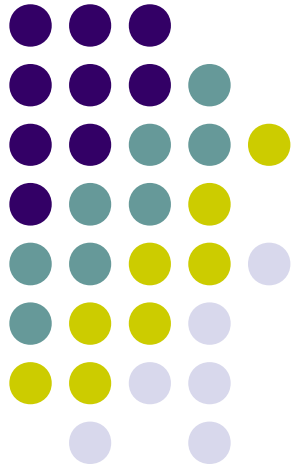
Sistemas Distribuídos

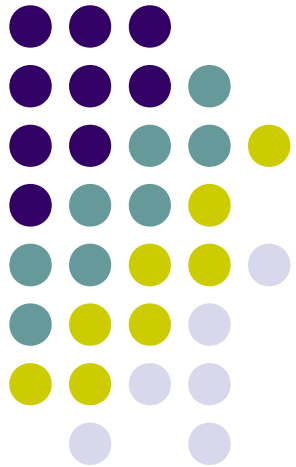
Professora: Ana Paula Couto

DCC 064

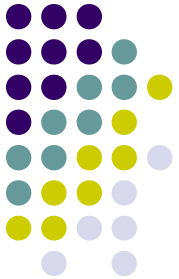


Introdução

Capítulo 1

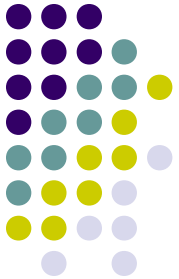


Definição



“ Um sistema distribuído é um conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente” – Tanenbaum/Van Steen

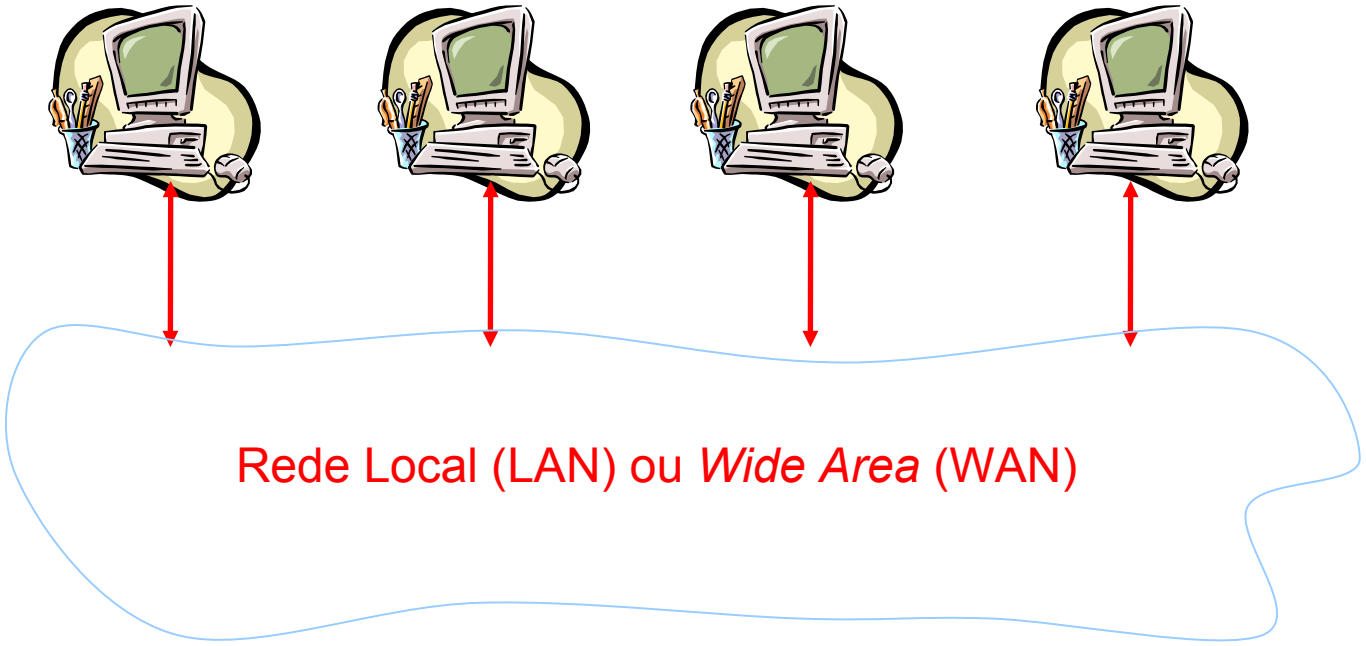
Definição



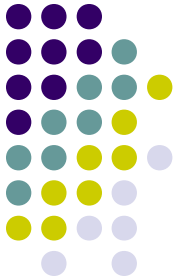
“ Coleção de computadores autônomos interconectados por uma rede, com software projetado para produzir uma aplicação integrada”

Definição

Computadores pessoais, estações de trabalho, servidores, etc

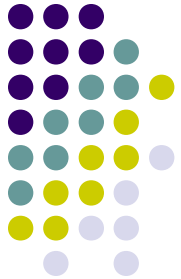


Definição



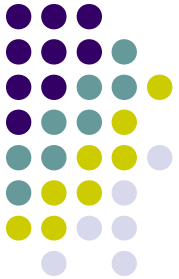
“Você sabe que existe um sistema distribuído quando a falha de um computador que você nunca ouviu falar impede que você faça qualquer trabalho” - Leslie Lamport

Exemplos



- Aplicações comerciais (reservas de bilhetes, bancos)
- Aplicações Internet (*WWW*)
- Aplicações de acesso a informações multimídia (Áudio (voz) e vídeo conferência, P2P-TV)
- Groupware (trabalho cooperativo)

Middleware



- Como suportar computadores e redes heterogêneos, oferecendo uma visão de sistema único?
- SDs são organizados por meio de uma camada de software

Middleware

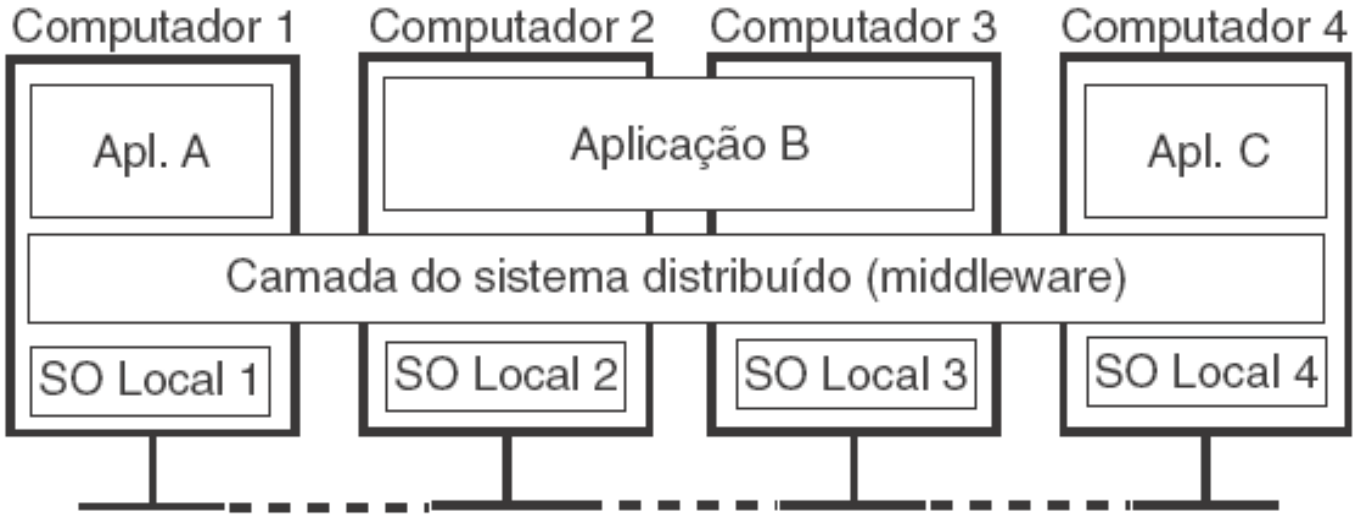
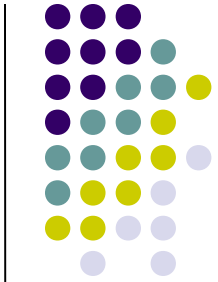


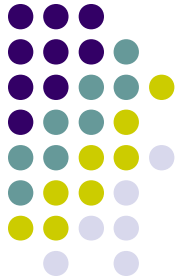
Figura 1.1 Sistema distribuído organizado como middleware. A camada de middleware se estende por várias máquinas e oferece a mesma interface a cada aplicação.

Metas



- Acesso a recursos
- Transparência
- Abertura
- Escalabilidade

Meta 1:
Acesso a Recursos



- Facilitar aos usuários e aplicações acesso a recursos remotos e o compartilhamento de maneira controlada e eficiente
- Razão óbvia: Economia
- Impressoras, computadores, dados, página Web
- Conectividade → Groupware e comércio eletrônico

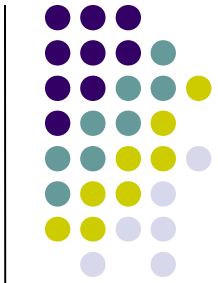
Meta 1:
Acesso a Recursos

- Problema: Segurança
- Senhas, autenticação de usuários
- Rastreamento de comunicações para

montar um perfil de preferências → violação de privacidade

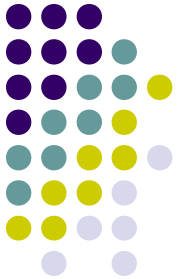
- Spam

Meta 2: Transparência



- Ocultar o fato de que seus processos e recursos estão fisicamente distribuídos por vários computadores

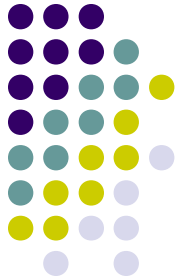
Meta 2:
Tipos de Transparência



Transparência	Descrição
Acesso	Oculta diferenças na representação de dados e no modo de acesso a um recurso
Localização	Oculta o lugar em que um recurso está localizado
Migração	Oculta que um recurso pode ser movido para outra localização
Relocação	Oculta que um recurso pode ser movido para uma outra localização enquanto em uso
Replicação	Oculta que um recurso é replicado
Concorrência	Oculta que um recurso pode ser compartilhado por diversos usuários concorrentes
Falha	Oculta a falha e a recuperação de um recurso

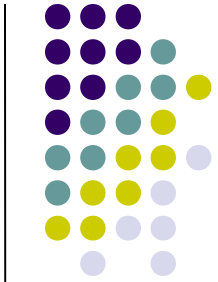
Tabela 1.1 *Diferentes formas de transparência em um sistema distribuído (ISO, 1995).*

Transparência - Acesso



- Ocultar diferenças entre arquiteturas de máquinas
- Mais importante: Acordo sobre como os dados devem ser representados
- Exemplo:Nomeação de arquivos em SOs diferentes

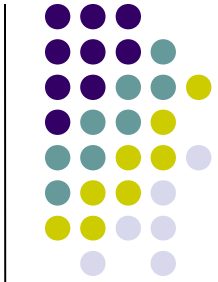
Transparência - Localização



- Usuários não são capazes de dizer a localização física do recurso
- Nomeação
- www.google.com → nome não dá pistas da localização física de um dos servidores google

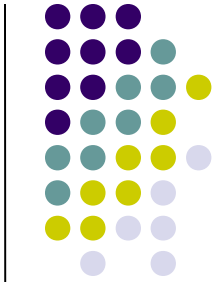
(Não vale adotar o comando traceroute ! :))

Transparência - Migração



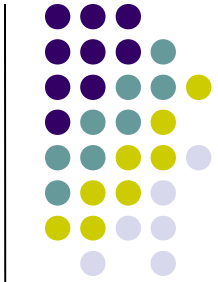
- Recursos podem migrar de uma localidade para outra, por questões de desempenho, segurança, etc
- Deve ser feita de forma automática pelo sistema
- Deve manter o nome do objeto
- Deve garantir a continuidade de comunicação

Transparência - Relocação



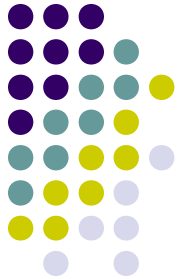
- Oculta que um recurso possa ser movido para outra localização durante o uso
- Exemplos
  - Celular se movimentando dentro da mesma área de cobertura
  - Um automóvel passando por várias redes de acesso sem fiio, com conexão ininterrupta

Transparência - Replicação



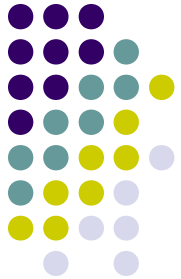
- Permite que várias instâncias de recursos sejam usadas para aumentar a confiabilidade e o desempenho
- Deve mascarar o conhecimento das réplicas por parte dos usuários
- Implica na transparência de localização
- Problemas de consistência

Transparência - Concorrência



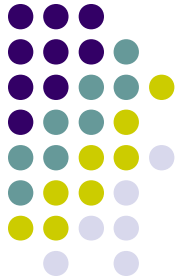
- Compartilhamento competitivo de recursos
- Deve garantir consistência
- Travas de acesso
- Tratamento mais refinado: transações

Transparência - Falhas



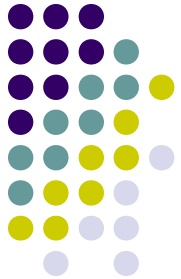
- Usuário não pode perceber que um recurso deixou de funcionar bem
- Mascarar falhas é uma das questões mais difícies
- Recurso morto ou incrivelmente lento?

Transparência - Falhas



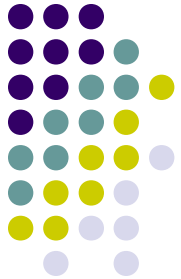
- Usuário não pode perceber que um recurso deixou de funcionar bem
- Mascarar falhas é uma das questões mais difícies
- Recurso morto ou incrivelmente lento?

Transparência – Sempre requerida?



- Compromisso entre um alto grau de transparência e o desempenho do sistema
  - Exemplo: Aplicações de Internet tentam contatar um servidor repetidas vezes antes de desistir. Talvez seja melhor desistir mais cedo ou permetir que o usuário cancele as tentativas
- Sistemas embutidos: laptop e impressora local

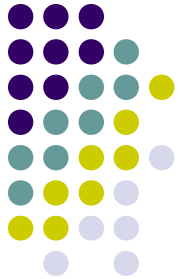
Meta 3: Abertura



- Característica que determina se um sistema pode ser estendido de diferentes maneiras
- Hardware

  - Inclusão de dispositivos de fabricantes distintos
- Software
  - Módulos de SO
  - Protocolos de Comunicação
  - Recursos compartilhados

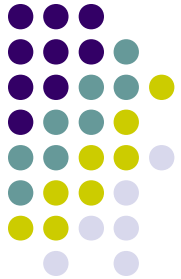
Meta 3: Abertura



- Interoperabilidade

"Define até que ponto duas implementações de sistemas ou componentes de fornecedores diferentes devem coexistir e trabalhar em conjunto, com base na confiança mútua nos serviços de cada um, especificados por um padrão comum"

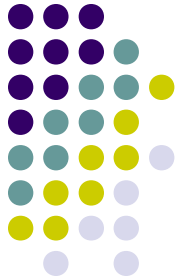
Meta 3: Abertura



- Interoperabilidade

"Capacidade de um sistema (informatizado ou não) de se comunicar de forma transparente (ou o mais próximo disso) com outro sistema (semelhante ou não)."

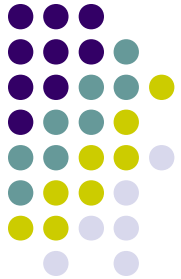
Meta 3: Abertura



- Portabilidade

"Caracteriza até que ponto uma aplicação desenvolvida para um sistema distribuído A pode ser executada, SEM MODIFICAÇÃO, em um sistema distribuído B que implementa as mesmas interfaces que A"

Meta 3: Abertura

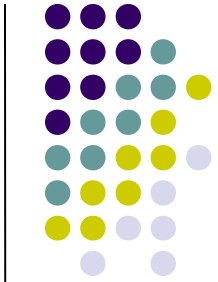


- Portabilidade

"Habilidade de reusar um código existente ao invés de refazê-lo quando este é movido de um ambiente para outro"

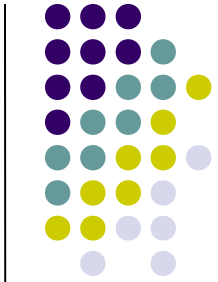
IMPORTANTE: PADRONIZAÇÃO!!!!!

Meta 4: Escalabilidade



- Três Dimensões [Neuman, 1994]
  - Tamanho
  - Termos Geográficos
  - Termos Administrativos

Escalabilidade - Tamanho

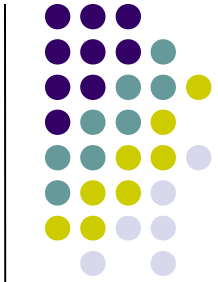


- Aumento do número de usuários e/ou processos
- PROBLEMAS

Conceito	Exemplo
Serviços centralizados	Um único servidor para todos os usuários
Dados centralizados	Uma única lista telefônica on-line
Algoritmos centralizados	Fazer roteamento com base em informações completas

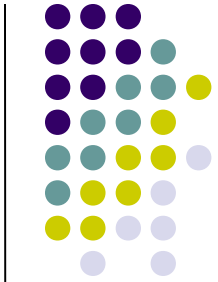
Tabela 1.2 Exemplos de limitações de escalabilidade.

Escalabilidade - Tamanho



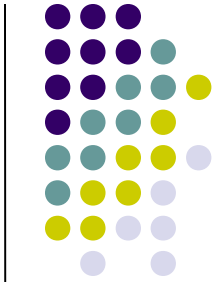
- Serviços Centralizados
  - Serviços que são implementados por meio de apenas um único servidor que executa em uma máquina específica no sistemma distribuído
  - Possível Gargalho no sistema

Escalabilidade - Tamanho



- Dados Centralizados
  - Números telefônicos estivessem em um único banco de dados → saturação de todas as linhas de comunicação que o acessam
  - DNS

Escalabilidade - Tamanho

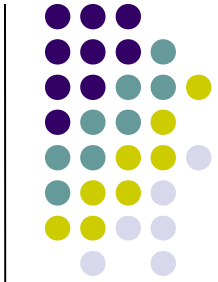


- Algoritmos Centralizados
  - Para evitar troca de mensagens → colher

informações de todas as máquinas e linhas e executar m algoritmo para computar todas as rotas ótimas → propagar as informações por todo o sistema

  - Péssima idéia!!

Escalabilidade - Geográfica



- Usuários e/ou recursos podem estar longe um dos outros
- PROBLEMAS
  - Dificuldade de ampliar sistemas distribuidos existentes que foram originalmente projetados para redes locais: COMUNICAÇÃO SÍNCRONA
  - Comunicação em redes de longa distância é inerentemente não confiável, ponto-a-ponto
    - Localização de serviços

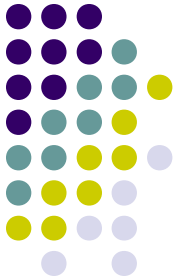
Escalabilidade - Administrativa

- Sistema pode ser fácil de gerenciar, mesmo que abranja muitas organizações diferentes
- PROBLEMAS
  - Políticas conflitantes em relação a utilização – e pagamento – de recursos, gerencialmente e segurança

Escalabilidade - Administrativa

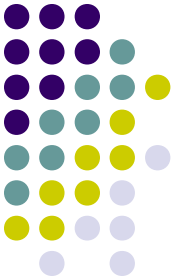
- Os usuários de um único domínio podem confiar em componentes de um sistema distribuído que residam dentro desse mesmo domínio
- Confiança não ultrapassa as fronteiras do domínio: a administração do sistema deve testar e certificar aplicações e tomar providências especiais para garantir que os componentes não sofram nenhuma ação indevida

Técnicas de Escalabilidade



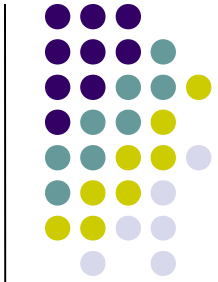
- Três técnicas [Neuman 1994]
  - Ocultar latências de comunicação
  - Distribuição
  - Replicação

Técnicas de Escalabilidade
Ocultar Latências



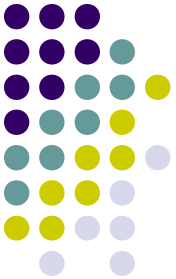
- Escalabilidade Geográfica
- Evitar esperar por respostas a requisições remotas
- Comunicação Assíncrona

Técnicas de Escalabilidade
Ocultar Latências



- Aplicações Interativas devem esperar por uma resposta
- Solução: Reduzir comunicação global, passando parte da computação do servidor para o cliente que está requerendo o serviço

Técnicas de Escalabilidade
Ocultar Latências



- Exemplo: Acesso a banco de dados por meio de formulários

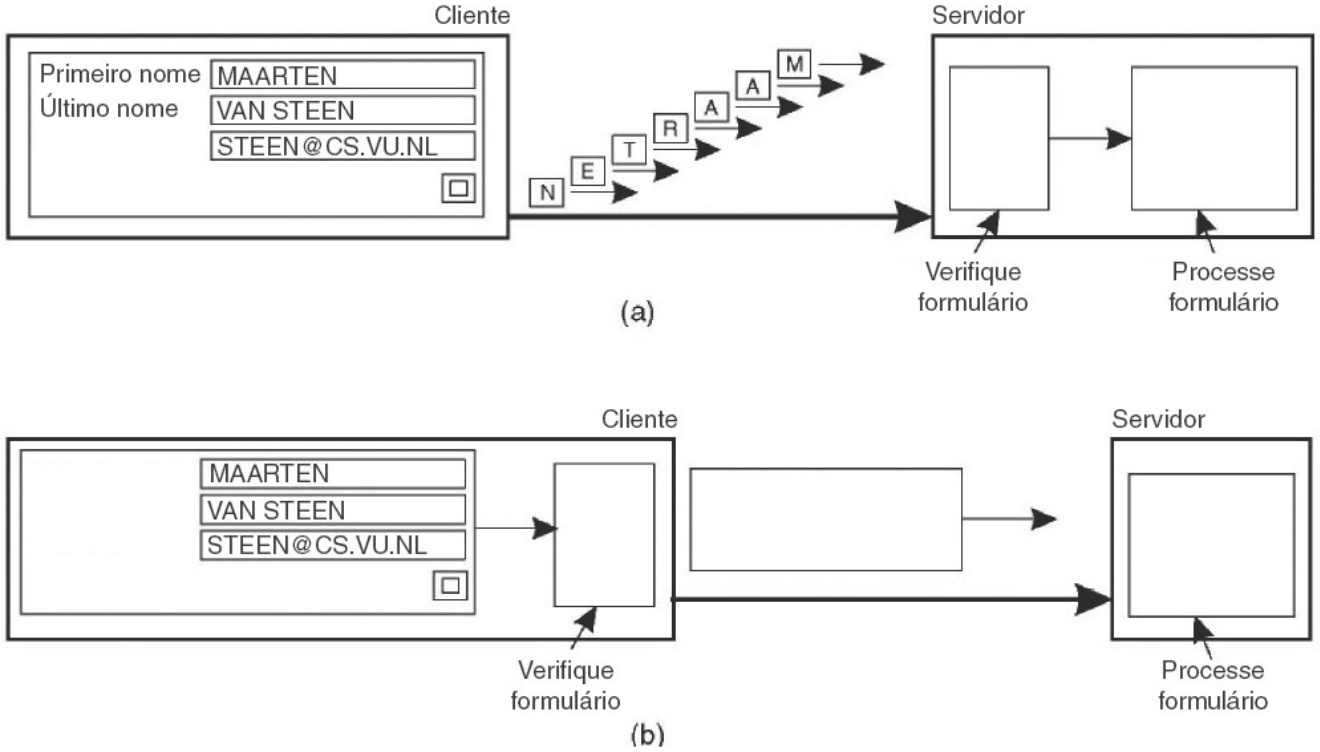
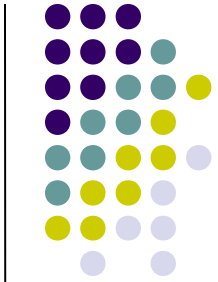


Figura 1.2 A diferença entre deixar (a) um servidor ou (b) um cliente verificar formulários à medida que são preenchidos.

Técnicas de Escalabilidadea Distribuição



- Escalabilidade de Tamanho

- Dividir um componente em partes menos e espalhar as sub-partes pelo sistema

Técnicas de Escalabilidade
Distribuição

Exemplo: DNS: hierarquia em árvore de domínios, dividida em zonas se sobreposição

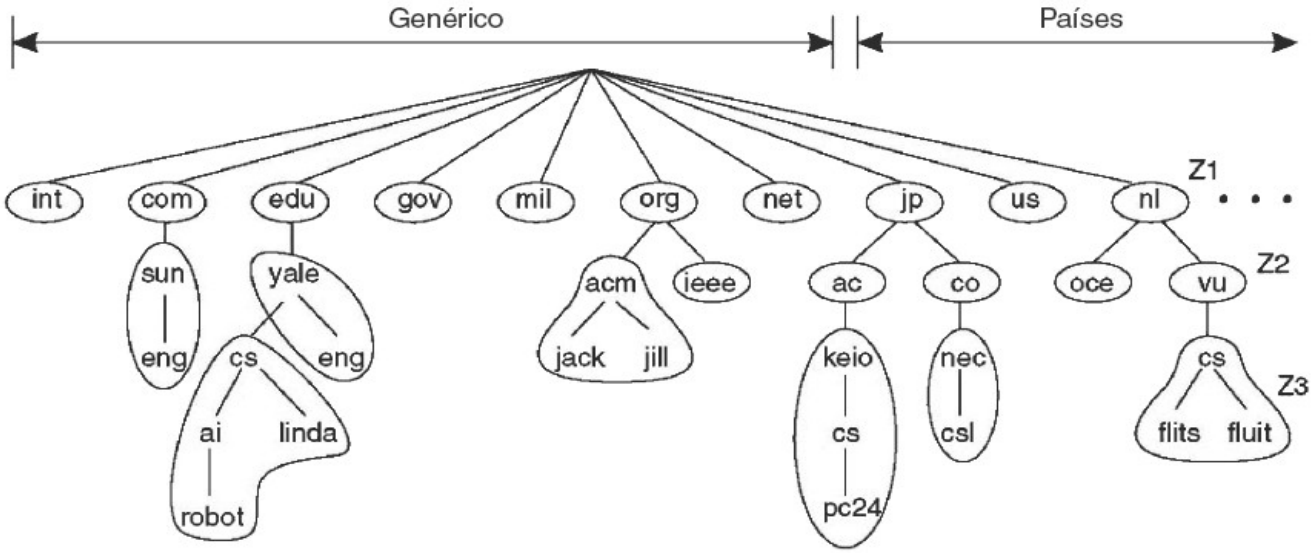
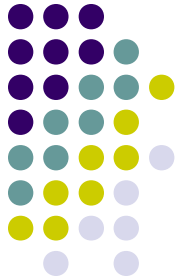


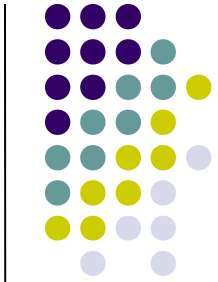
Figura 1.3 Exemplo de divisão do espaço de nomes do DNS em zonas.

Técnicas de Escalabilidade Replicação



- Aumenta a disponibilidade dos recursos
- Equilibra a carga entre os componentes
- Sistemas com ampla dispersão geográfica ->ocultar os problemas de latência
- Cache
  - Forma especial de replicacão
  - Cache é uma decisão do cliente do sistema

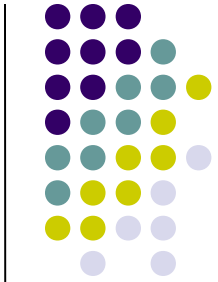
E a Escalabilidade Administrativa?



- Problemas políticos estão envolvidos
- Progresso na área: ignorar domínios adminstrativos
- P2P – usuários finais tomam o controle
  - Vários problemas de direitos autorais,

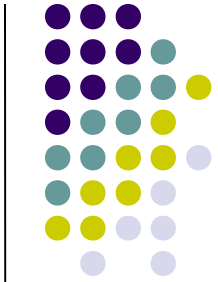
sobrecarga de informação nos Sistemas Autônomos

Ciladas



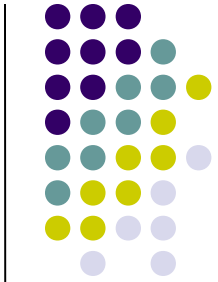
- Premissas falsas adotadas ao se desenvolver pela primeira vez uma aplicação distribuída
  - Rede é confiável
  - Rede é segura
  - Rede é homogênea
  - Topologia constante
  - Latência zero
  - Largura de banda é infinita
  - Custo de Transporte é zero
  - Existe somente um administrador

Tipos de Sistemas Distribuídos



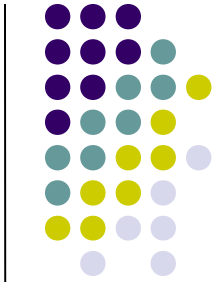
- Sistemas de Computacão
- Sistemas de Informação
- Sistemas Pervasivos

Sistemas de Computação



- Computação de Cluster
- Computação em Grade

Sistemas de Computação - Cluster



- Hardware consiste em um conjunto de estações de trabalho ou Pcs semelhantes
- Conexão feita através de uma rede local
- Em quase todos os casos, a computação de cluster é usada para programação paralela na qual um único programa é executado em paralelo

Sistemas de Computação - Cluster

- Clusters Beowulf baseados em Linux

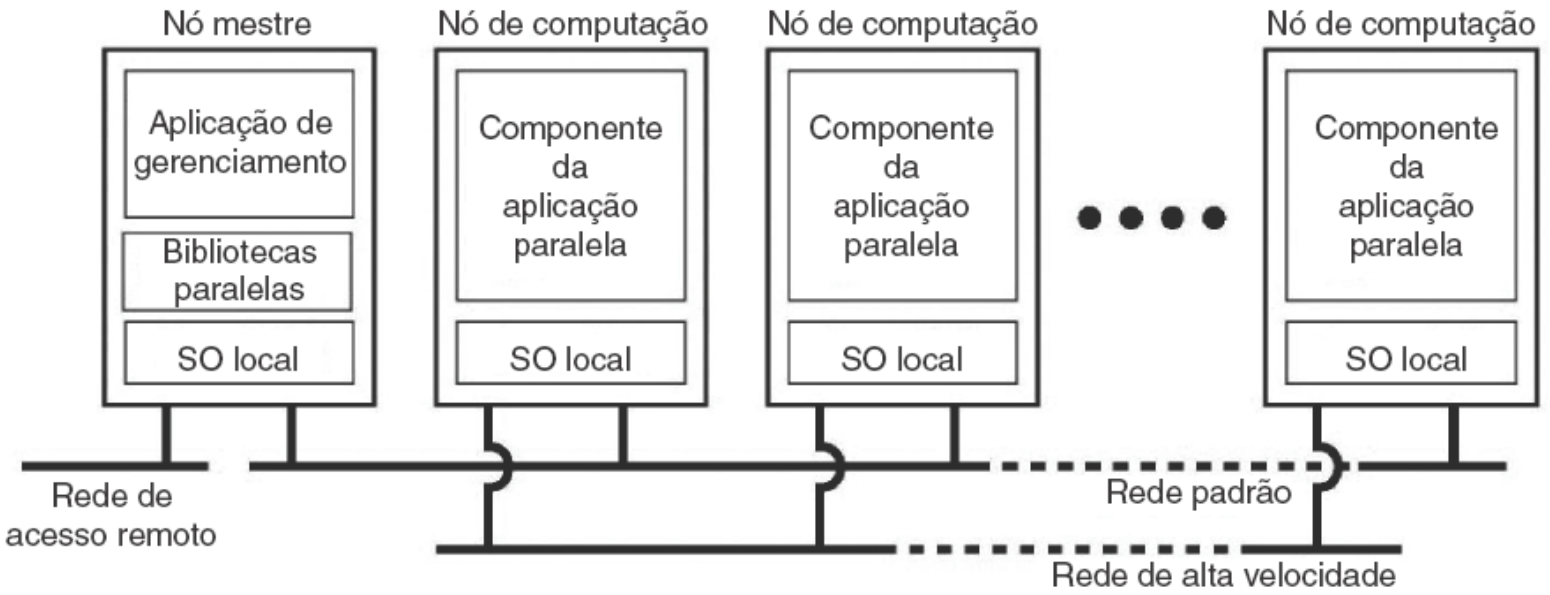
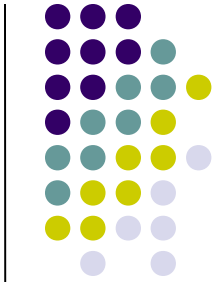


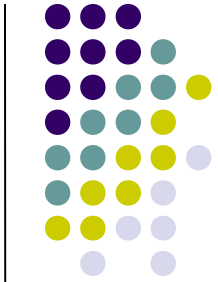
Figura 1.4 Exemplo de um sistema de computação de cluster.

Sistemas de Computação - Grade



- Heterogeneidade
- Recursos de diferentes organizações são reunidos para permitir a colaboração de um grupo de pessoas ou instituições
- PlanetLab: http://www.planet-lab.org

Sistemas de Informação



- Sistemas empresariais desenvolvidos para integrar diversas aplicações individuais, onde a interoperabilidade se mostrou “dolorosa”
  - Sistemas de processamento de Transações
  - Integração de Aplicações Empresariais

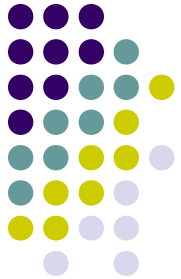
Sistemas de Informação - Processamento de Transações

- Requer primitivas especiais que devem ser fornecidas pelo sistema distribuído ou pelo sistema de linguagem

Primitiva	Descrição
BEGIN_TRANSACTION	Marque o início de uma transação
END_TRANSACTION	Termine a transação e tente comprometê-la
ABORT_TRANSACTION	Elimine a transação e restaure os valores antigos
READ	Leia dados de um arquivo, tabela ou de outra forma
WRITE	Escreva dados para um arquivo, tabela ou de outra forma

Tabela 1.3 Exemplos de primitivas para transações.

Sistemas de Informação - Processamento de Transações



- Características
  - Atômicas: para o mundo exterior, indivisível
  - Consistentes: não viola invariantes de sistema
  - Isoladas: transações concorrentes não interferem umas com as outras
  - Duráveis: uma vez comprometida uma transação, as alterações são permanentes

Sistemas de Informação - Processamento de Transações

- Transação Aninhada
  - Transação é construída com base em uma quantidade de subtransações

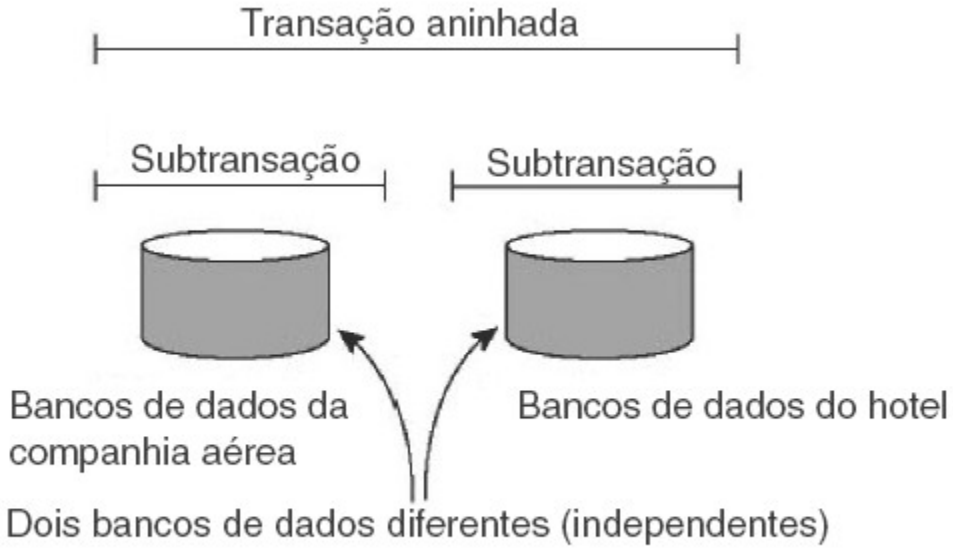
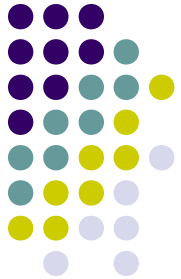


Figura 1.6 Transação aninhada.

Sistemas de Informação - Processamento de Transações



- No começo, o componente que manipulava transações distribuídas, ou aninhadas, formava o núcleo para integração de aplicações no nível do servidor ou do banco de dados
- Monitor de processamento de transação: permitir que uma aplicação acessasse vários servidores/bancos de dados

Sistemas de Informação - Processamento de Transações

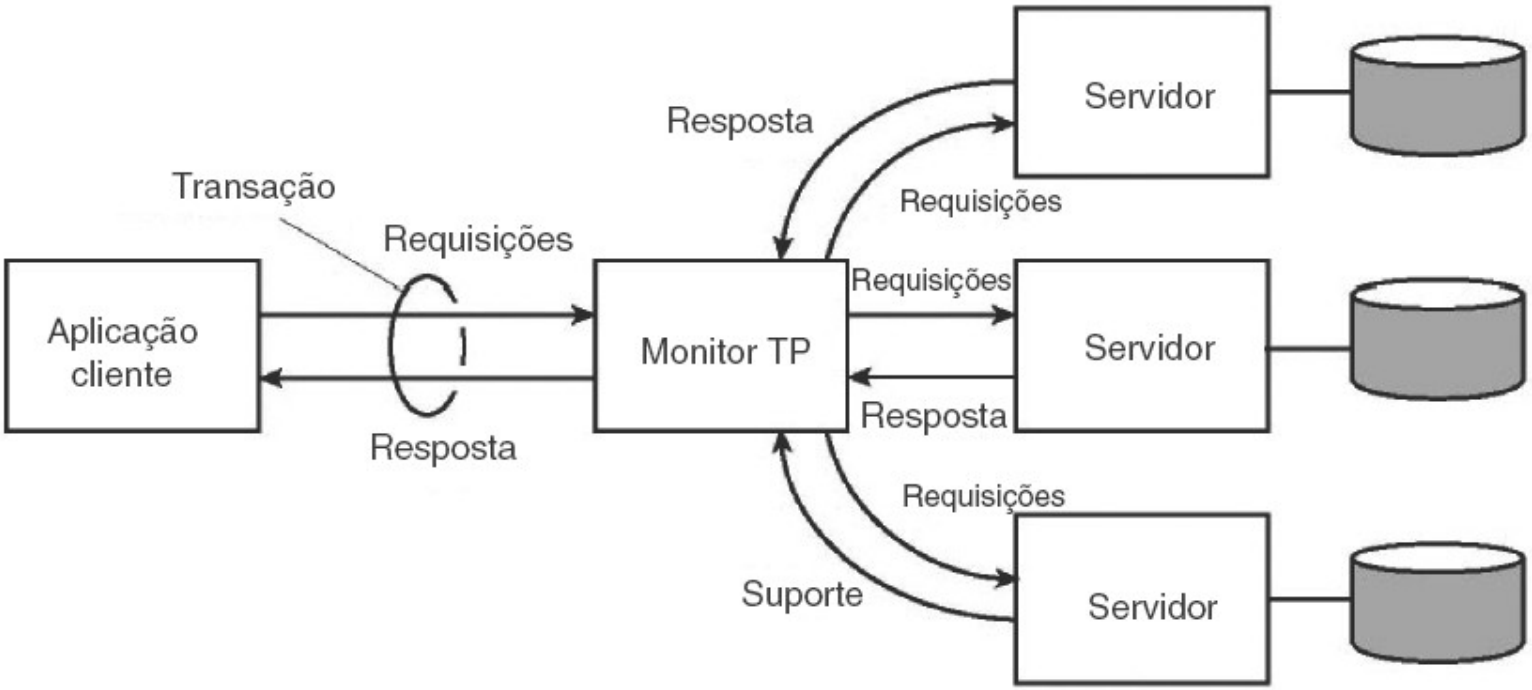
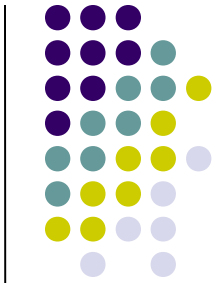


Figura 1.7 O papel do monitor TP em sistemas distribuídos.

Sistemas de Informação - Integração de Aplicações Empresariais



- Aplicações querem muito mais em termos de comunicação, não somente modelo de requisição/resposta
- Middleware de Comunicação
  - Chamadas de Procedimento Remoto
  - Invocações de Método Remoto
  - Middleware Orientado a Mensagem

Sistemas de Informação - Integração de Aplicações Empresariais

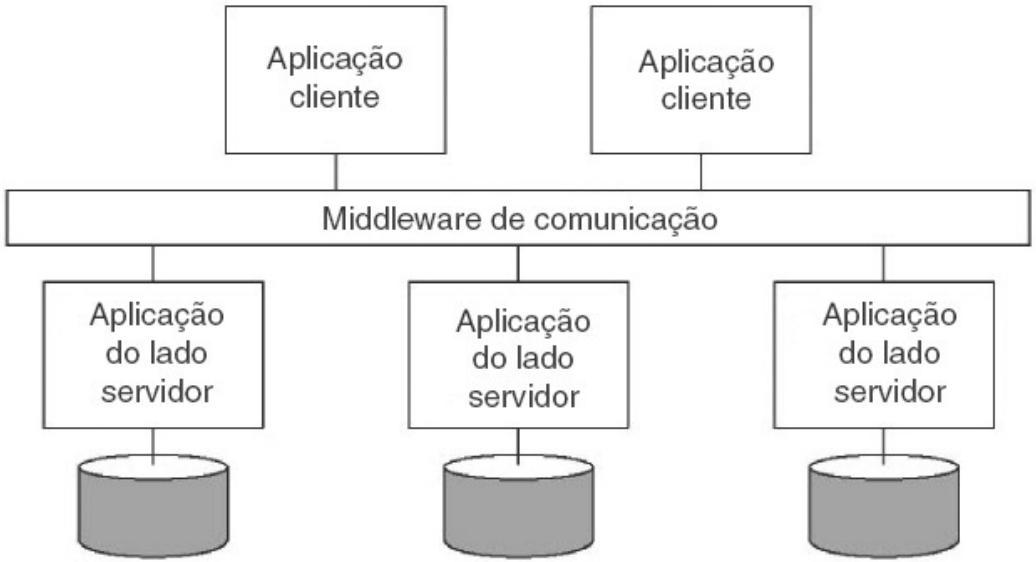
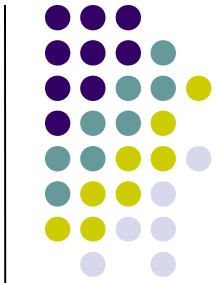


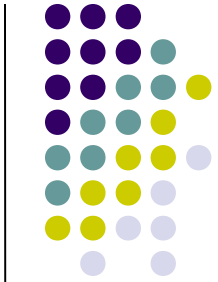
Figura 1.8 Middleware como facilitador de comunicação em integração de aplicações empresariais.

Sistemas de Informação - Middleware de Comunicação



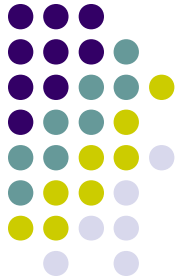
- Chamadas de Procedimento Remoto (RPC)
  - Componente de aplicação pode enviar a um outro componente de aplicação
  - Requisição e Resposta são empacotadas em mensagens

Sistemas de Informação - Middleware de Comunicação



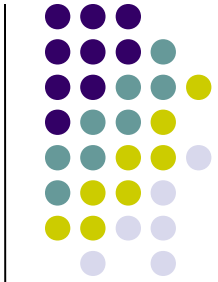
- Invocações de Método Remoto (RMI)
  - Popularidade da Tecnologia de Objetos
  - RMI semelhante a RPC, exceto que funciona com objetos em vez de com aplicações

Sistemas de Informação - Middleware de Comunicação



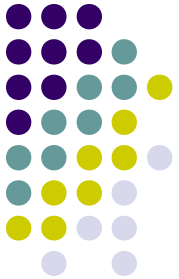
- Desvantagens do RPC e RMI:
  - Componentes da comunicação devem estar ligados e em funcionamento
  - Precisam saber exatamente como se referir um ao outro
- Middleware Orientado a Mensagem (MOM)
  - Aplicações enviam mensagens a pontos lógicos de contato
  - O Middleware se encarrega de entregar todas as mensagens destinadas a uma aplicação

Sistemas Pervasivos



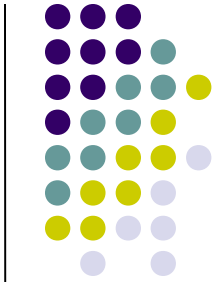
- Instabilidade é o comportamento esperado destes sistemas
- Dispositivos de computação móveis e embutidos
  - Pequenos
  - Alimentação por bateria
  - Mobilidade
  - Conexão sem fio

Sistemas Pervasivos



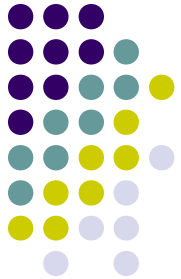
- Parte do nosso entorno
- Ausência geral de controle administrativo humano
- Requisitos para as aplicações pervarsivas:
  - Adotar mudanças contextuais
  - Incentivar composição ad hoc
  - Reconhecer compartilhamento como padrão

Tipos de Sistemas Pervasivos



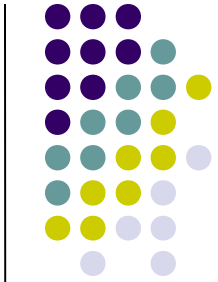
- Sistemas Domésticos
- Sistemas Eletrônicos para Tratamento de Saúde
- Redes de Sensores

Questões



- A transparência de distribuição pode não estar presente em sistemas pervasivos. Essa declaração não vale para todos os tipos de transparências. Exemplo.
- Por que nem sempre é uma boa idéia visar a implementação do mais alto grau de transparência possível?

Próxima Aula



- Arquitetura
  - Estilos Arquitetônicos (software)
  - Arquiteturas de Sistemas (local físico das 'peças' de software)