A transparência pode ser obtida em diversos níveis;
- A maneira mais simples de obtê-la é escondendo do usuário o fato deles estarem tratando com várias máquinas diferentes.

Transparência

- Seis Aspectos:
  - Transparência de Acesso;
  - Transparência de Localização;
  - Transparência de Migração;
  - Transparência de Replicação;
  - Transparência de Concorrência;
  - Transparência de Paralelismo.

Transparência

Acesso	Permite que recursos locais e remotos sejam acessados com o uso de operações idênticas.
Localização	Permite que os recursos sejam utilizados sem o conhecimento da sua localização física.
Migração	Os recursos podem ser movidos sem afetar a operação de usuários ou aplicações.
Replicação	Permite que os recursos sejam replicados para aumentar a confiabilidade e o desempenho do sistema, sem o conhecimento dos usuários ou suas aplicações.
Concorrência	Múltiplos usuários podem compartilhar recursos automaticamente.
Paralelismo	As atividades podem ocorrer em paralelo sem o conhecimento dos usuários.

Transparência

- Exemplos:
  - E-mail
    - transparência de localização
    - transparência de acesso
    - aleteia@ucb.br
      - Não é necessário saber a localização física
      - Os comandos são os mesmos, independente da localização
  - rlogin (unix)
    - não tem transparência de acesso
      - rlogin é usado para máquinas remotas
    - não tem transparência de localização
      - comando rlogin + nome da máquina

Ciladas em um SD...

- Premissas erradas 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

Modelos de Sistemas Distribuídos

- **Modelos Arquiteturais:**
  - Aspectos físicos
  - Aspectos lógicos
  - ➢ Definem a forma por meio da qual os componentes interagem uns com os outros.
- A organização lógica do conjunto de componentes é chamada de Estilo Arquitetônico.
- A organização física do conjunto de componentes é chamada de Arquitetura do Sistema.

Estilo Arquitetônico

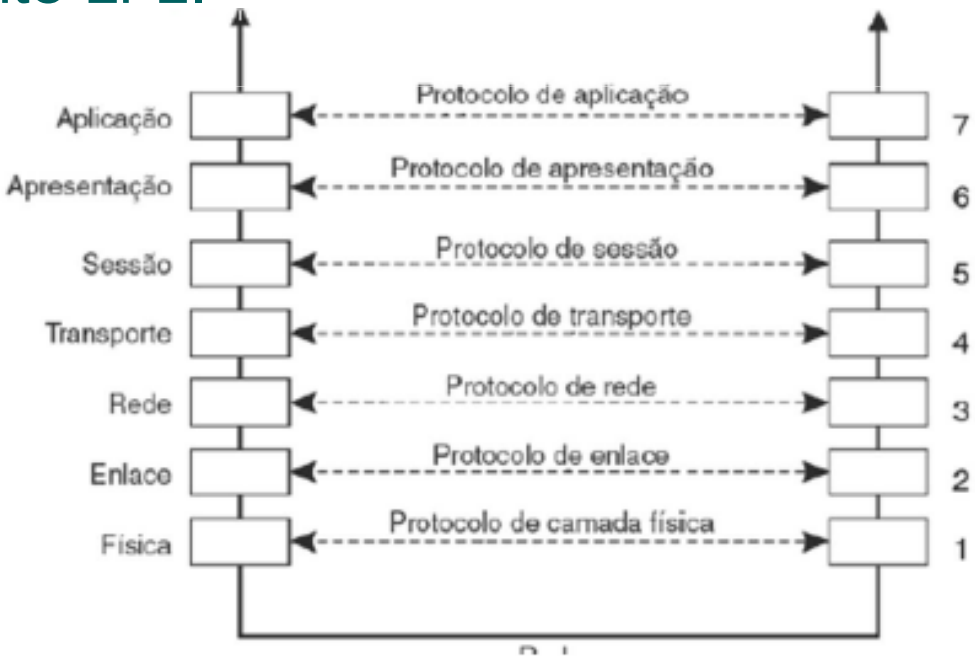
- Um estilo arquitetônico é formulado em termos:
  - de componentes;
  - do modo como esses componentes estão conectados uns aos outros;
  - dos dados trocados entre componentes e, por fim, da maneira como esses componentes são configurados para formar um sistema.
- Um **componente é uma unidade modular com interfaces** requeridas e fornecidas bem definidas que é substituível dentro do seu ambiente.

Estilo Arquitetônico

- Um **conector, em geral, é descrito como um mecanismo** que serve de mediador da comunicação ou da cooperação entre componentes (por exemplo, RPC).
- Assim, usando componentes e conectores, é possível ter várias configurações diferentes, ou estilos arquitetônicos.
- Os principais são:
  - Arquitetura em Camadas
  - Arquitetura Baseada em Objetos
  - Arquitetura Centrada em Dados
  - Arquitetura Baseada em Eventos

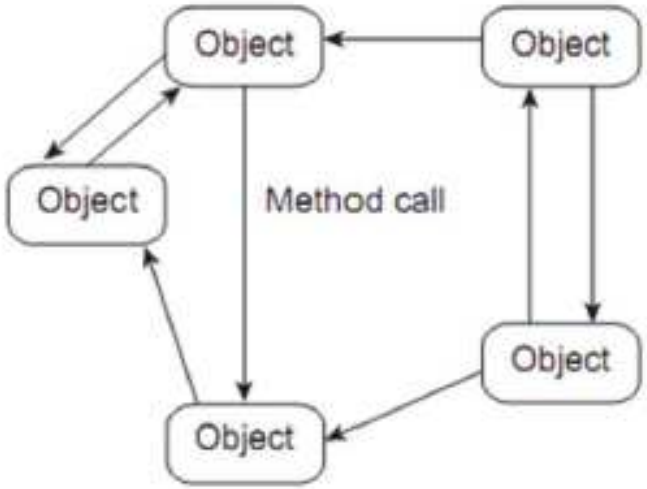
Estilo Arquitetônico – Arquitetura em Camadas

- **Idéia Básica:** um componente na camada Li tem permissão de chamar componentes na camada subjacente Li-1.



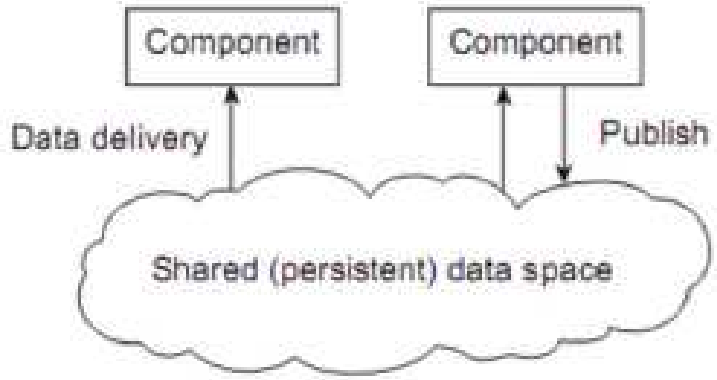
Estilo Arquitetônico – Arquitetura Baseada em Objetos

- **Idéia Básica:** Cada objeto corresponde ao que definimos como componente, e esses componentes são conectados por meio de chamada de procedimento (remota), por exemplo Java RMI.



Estilo Arquitetônico – Arquitetura Centrada em Dados

- **Idéia Básica:** Processos se comunicam por meio de um repositório comum.
- Exemplo: processos que se comunicam por meio da utilização de serviços de dados baseados na web.



Estilo Arquitetônico – Arquitetura Baseada em Eventos

- **Idéia Básica:** Nesse caso os processos demonstram o interesse por um evento ou conjunto de eventos (processo se subscreve) e esperam pela notificação de qualquer um desses eventos, gerados por um processo notificador.
- Em outras palavras, o produtor publica uma informação em um gerenciador de eventos, e o *middleware* assegura que somente os processos que se subscreveram para receber este evento os receberão.
- Exemplo: Consultas em vários bancos de dados.

Estilo Arquitetônico – Arquitetura Baseada em Eventos

- **Idéia Básica:** Nesta arquitetura, processos demonstram o interesse por um evento ou conjunto de eventos (processo se subscreve) e esperam pela notificação de qualquer um desses eventos, gerados por um processo notificador.
- Em outras palavras, o produtor publica uma informação em um gerenciador de eventos, e o *middleware* assegura que somente os processos que se subscreveram para receber este evento os receberão.
- Exemplo: Consultas em vários bancos de dados.

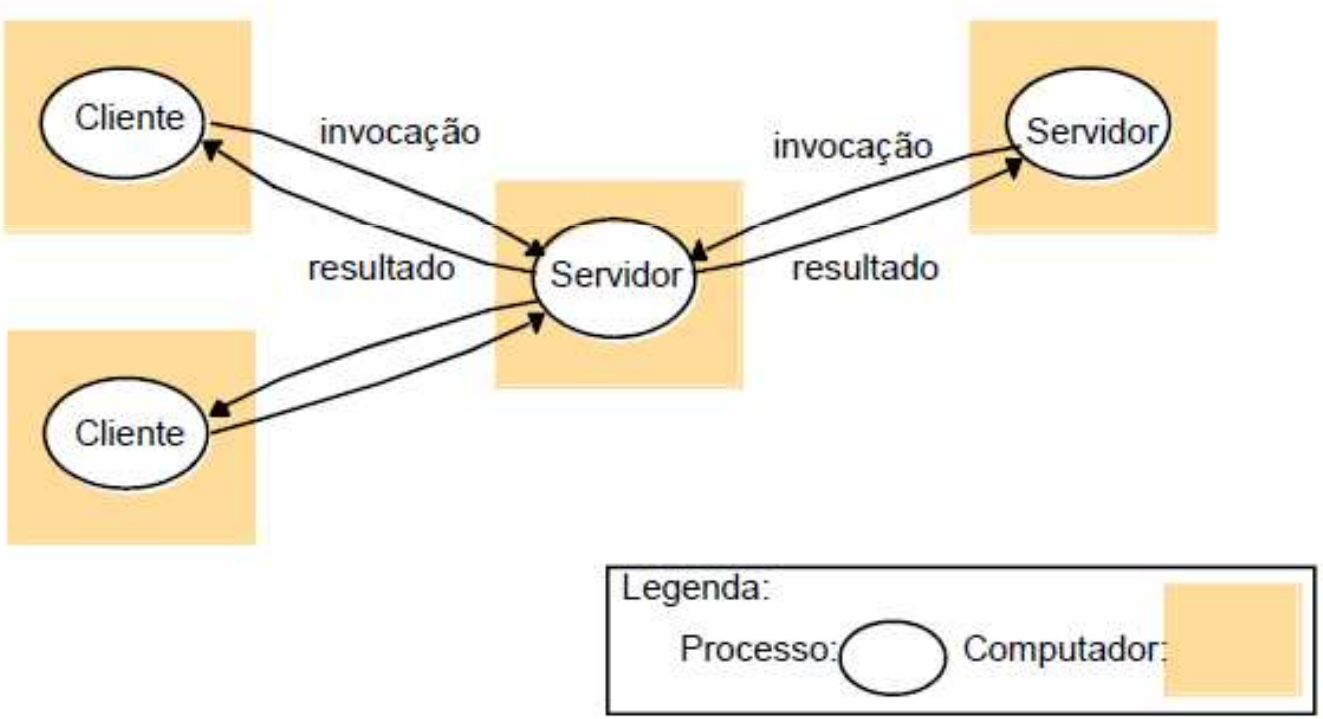
Arquitetura de Sistemas

- Como diversos sistemas distribuídos são realmente organizados?
- Onde são colocados os componentes de software?
- Como é estabelecida a interação entre as peças de software?
- As combinações possíveis para as respostas acima, são chamadas Arquitetura de Sistemas:
  - Cliente-Servidor (ou Arquitetura Centralizada)
  - *Peer-to-Peer (ou Arquitetura Descentralizada)*

Modelo Cliente/Servidor

- Essa é a arquitetura mais comentada dos SDs.
- É ainda a mais importante e continua sendo amplamente empregada.
- Processos são classificados de acordo com as responsabilidades de cada um.
- **Processo servidor** é um processo que implementa um serviço específico – por exemplo, um serviço de sistema de arquivos.
- **Processo cliente** é um processo que requisita um serviço de um servidor enviando-lhe uma requisição e, na sequência, esperando pela resposta do servidor.

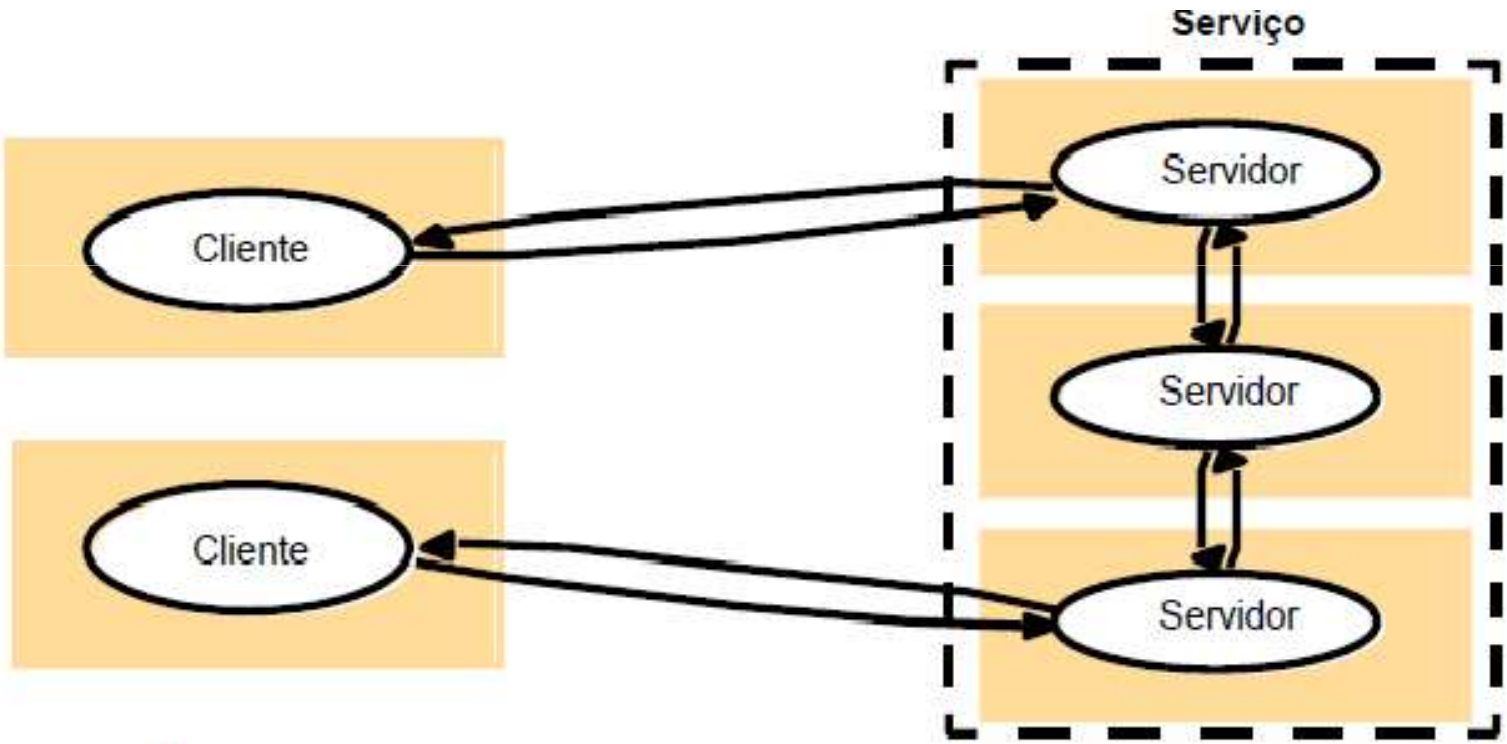
Modelo Cliente/Servidor



Variações

- Diversas variações do modelo básico podem ser extraídas a partir da consideração dos seguintes fatores:
  - O uso de vários servidores e de cache para aumentar o desempenho e a robustez;
  - O uso de código móvel e agentes móveis;
  - A necessidade dos usuários possuírem computadores de baixo custo, com recursos de hardware limitados e simples de gerenciar;
  - A necessidade de adicionar e remover dispositivos móveis de maneira conveniente.

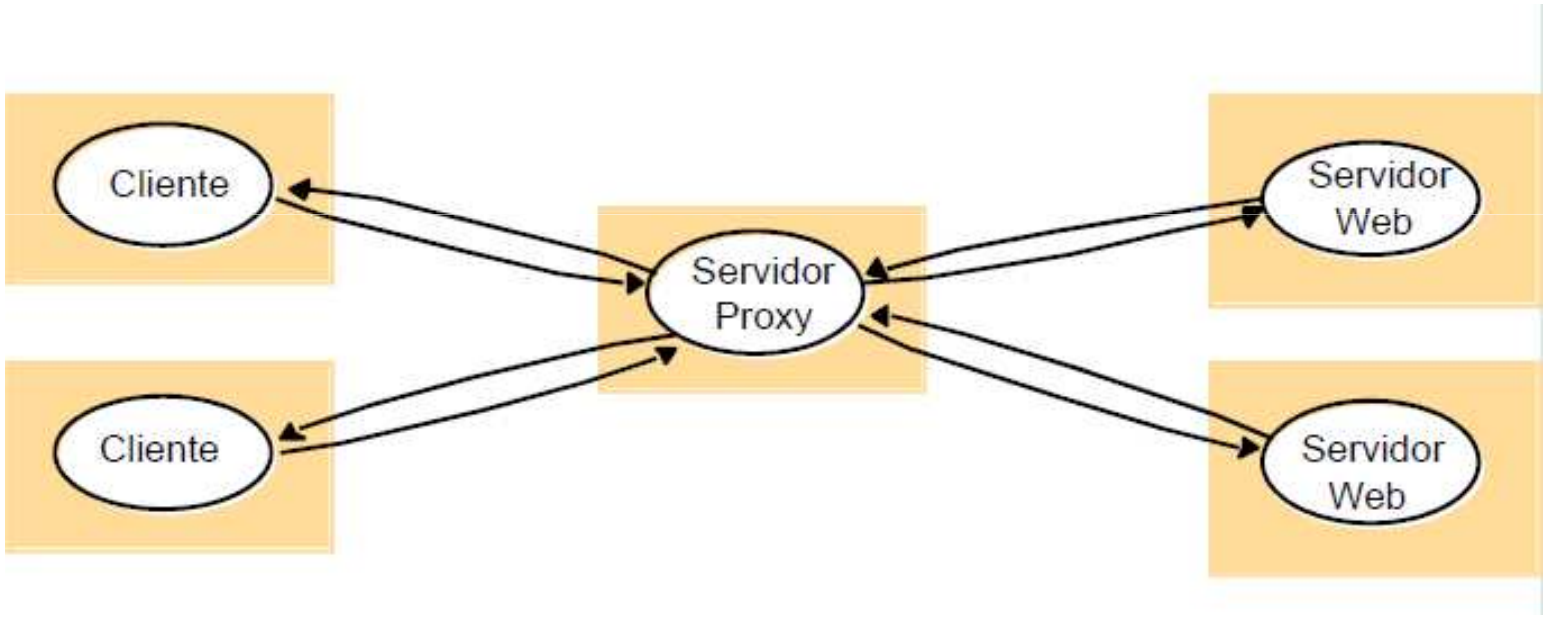
Modelo Cliente/Servidor- Com serviços fornecidos por vários servidores



Modelo Cliente/Servidor- Com serviços fornecidos por vários servidores

- Os serviços podem ser implementados como vários processos servidores em diferentes computadores. Assim, pode-se:
  - Particionar o conjunto de objetos nos quais o serviço é baseado e distribuí-los entre eles mesmos.
    - A web oferece um exemplo de particionamento de dados, no qual cada servidor web gerencia seu próprio conjunto de recursos. O usuário pode usar um navegador para acessar um recurso em qualquer um desses servidores.
  - Manter cópias duplicadas deles em várias outras máquinas (exemplo, NIS, d*a SUN).*
    - Cada servidor NIS tem sua própria réplica de um arquivo de senhas que contém uma lista de *login e senhas*.

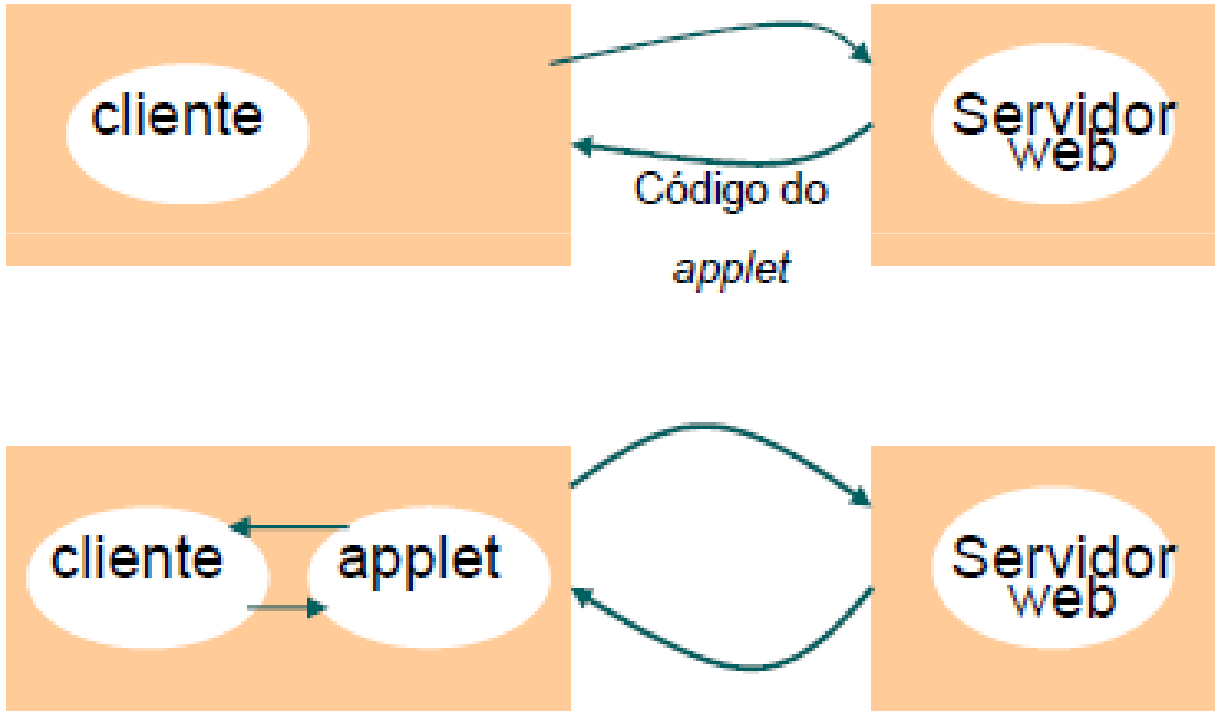
Modelo Cliente/Servidor – *Usando Cache e Servidor Proxy*



Modelo Cliente/Servidor – *Usando Cache e Servidor Proxy*

- *Cache* consiste em realizar um armazenamento de objetos de dados recentemente usados em um local mais próximo que a origem real dos objetos em si.
- Quando um processo cliente solicita um objeto, o serviço de cache primeiro verifica se possui armazenado uma cópia desse objeto.
- *Caches* podem ser mantidas para clientes ou estar localizadas em um servidor *proxy* – podendo ser compartilhadas por vários clientes.
- Assim, servidores *proxies* aumentam a disponibilidade e o desempenho do serviço, reduzindo a carga da rede e dos servidores web.

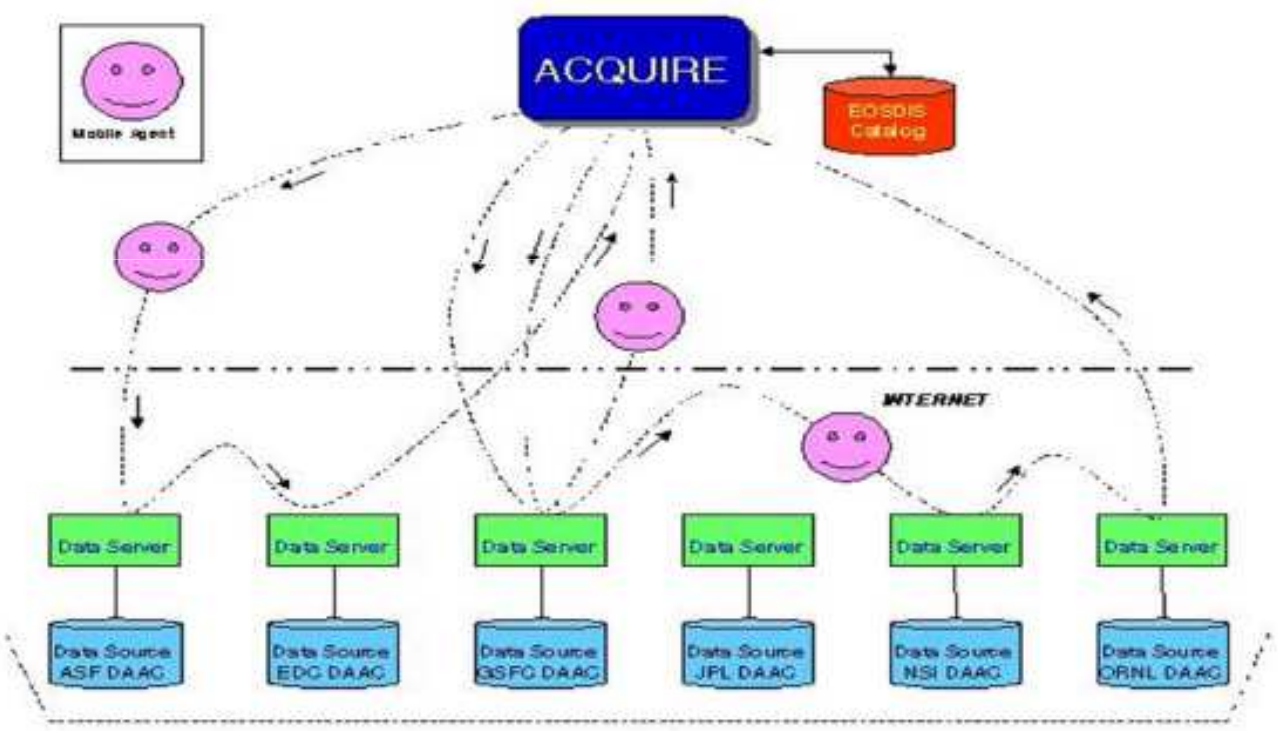
Modelo Cliente/Servidor – *Usando Código Móvel*



Modelo Cliente/Servidor – *Usando Código Móvel*

- Pedidos de clientes resultam em *downloads de applets.*
- O usuário, executando um navegador, seleciona um *link* que aponta para um *applet,* cujo código é armazenado em um servidor web. O código é carregado e, posteriormente, executado.
- Uma vantagem de executar localmente é que ele pode dar uma boa resposta interativa, pois não sofre os atrasos e nem a variação da largura de banda associada à comunicação da rede.
- O uso de código móvel é uma ameaça à segurança do computador destino. Uma solução é o navegador dar aos *applets* um acesso limitado aos seus recursos locais.

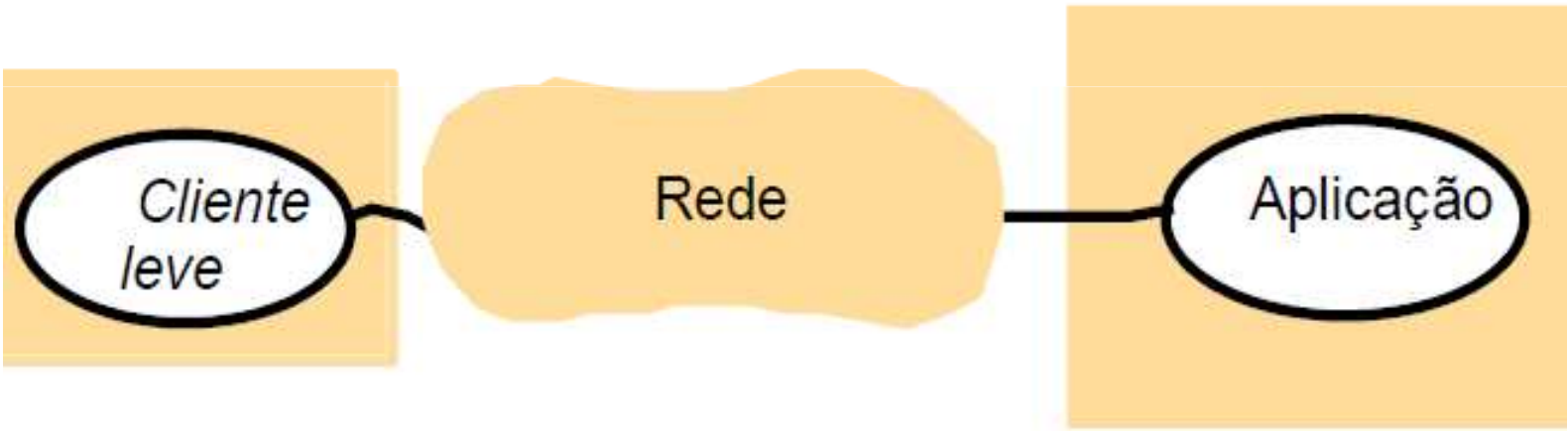
Modelo Cliente/Servidor – *Usando Código Móvel*



Modelo Cliente/Servidor – *Usando Código Móvel*

- Um agente móvel é um programa em execução (inclui código e dados) que migra entre máquinas, realizando uma tarefa em nome de alguém.
- Um agente móvel pode efetuar várias requisições aos recursos locais de cada *site* que visita (por exemplo, acessar banco de dados).
- Comparando esse modelo com o modelo que usa cliente estático que solicita, via requisições remotas, acesso a alguns recursos, há uma redução no custo e no tempo de comunicação.
- A identidade do agente deve ser incluída de maneira segura com o código e com os dados do agente móvel.

Modelo Cliente/Servidor - Uso de Clientes Leves (*Thin Clients*)



Modelo Cliente/Servidor - Uso de Clientes Leves (*Thin Clients*)

- O termo cliente “leve” refere-se a uma camada de software, em um computador local, que oferece ao usuário uma interface baseada em janelas para que este possa executar programas aplicativos em outro computador.
- Ao invés de fazer *download* do código de aplicativos no computador do usuário, ele os executa em um servidor de computação, que é um computador com capacidade suficiente para executar um grande número de aplicativos.
- O ruim dessa arquitetura está nas atividades gráficas altamente interativas, onde os atrasos sentidos pelos usuários aumentam em função da necessidade de transferir imagens entre o cliente leve e o processo aplicativo.

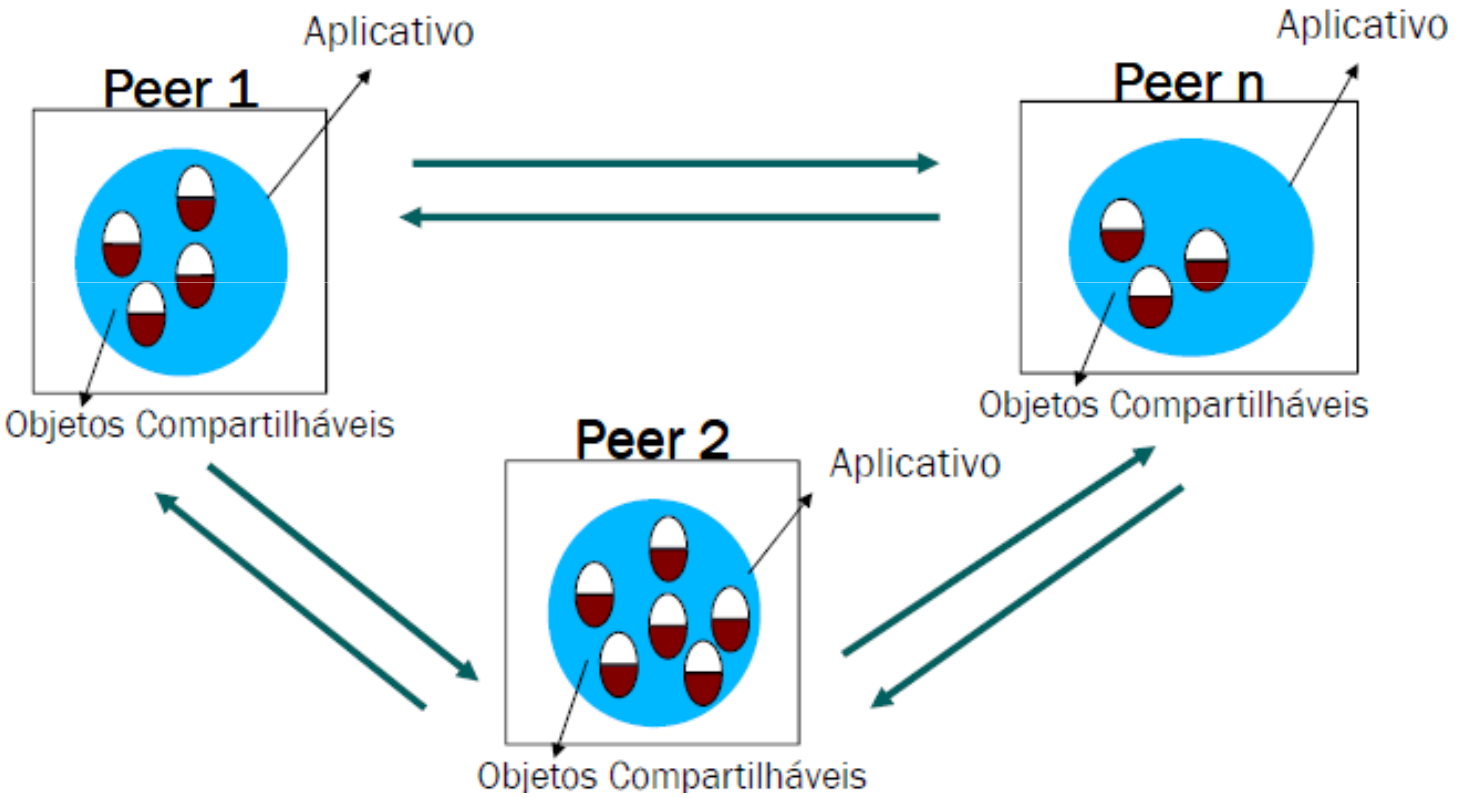
Modelo *Peer To Peer (P2P)*

- Todos os processos envolvidos em uma tarefa ou atividade desempenham funções semelhantes, interagindo cooperativamente como pares (*peers)*, sem distinção entre processos clientes e servidores.
- O objetivo é explorar os recursos de um grande número de computadores para a realização de uma dada tarefa ou atividade.
- Tem-se construído aplicativos que permitem que centenas de milhares de computadores possam acessar dados que eles gerenciam e armazenam coletivamente.
- Exemplo: Napster (compartilhamento de música digital).

Modelo *Peer To Peer (P2P)*

- Os aplicativos são compostos de grandes números de processos (*peers)* executados em distintos computadores e o padrão de comunicação entre eles depende inteiramente do que o aplicativo faz.
- Cada serviço pode ser replicado em vários computadores para distribuir melhor a carga, e para proporcionar maior robustez, no caso de desconexão de computadores individuais.
- A necessidade de ter objetos individuais, recuperá-los e manter réplicas entre muitos computadores torna essa arquitetura mais complexa do que a arquitetura anterior.

Modelo *Peer To Peer (P2P)*



Modelo *Peer To Peer (P2P)*

- Assim, arquiteturas *peer-to-peer* se desenvolvem em torno de como organizar os processos em uma **rede de sobreposição.**
- Rede na qual os nós são formados pelos processos e os enlaces representam os canais de comunicação possíveis.
- Existem dois tipos de redes de sobreposição:
  - Redes Estruturadas
  - Redes Não-estruturadas

P2P Estruturada

- Neste caso a rede de sobreposição é construída com a utilização de um procedimento determinístico.
- Por exemplo, os nós estão logicamente organizados em um anel de modo tal que um item de dado com chave *k* seja mapeado para o nó que tenha o menor identificador id ≥ *k.*
- Assim, a aplicação pode consultar diretamente o nó para obter uma cópia do item de dado.

P2P Estruturada

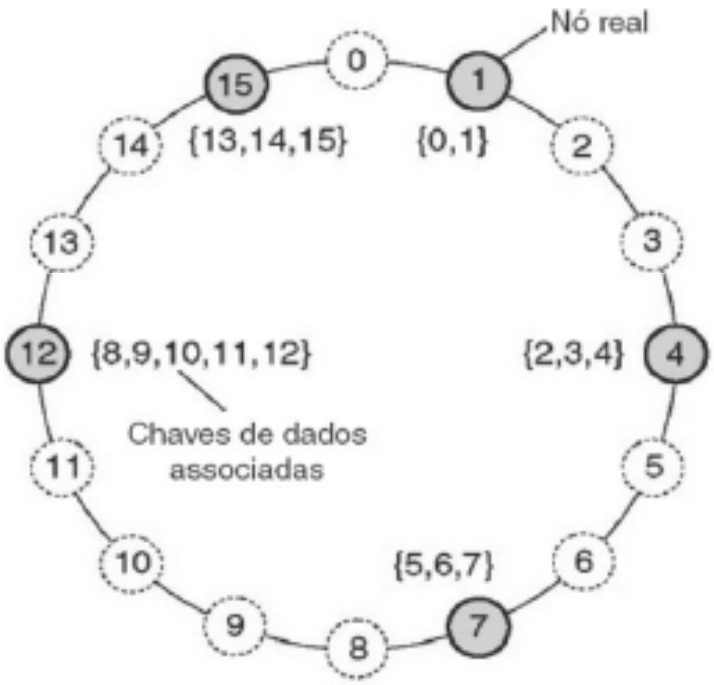


Figura 2.7 Mapeamento de itens de dados para nós em Chord.

P2P Não-estruturada

- Sistemas *peer-to-peer* não-estruturados dependem, em grande parte, de algoritmos aleatórios para construir uma rede de sobreposição.
- A idéia principal é que cada nó mantenha uma lista de vizinhos (visão parcial), mas que essa lista seja construída de modo mais ou menos aleatório.
- Da mesma maneira, admite-se que itens de dados sejam colocados aleatoriamente em nós.
- Para encontrar dados, é preciso inundar a rede (no pior caso) com consultas de busca.

P2P Não-estruturada

- Um das estratégias é construir uma rede aleatória.
  - O modelo básico é que cada nó mantenha uma lista de *c* vizinhos, na qual seja dado preferência.
  - Para atualizar os vizinhos:
    - *Threads* solicitam aos vizinhos a visão parcial (pull) ou empurram (*push)* a visão a seus vizinhos.
    - Algoritmos que atualizem a vizinhança a cada *x* unidades de informação enviadas.

P2P Não-estruturada

- Como encontrar os dados de maneira eficiente?
  - Muitos sistemas utilizam nós especiais, que possuem um índice de itens de dados, os chamados *Superpeers*.
  - Resultando em uma estrutura hierárquica.

Fontes Bibliográficas

G. Coulouris, J. Dollimore, T. Kindberg. *Sistemas Distribuídos – Conceitos e Projetos*. 4○ edição. Bookman, 2007.

A. Tanenbaum, M. Steen. Sistemas Distribuídos – Princípios e Paradigmas. 2○ edição. Prentice Hall, 2007.