Programação Distribuída

Parte 1

Roteiro

- Introdução
- Definição
- Aspectos Relevantes

SD (Sistemas Distribuídos) – Introdução

- De 1945 até 1985, computadores grandes e caros, destinados a umas poucas organizações.
- A partir da década de 80 surgem os microcomputadores de 8, 16, 32 e 64 bits.
- Surgem processadores equivalentes a capacidade computacional de mainframes, porém com custo menos
- Em 50 anos uma máquina que custava dez milhões de dólares (uma instrução por segundo), passou a custar mil dólares (um bilhão de instruções por segundo).
- Se um Rolls Royce tivesse tido evolução equivalente, atualmente esse carro custaria um dólar e faria um bilhão de quilômetros por litro, mas para abrir a porta seria necessário ler um manual de muitas páginas.

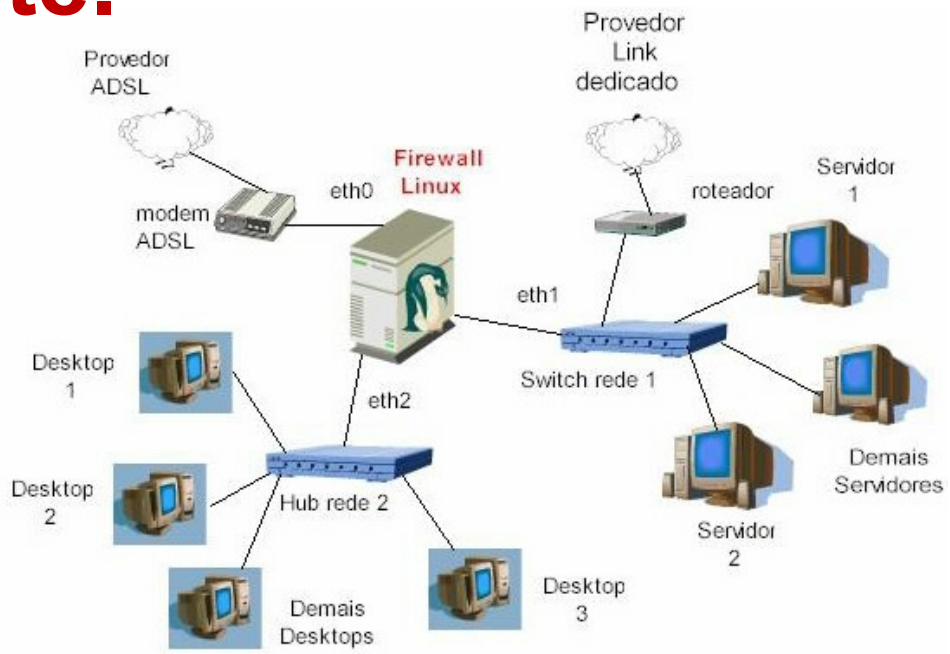
Introdução

- As redes de computadores tornaram-se acessíveis e se popularizaram a partir da década de 80, permitindo que centenas de máquinas se conectassem para informações fossem transferidas em microssegundos.

- O cenário atual permite troca de informações com velocidades altíssimas, como de 64 KBits/s a gigabits por segundo.

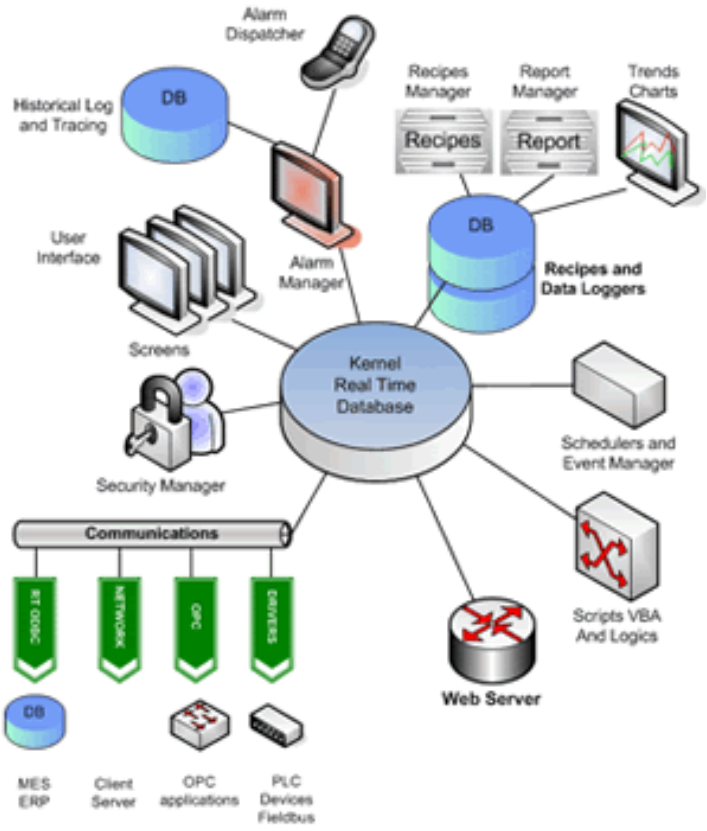
Definição de Sistemas Distribuídos

Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente.



Aspectos Importantes dos SD

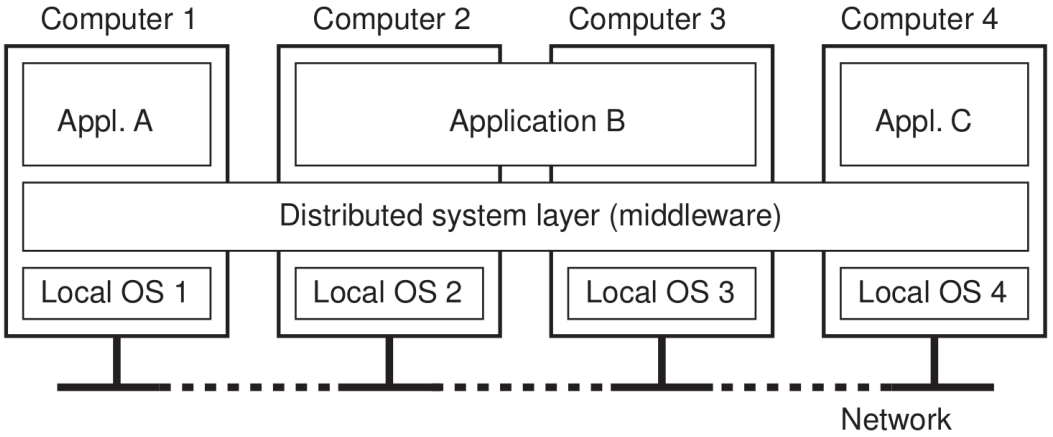
- Um sistema distribuído consiste em componentes (computadores) autônomos.
- Os usuários (pessoas ou programas) têm a percepção que estão trabalhando com um único sistema.
- Estabelecer colaboração entre componentes autônomos é o cerne do desenvolvimento de SD



Aspectos Importantes dos SD

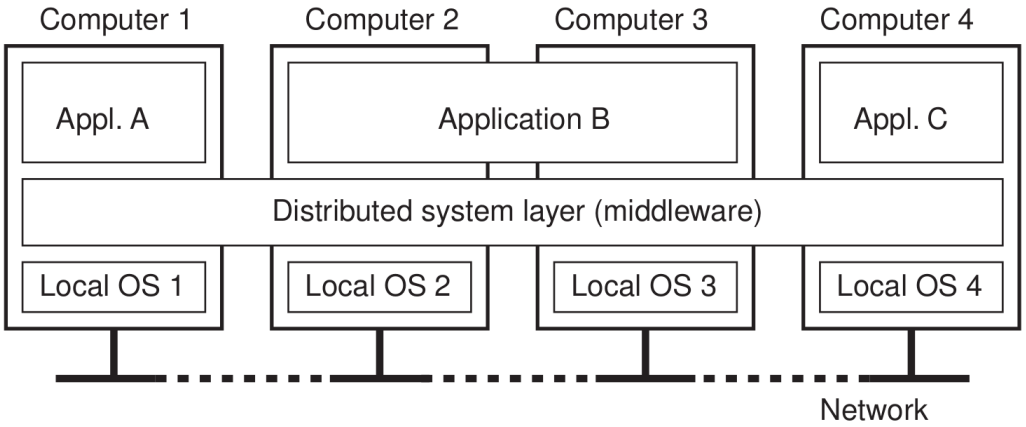
- Quaisquer usuários ou aplicações podem interagir com um sistema de maneira consistente e uniforme, independentemente de onde e como a interação ocorra.
- Um sistema distribuído, em geral, está sempre disponível mesmo que uma ou outra parte esteja temporariamente avariada.
- Usuários não devem perceber quais partes estão sendo substituídas ou reparadas.

Aspectos Importantes dos SD



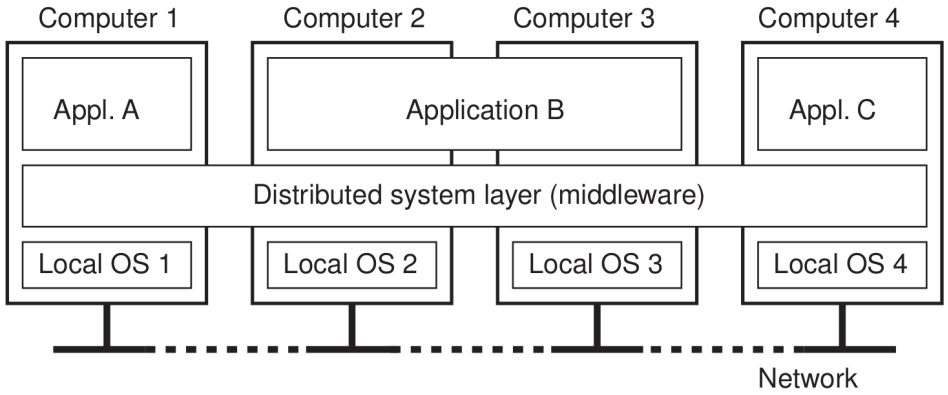
- Para suportar redes heterogêneas formada por todo tipo de computadores oferecendo sempre uma visão unificada.
- Os sistemas distribuídos costumam ser organizados por meio de uma camada de software situada logicamente entre o nível mais alto, composta por usuários e aplicações, e uma camada mais baixa, que consiste de sistemas operacionais e facilidades básicas de comunicação.
- Muitas vezes esses sistemas são chamados de **middleware**, como mostrado na figura desse slide.

Aspectos Importantes dos SD



- Na figura é mostrada quatro computadores em rede e três aplicações, das quais a aplicação “B” é distribuída entre os computadores 2 e 3.
- Uma mesma interface é oferecida a cada aplicação.
- O sistema distribuído proporciona os meios para que os componentes de uma única aplicação distribuída se comuniquem uns com os outros, mas também permite que diferentes aplicações se comuniquem.
- Ao mesmo tempo, ele oculta, do melhor e mais razoável modo possível, as diferenças em hardware e sistemas operacionais para cada aplicação.

Aspectos Importantes dos SD



- A principal facilidade proporcionada por um sistema distribuído é facilitar aos usuários e aplicações, o acesso a recursos remotos e seu compartilhamento de maneira controlada e eficiente.
- A conexão de usuários e recursos, facilita a colaboração e a troca de informações, o que é claramente ilustrado pelo sucesso da Internet com seus protocolos simples para trocar arquivos, correio, documentos, áudio e vídeo.
- Assim pessoas dispersas geograficamente podem trabalhar juntas por meio de **groupware**, que são softwares para edição colaborativa, teleconferência, entre outras funções.

Segurança

- Quanto maior o número de pessoas trabalhando conectadas e compartilhando trabalhos, mais a segurança se torna importante.

- Os sistemas atuais não foram concebidos para se protegerem de pessoas com conhecimentos na construção de softwares, mas apenas de usuários desatentos ou mal intencionados, todavia sem condições de causarem maiores estragos por não terem conhecimentos técnicos.

Segurança

- Senhas são enviadas em texto comum (não criptografadas) pela rede e armazenadas em locais certamente não plenamente confiáveis.

- Softwares mais antigos não foram concebidos para serem processados fora do ambiente interno das empresas e, não raro, quando usados de fora da empresa apresentam toda sorte de vulnerabilidades.

Segurança

- Finalmente, o rastreamento de comunicações pode traçar o perfil de preferências de um usuário em especial, algo que viola explicitamente a privacidade, em especial quando isso é feito sem o prévio conhecimento do usuário.

- A inclusão de softwares de monitoramento ou arquivos de registro sobre preferências pessoais do usuário como *cookies*, tanto podem ser positivas para aquele usuário que deseja obter informações de acordo com sua preferência, como podem ser invasivas.

- Nada (*a não ser a ética*) impede que alguém comercialize informações pessoais obtidas por meio de tecnologia não autorizada.

Transparência - Tipos

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

- Segundo Leslie Lamport **“você sabe que tem um SD quando a falha de um computador do qual você nunca ouviu falar o impede que você faça qualquer trabalho.”**

Transparência – Exceções

- Se um processo executado em São Francisco (USA) se conecta a outro executado em Amsterdã (Holanda), nada fará com que um processo envie ao outro algo e essa “conversa” demore menos que 35 milissegundos.
- Na prática isso nunca ocorrerá, pois vários comutadores intermediários irão impedir que a demora fique limitada apenas a restrição imposta pela velocidade da luz.
- Se várias réplicas, localizadas em distintos continentes, tiverem que ser atualizadas simultaneamente, uma simples atualização pode levar alguns segundos, algo que não pode ser furtado dos usuários.

Transparência – Exceções

- Se requisitarmos um jornal eletrônico sempre às 7 horas da manhã, mas naquele momento estivermos do outro lado do mundo, seu jornal matutino certamente não será o esperado.

- Se um servidor estiver inoperante, muitas vezes é melhor permitir que o usuário cancele múltiplas tentativas de acesso ou se tentar outro servidor, a se tentar “esconder” a falha em nome da transparência num processo que poderá ficar absurdamente mais lento.

Abertura

- Um sistema distribuído é aberto porque oferece serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços.
- Numa rede de computadores há regras padronizadas que governam o formato, o conteúdo e o significado das mensagens enviadas e recebidas.
- Essas regras são formalizadas em protocolos,sendo especificadas por meio de uma interface **IDL** (***Interface Definition Language*** – Linguagem de Definição de Interface).

Abertura

- Quando adequadamente especificada, uma definição de interface permite que um processo arbitrário que necessite de certa interface se comunique com um outro processo que fornece aquela interface.
- Permite que duas partes independentes, por exemplo, construam implementações completamente distintas dessas interfaces, que resulta em dois sistemas distribuídos separados que funcionam exatamente da mesma maneira.

Abertura

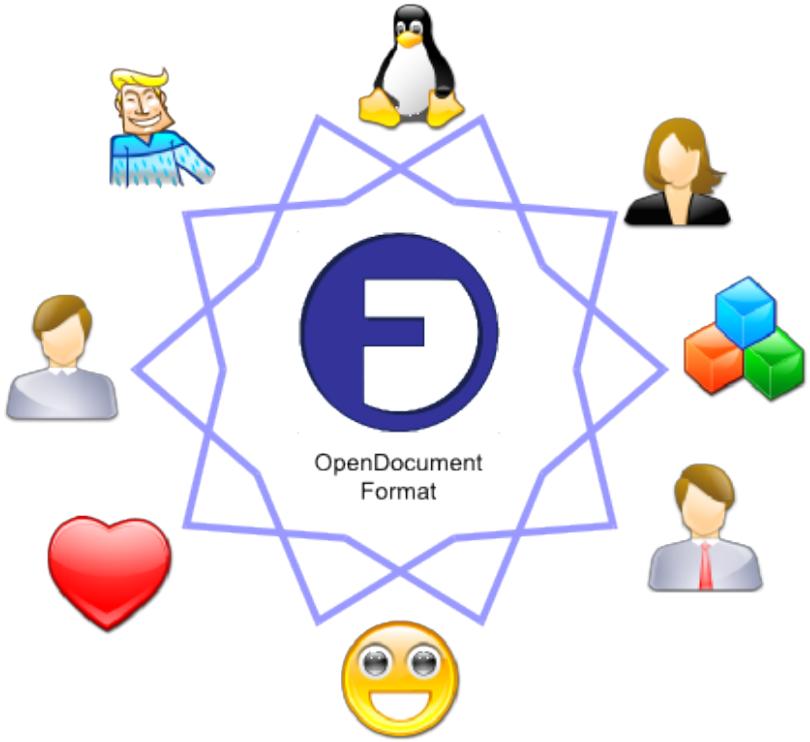
- Diz-se que uma especificação adequada deve ser **completa** e **neutra.**

- **Completude** significa que tudo o que é necessário para uma implementação foi de fato especificado.

- **Neutralidade** significa que as especificações nunca prescrevem como deve ser a aparência/estrutura de uma implementação.

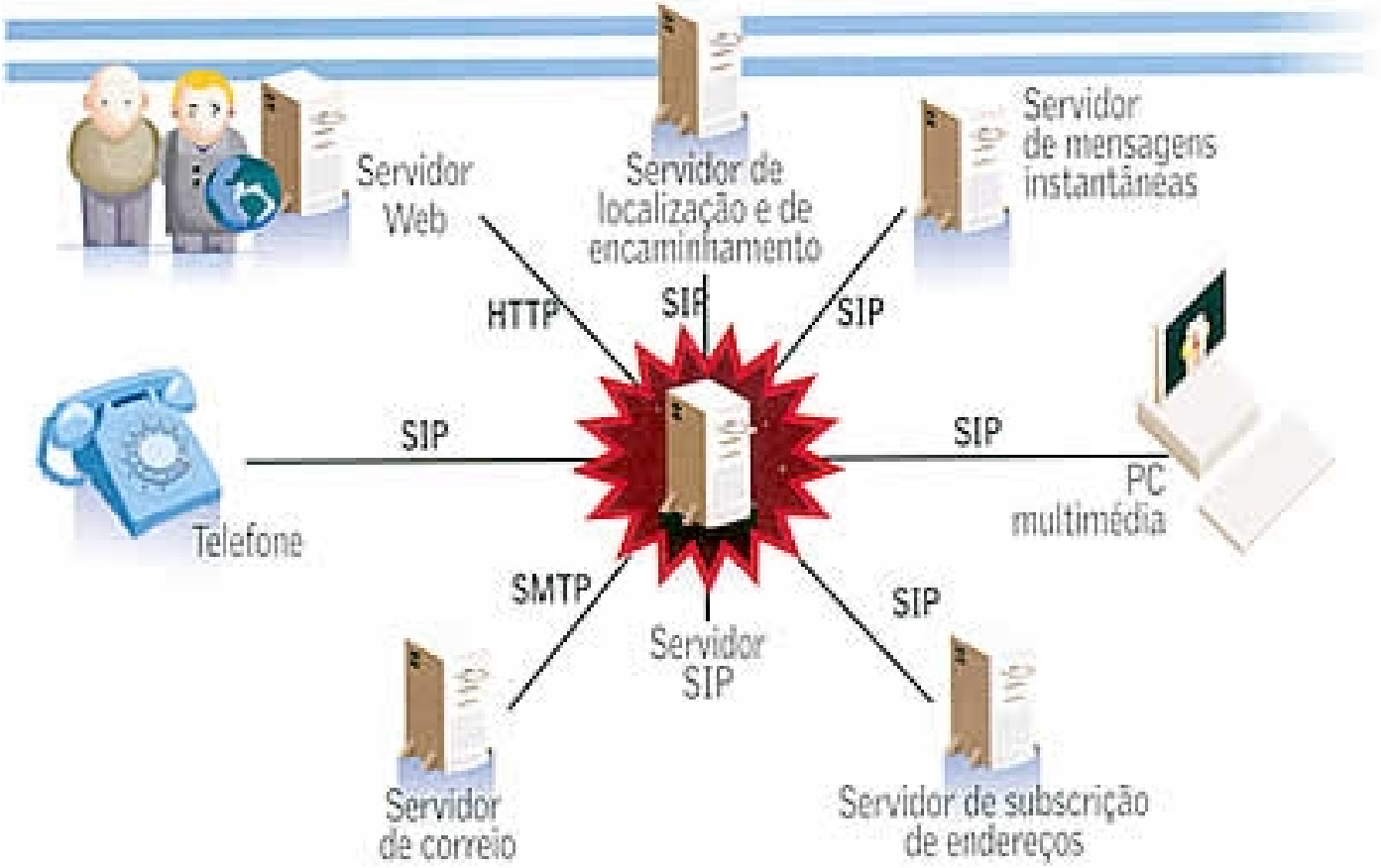
Interoperabilidade

- A **interoperabilidade** se caracteriza por determinar ***até que ponto*** duas implementações de sistemas ou componentes de fornecedores distintos devem coexistir e trabalhar em conjunto, com base na mera confiança mútua nos serviços de cada um, especificados por um padrão comum.



Portabilidade

- A **portabilidade**, por sua vez, se caracteriza por permitir ***<u>até que ponto</u>*** uma aplicação criada para um sistema distribuído ***A*** pode ser executada sem modificações num sistema distribuído distinto ***B***, que implementa as mesmas interfaces de ***A***.



Políticas e Mecanismos

- Todo sistema distribuído deve estar organizado com um conjunto de componentes relativamente pequenos e de fácil substituição ou adaptação.
- Isso implica que as especificações não fiquem restritas somente as interfaces de nível mais alto, mas também aquelas das partes mais internas dos sistemas e de sua interação.

Políticas e Mecanismos

- Nas especificações monolíticas a substituição ou adaptação de um único componente pode afetar o sistema todo.
- Separação dos componentes nas especificações monolíticas é apenas lógica, isto é, estes continuam a ser implementados num único e imenso programa,
- Esse tipo de pensamento não é adequado a sistemas abertos, mas a fechados.

Políticas e Mecanismos

- Necessita-se de uma clara separação entre o que é política e o que é mecanismo.
- Por exemplo
  - Um sistema contém um mecanismo que permite armazenar documentos e um usuário tem que seguir uma política de armazenamento de documentos
  - Um boa opção é que o usuário possa implementar sua própria política sob a forma de um componente que realize a política por ele desejada.

Estudo de Caso - browser

- Cache na *World Wide Web*.
  - Em geral, os browsers permitem aos usuários adaptarem sua política de ***cache*** especificando o tamanho da ***cache*** e se a consistência de um documento em ***cache*** deve ser verificada sempre ou apenas uma vez por sessão.
  - Contudo, esse mesmo usuário não pode determinar o tempo que um documento deve ficar no ***cache***.

Estudo de Caso - browser

- Cache da *World Wide Web*
  - Além disso, esse usuário não pode criar políticas que visem determinar que documentos devem ficar nas ***caches*** em função de seus *conteúdos*.
  - Usuário pode até decidir manter na ***cache*** os horários de partidas de ônibus intermunicipais, até porque esses horários raramente se modificam.
  - Contudo o mesmo não se aplica para as condições das rodovias, que potencialmente mudam de uma hora para outra, sem aviso prévio.

Escalabilidade

São três os critérios para se estabelecer a escalabilidade:

1. **Quanto ao seu tamanho**: Um sistema é **escalável** em relação ao seu **tamanho**, quando é possível se inserir mais usuários e recursos ao sistema em si.
2. **Quanto ao local:** Um sistema é **escalável** do ponto de vista **geográfico**, quando os usuários e recursos podem estar um longe dos outros.
3. **Quanto a sua administração:** Um sistema é **escalável** do ponto de vista de sua **administração** quando ele pode ser facilmente gerenciado, mesmo que abranja muitas organizações administrativas distintas.

Escalabilidade

Problemas e limitações

- Quando um sistema escalável contém essas várias dimensões, muitas vezes apresenta queda de desempenho na medida em que é ampliado

Característica	Exemplo
Serviços Centralizados	Único servidor para todos os dados
Dados Centralizados	Lista telefônica on-line
Algoritmos Centralizados	Fazer roteamento com base em informações completas

Escalabilidade - Problemas

- **Tamanho:** Ao inserirmos continuamente mais usuários tendemos a chegar a um ponto em que o servidor centralizado atinja seu limite operacional, transformando-se no gargalo do sistema.
- Mesmo que admitamos que esse servidor tenha capacidade de processamento e de armazenagem ilimitadas, ainda assim sofrerá restrições devido a sua capacidade de comunicação ser limitada.

Escalabilidade - Problemas

- **Segurança**: Por exemplo, um servidor destinado apenas ao armazenamento de informações sigilosas sobre dados médicos, financeiros ou confidenciais de pessoas.
- Nesse caso, a implementação obrigatoriamente terá que ser feita num único servidor que deve estar numa sala separada, de alta segurança, separado de outras partes do sistema.
- Copiar o servidor para melhorar o desempenho para outras localizações, definitivamente, não seria uma boa ideia, nessa situação.

Escalabilidade - Problemas

- Impensável também seria uma solução com dados centralizados que contivesse os dados de 50 milhões de pessoas.
- Não haveria limitações quanto ao espaço em disco armazenado, mas sem dúvida haveria saturação de todas as linhas de comunicação que o acessam.
- É o que ocorreria, por exemplo, se todo Sistema de Nomes de Domínio (***Domain Name System* – DNS**) estivesse implementado numa única tabela.
- Se cada requisição para resolver um URL fosse passada para esse hipotético servidor, ninguém estaria usando a Web.

Escalabilidade e o tipo de comunicação

- A localização de um serviço numa rede local é baseada em ***broadcast***.
- Um processo envia a todas máquinas, perguntando a cada uma se está executando o serviço que ele necessita.
- Apenas as máquinas que estão executando tal serviço respondem informando seu endereço.
  - O que ocorreria se tentássemos localizar dessa maneira um serviço na Internet?
- Em vez disso, é preciso projetar serviços especiais de localização, que talvez tenham alcance mundial e capazes de atender a bilhões de usuários.

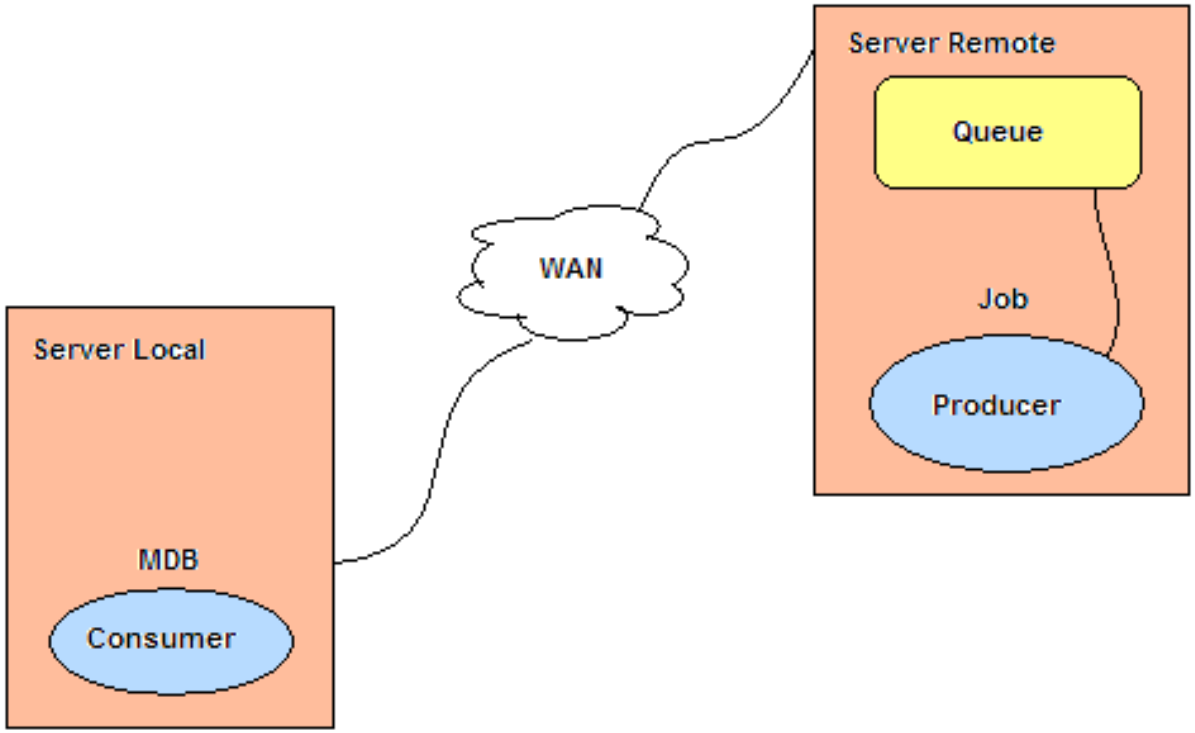
Escalabilidade e o tipo de comunicação

- Ampliar sistemas distribuídos a partir de sistemas projetados para redes locais é algo difícil, pois estes foram criados a partir da filosofia **síncrona**.
- Assim quando uma parte requisita um serviço, fica bloqueada até que uma mensagem seja enviada de volta.
- Numa rede local (LANs), na pior das hipóteses o retorno dessa mensagem demora algumas centenas de microssegundos.
- Todavia, quando pensamos numa rede de longa distância, essa comunicação pode demorar centenas de milissegundos (três ordens de grandeza mais lenta).
- Ainda existe a questão de que em redes de longa distância a comunicação é não confiável.

Técnicas de Escalabilidade

1. Ocultar latências de comunicação

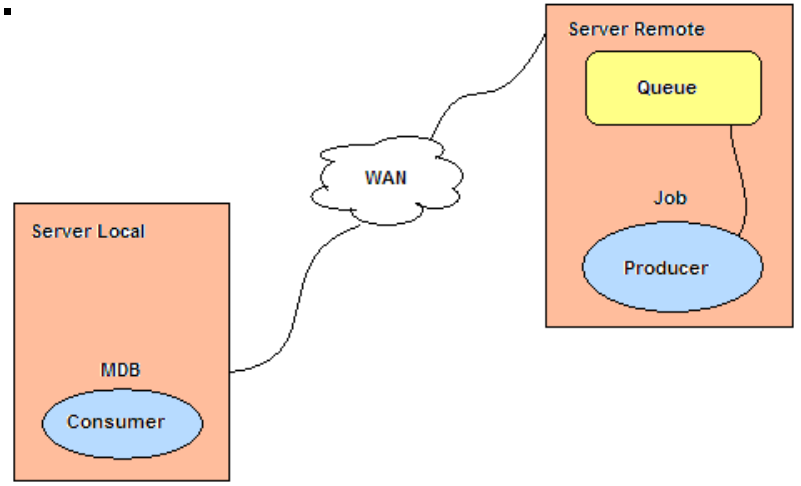
- A ideia central é: evitar espera por respostas a requisições remotas e lentas.
- Uma requisição a uma máquina remota fica pendente, enquanto esta é liberada para continuar executando trabalhos úteis ao lado requisitante.



Técnicas de Escalabilidade

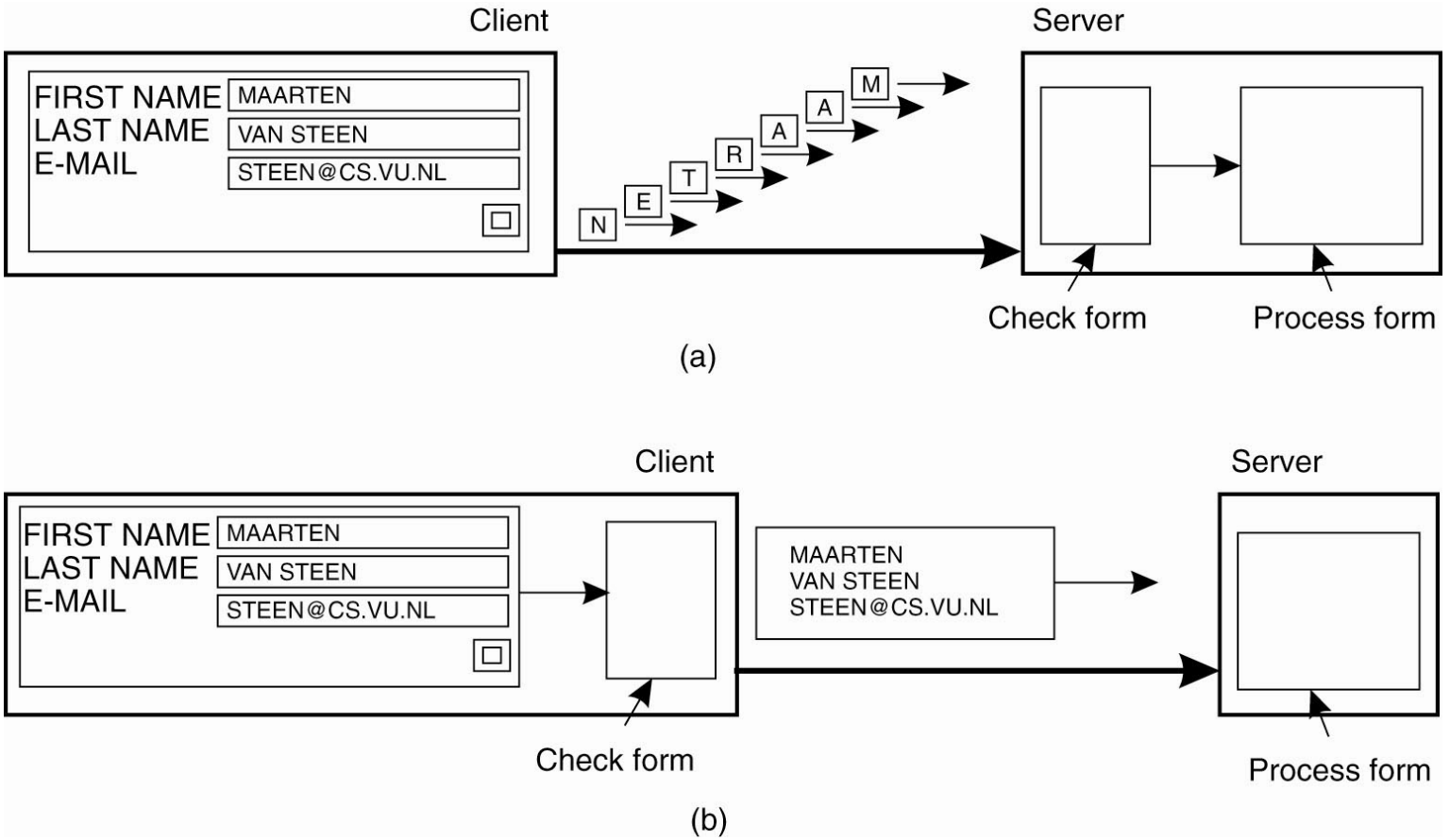
1. **Ocultar latências de comunicação**
   - Essencialmente a aplicação requisitante deve ser escrita usando comunicação **assíncrona** para evitar bloquear processos à espera de receber respostas.
   - Quando a resposta retorna, a aplicação é interrompida e um manipulador especial é chamado para concluir a requisição emitida anteriormente.

- Isso implica no tratamento de respostas como eventos e o uso do conceito de várias tarefas em paralelo para continuar o processamento.



Técnicas de Escalabilidade

1. Ocultar latências de comunicação



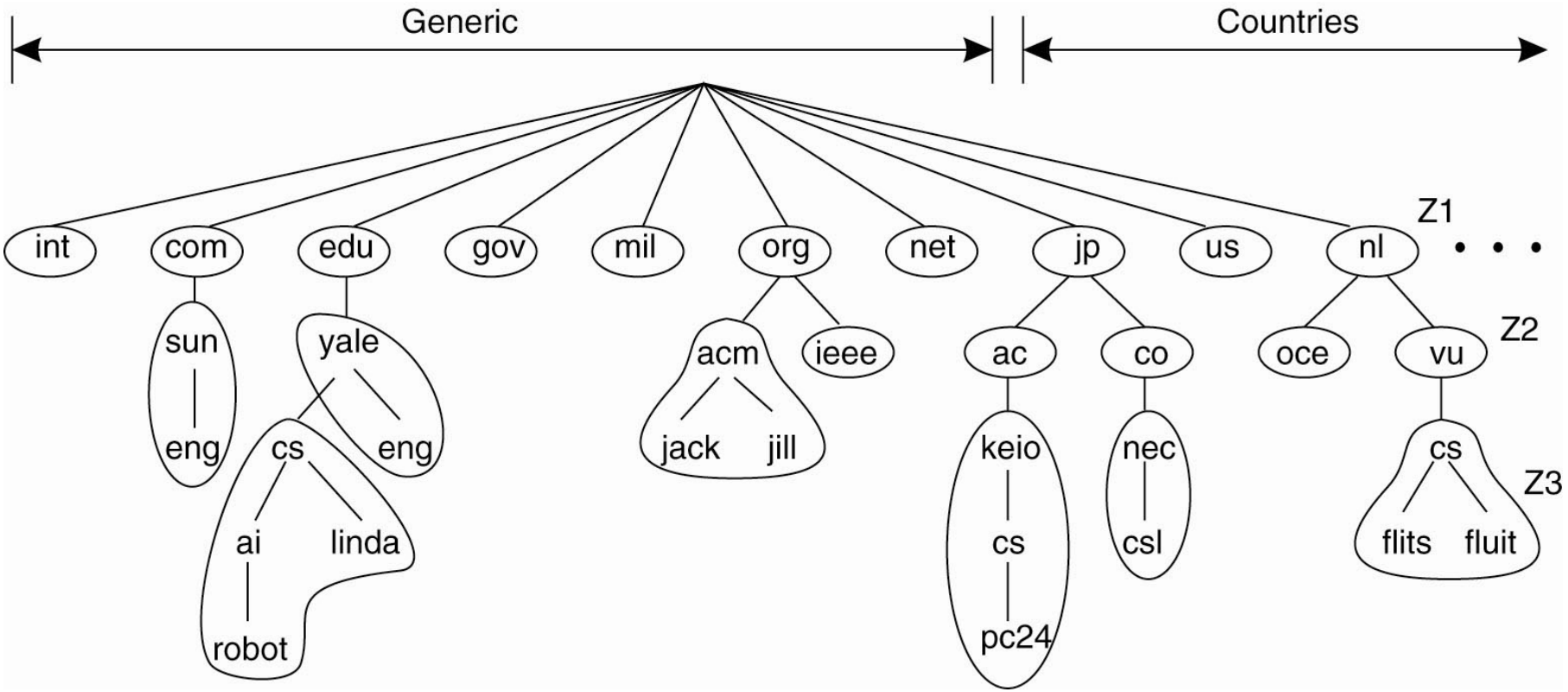
Técnicas de Escalabilidade

1. Ocultar latências de comunicação

- Exemplificando, se um acesso for feito a um banco de dados por meio de formulários (a), o preenchimento de formulários pode ser feito com o envio de uma mensagem separada para cada campo e a espera por um reconhecimento do servidor, que verifica se há algum erro sintático antes de aceitar a entrada.
- Uma alternativa bem melhor é apresentada no outro esquema (b), em que o cliente é quem preenche o formulário e devolve completamente preenchido.
- Essa é a forma usada atualmente suportada pela Web, sob a forma de linguagens como o **Javascript**.

Técnicas de Escalabilidade

2. Distribuição



Técnicas de Escalabilidade

2. Distribuição

- A ideia é distribuir entre vários computadores os serviços e dados. Decompondo os componentes em partes menores, que por sua vez são distribuídas por todo o sistema distribuído.
- Exemplificando, o DNS (Sistema de Nomes de Domínios da Internet), o espaço de nomes do DNS é organizado em hierarquia em uma árvore de domínios, dividida em zonas de superposição em acordo com a figura.
- Assim, nenhum servidor prestará todos os serviços nem possuirá todos os dados, portanto se um servidor falhar, nem tudo está perdido...

Técnicas de Escalabilidade

3. Replicação

- Aumenta a disponibilidade dos serviços e melhora o balanceamento de carga no sistema, resultando na melhoria do desempenho.
- Aumenta a disponibilidade do serviço nas zonas da rede onde há réplicas.
- ***Caching*** é uma forma de replicação controlada pelo cliente. A existência de várias cópias pode levar a problemas de **consistência**.
- A cache ocorre sob demanda, a replicação costuma ser planejada antecipadamente
- Se for necessário ter garantias fortes de consistência tem de se atualizar as cópias imediatamente.
- Além disso, se duas atualizações ocorrerem simultaneamente, também se exige a atualização de cada cópia na mesma ordem.

As ciladas da escalabilidade

- Premissas **falsas** adotadas ao se desenvolver pela primeira vez uma aplicação distribuída:

1. Rede é confiável
2. Rede é segura
3. Rede é homogênea
4. Topologia constante
5. Latência é zero
6. Largura de banda é infinita
7. Custo de Transporte é zero
8. Existe somente um administrador

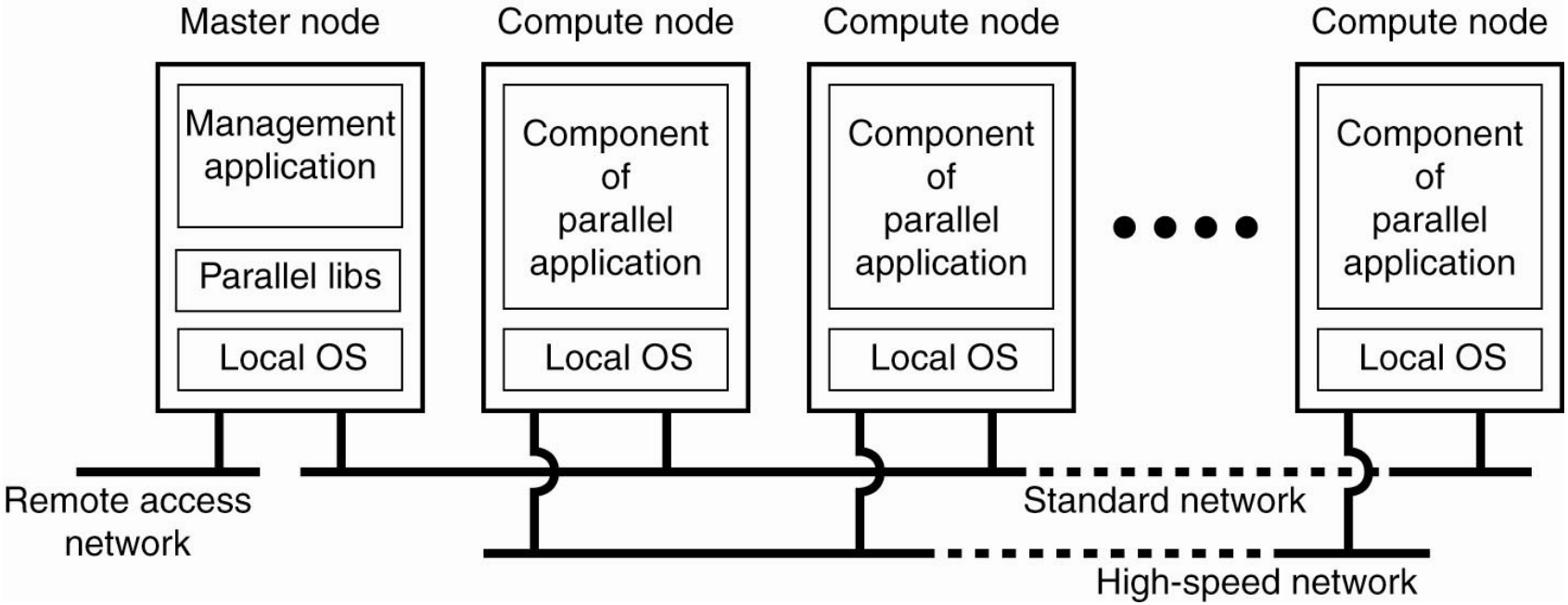
Sistemas de Computação Distribuídos

- Os sistemas de computação distribuídos são utilizados para tarefas de alto desempenho e podem ser subdivididos em duas classes distintas:
  - **Sistemas de computação de cluster**
  - **Sistemas de computação em grade**

- Homogêneo x Heterogêneo

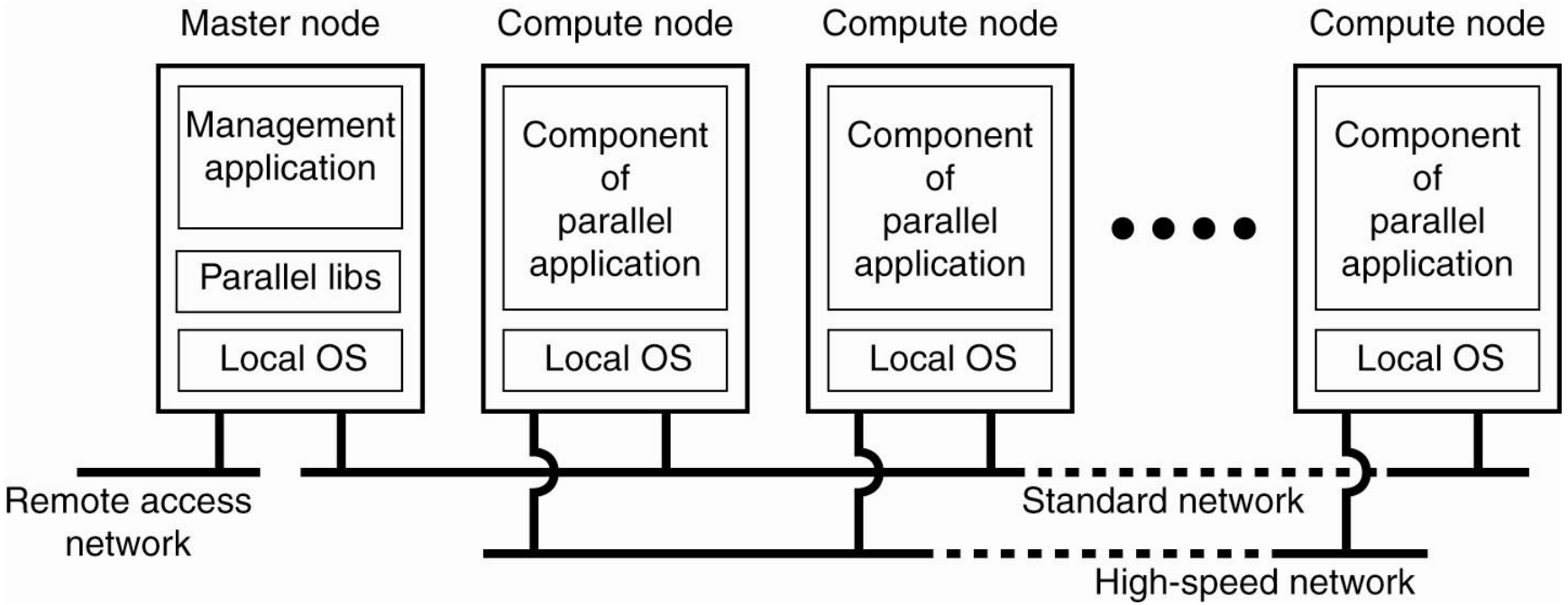
Sistemas de Computação Distribuídos

- **Sistemas de Computação – *Cluster***
- Hardware consiste em um conjunto de estações de trabalho conectadas e se desenvolveu a partir do barateamento dos computadores pessoais.
- Conexão é feita por meio de uma rede local, e em quase todos os casos, a computação de ***cluster*** é usada para programação paralela na qual um único programa é executado em paralelo.



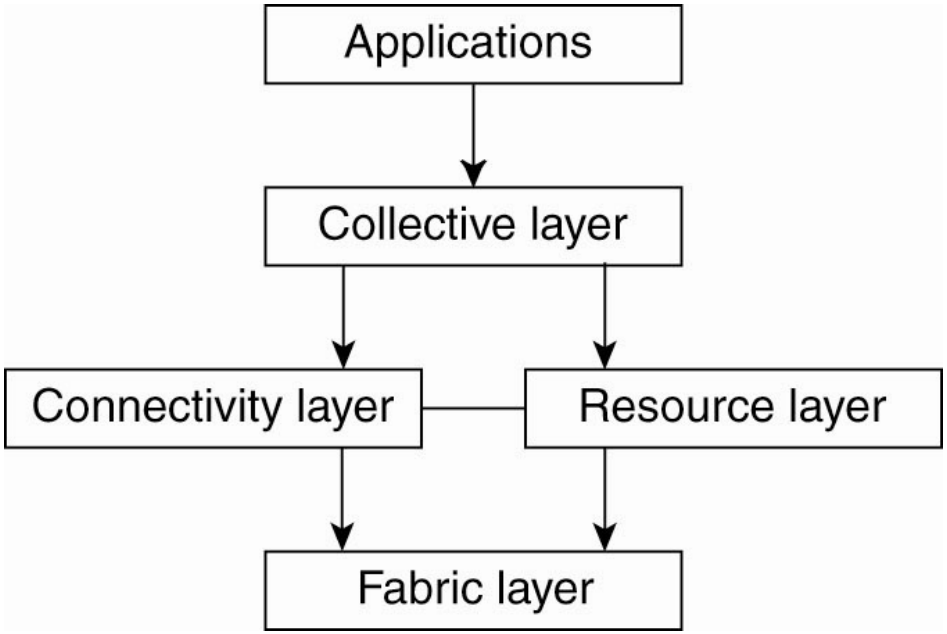
Sistemas de Computação Distribuídos

- **Sistemas de Computação – *Cluster***
- Exemplo: Sistemas Beowulf baseado em Linux, em que cada *cluster* consiste em um conjunto de nós de computação controlados e acessados por meio de um único só mestre, cujas tarefas são manipular e alocar determinados programas paralelos.

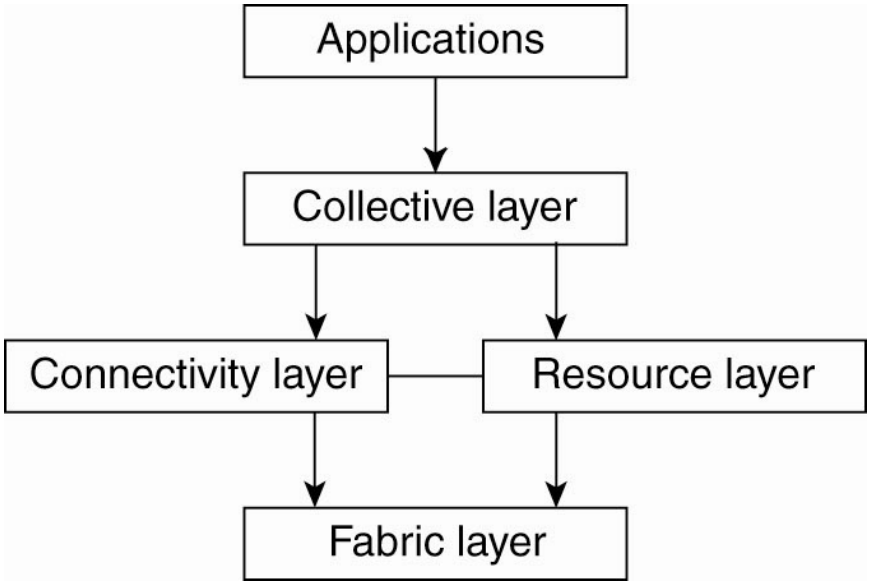


Sistemas de Computação Distribuídos

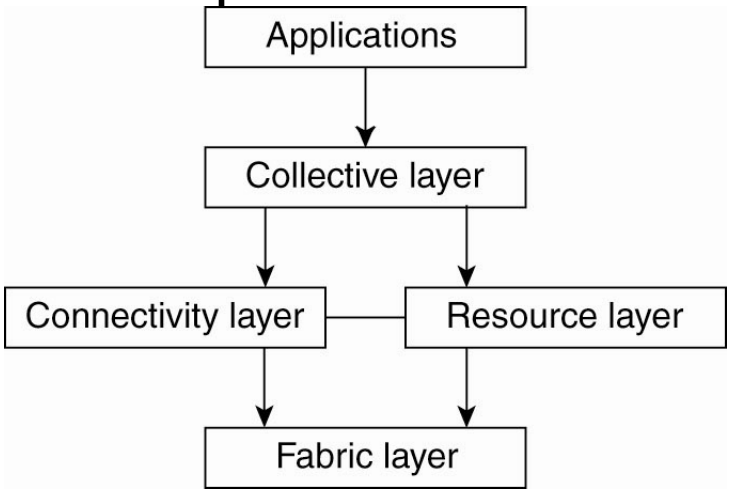
- **Sistema de Computação em Grade**
- Sistemas Distribuídos montados como federação de computadores, na qual cada sistema pode cair sob um sistema administrativo diferente, e pode ser muito diferente no que tange a hardware, software e tecnologia empregada.
- Apresentam alto grau de heterogeneidade.



- **Sistema de Computação em Grade**
- Software usado para permitir acesso a recursos (muitas vezes *clusters*) de diferentes organizações reunidos para permitir a colaboração de um grupo (conceito de organização virtual).
- Foco dirigido para a arquitetura, está dividida em quatro etapas, em acordo ao esquema apresentado na figura

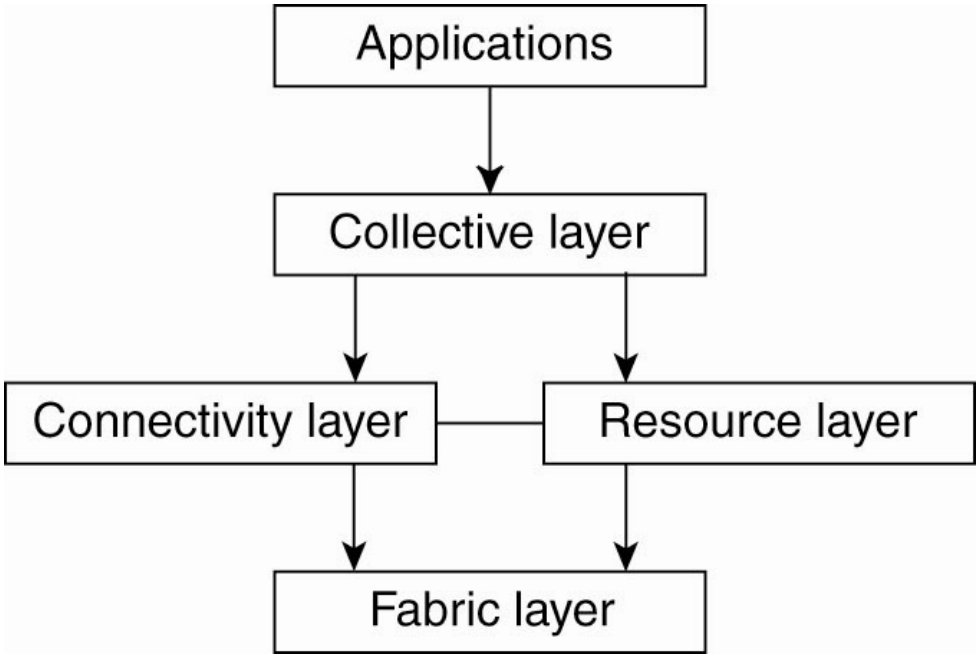


- **Sistema de Computação em Grade**
- A camada mais baixa, chamada **Fabric Layer** (camada base), provê interfaces para recursos locais em um site específico.
- Essas interfaces são projetadas para permitir compartilhamento de recursos dentro de uma organização virtual.
- Provê funções para consultar o estado e as capacidades de um recurso, em conjunto com funções para o gerenciamento propriamente dito. Exemplo: Travar um recurso.

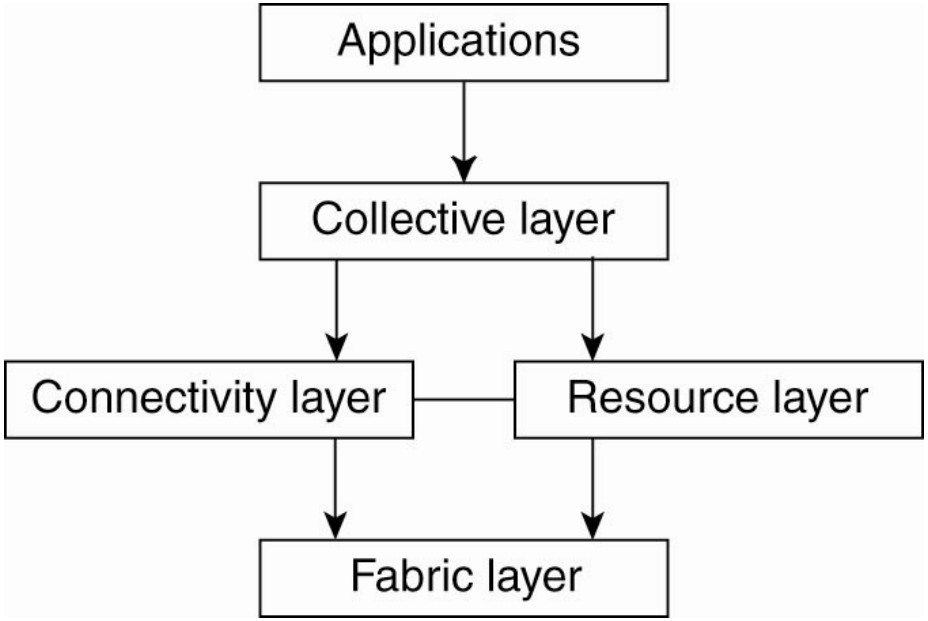


Sistemas de Computação Distribuídos

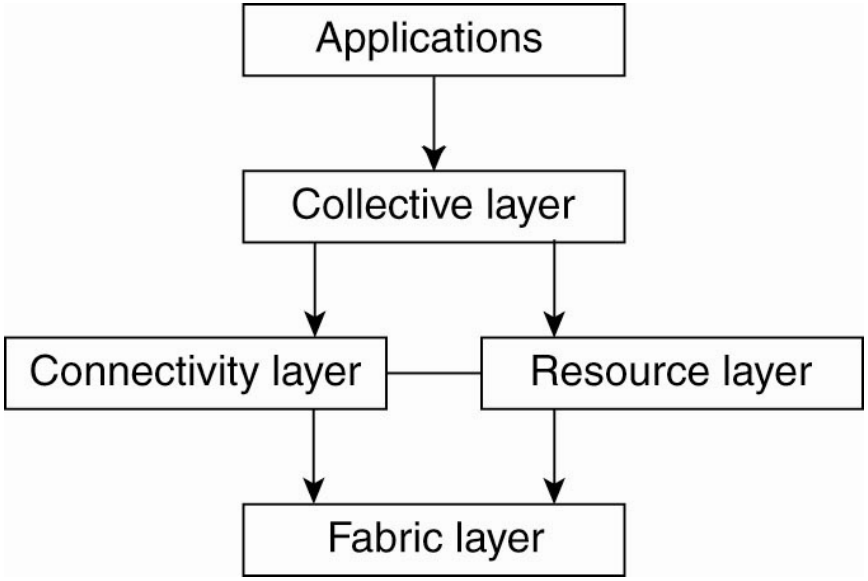
- **Sistema de Computação em Grade**
- A camada de **conectividade** consiste em protocolos de comunicação para suportar transações da grade que abranjam a utilização de múltiplos recursos.
- Exemplo: Protocolos de transferência de dados entre recursos ou para localizar um recurso desde uma localização remota.



- **Sistema de Computação em Grade**
- A camada de **recursos** é responsável pela gestão de um único recurso, utilizando funções fornecidas pela camada de **conectividade** e chama diretamente as interfaces disponibilizadas pela camada **base** (*Fabric Layer*).
- Exemplo: Recursos é responsável pelo acesso, dependendo da autenticação realizada pela camada de conectividade.



- **Sistema de Computação em Grade**
- A camada **coletiva** manipula o acesso a múltiplos recursos e normalmente consiste em serviços para descoberta de recursos, alocação e escalonamento de tarefas para múltiplos recursos, como replicação de dados.
- A camada de **aplicação** consiste em aplicações que funcionam dentro de uma organização.

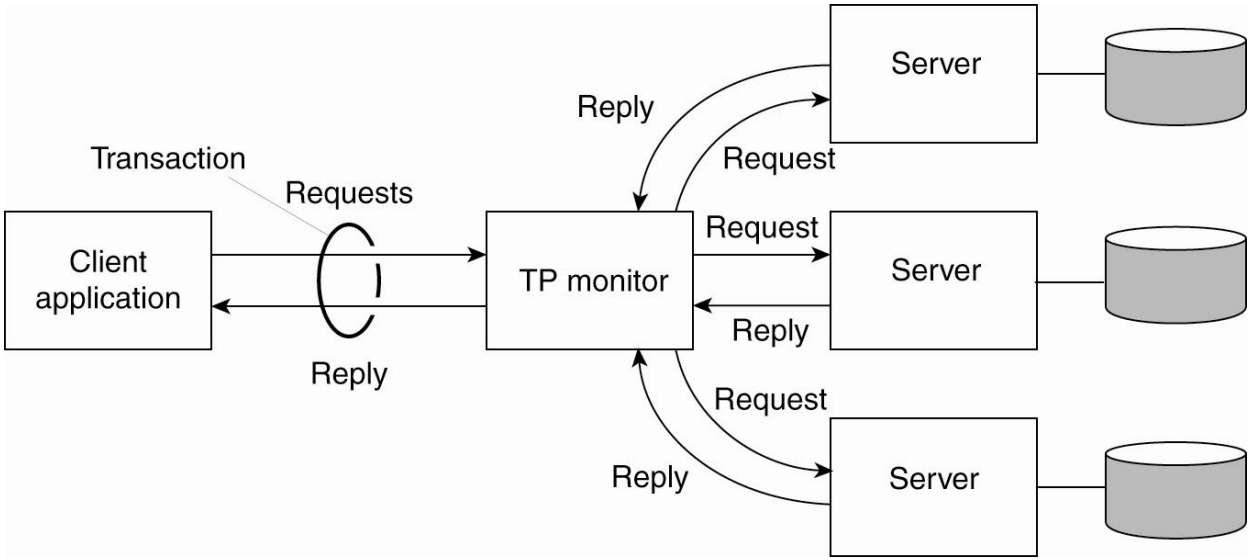


- **Sistemas de Processamento de Transações**
- A partir do conhecimento de que as operações num banco de dados são feitas sob a forma de **transações**, fica natural pensarmos num processo que tem em algum ponto um início (BEGIN_TRANSACTION) e num outro local um término (END_TRANSACTION).
- Supõe-se que se todo processo ocorrer como se espera inicialmente a transação será realizada, de forma integral, no banco de dados (COMMIT).
- Todavia, se por qualquer que seja o motivo alguma parte dessa transação não puder ser executada, toda a transação é desfeita (ROLLBACK). Em geral, chama-se a esse processo de **ACID**.

- **Sistemas de Processamento de Transações**
  - **Atômicas:** Para todo restante do sistema, a transação acontece como se fosse indivisível.
  - **Consistente:** A transação não viola invariantes de sistema. [Notar que invariantes podem ser violados por breves instantes]
  - **Isoladas:** Transações concorrentes não interferem entre si. [Serializáveis]
  - **Duráveis:** Uma vez comprometida uma transação, as alterações serão permanentes.

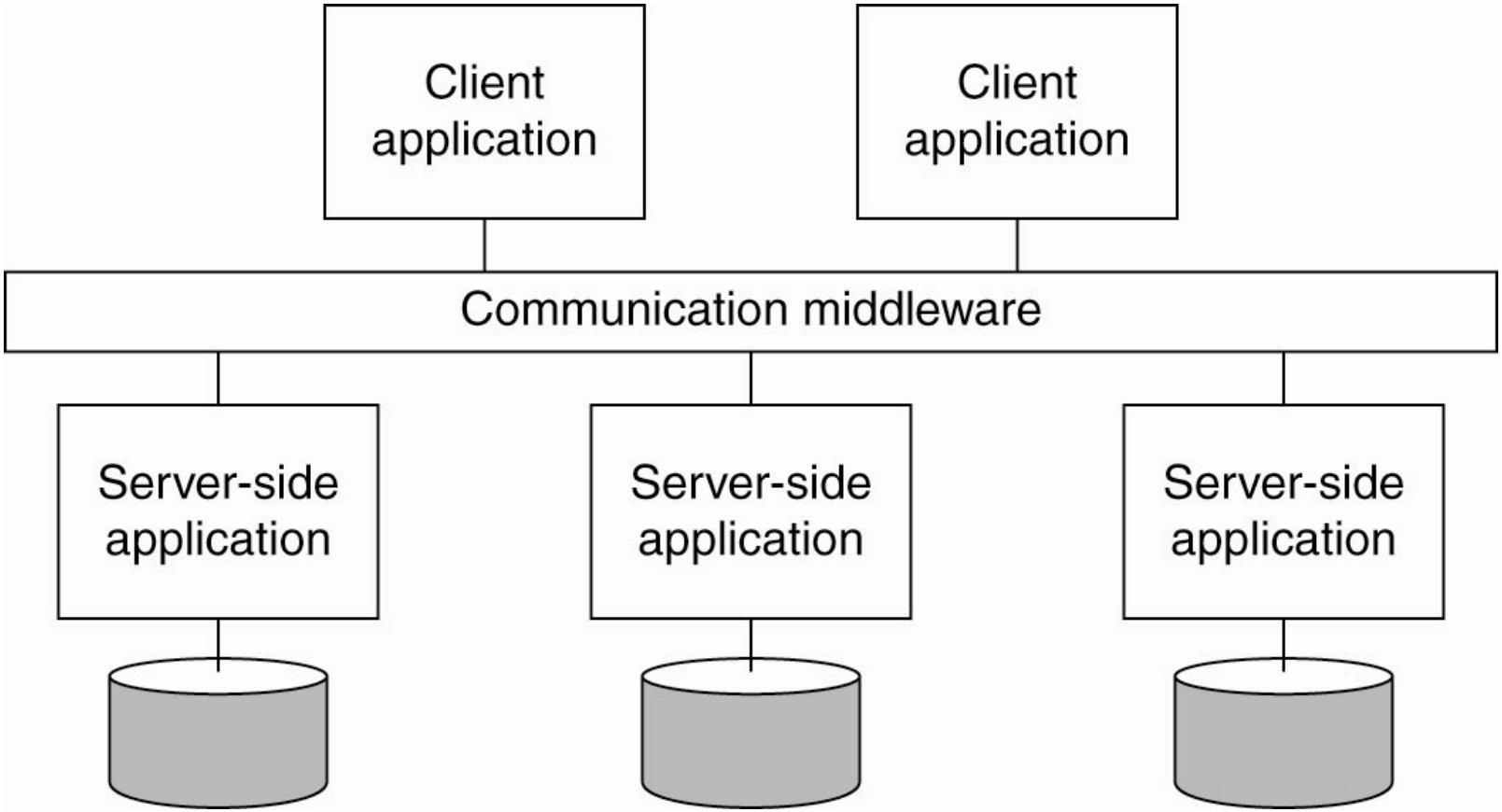
Sistemas de Informação Distribuídos

- **Sistemas de Processamento de Transações**
- O monitor de TP (Monitor de Processamento de Transação) permite que uma aplicação acesse vários servidores de banco de dados simultaneamente, dando a impressão de um processo único.
- Figura abaixo representa algo distinto, como o banco da cia. Aérea, o banco do hotel e o banco do receptivo, para somente em caso de possibilidade simultânea nos três bancos, uma transação de turismo ser realizada.



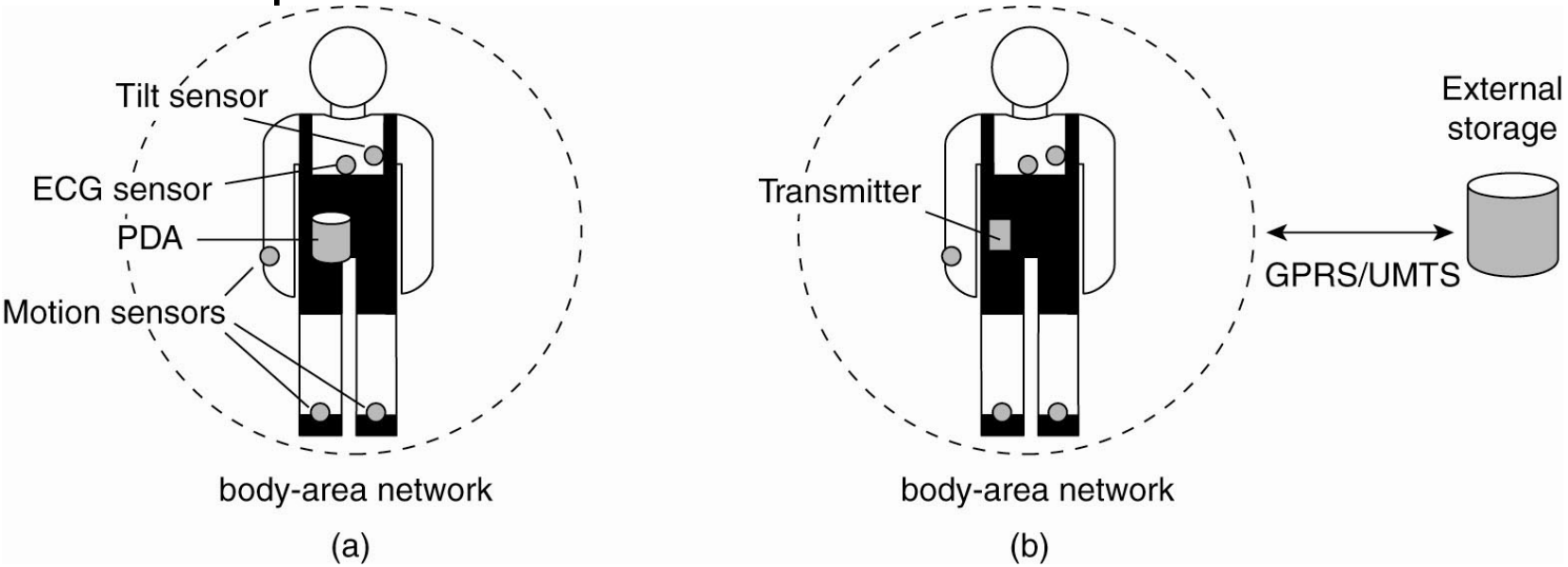
Sistemas de Informação Distribuídos

- **Sistemas de Processamento de Transações**
- *Middleware* como um facilitador de comunicação na integração entre aplicações distintas.



Sistemas Pervasivos

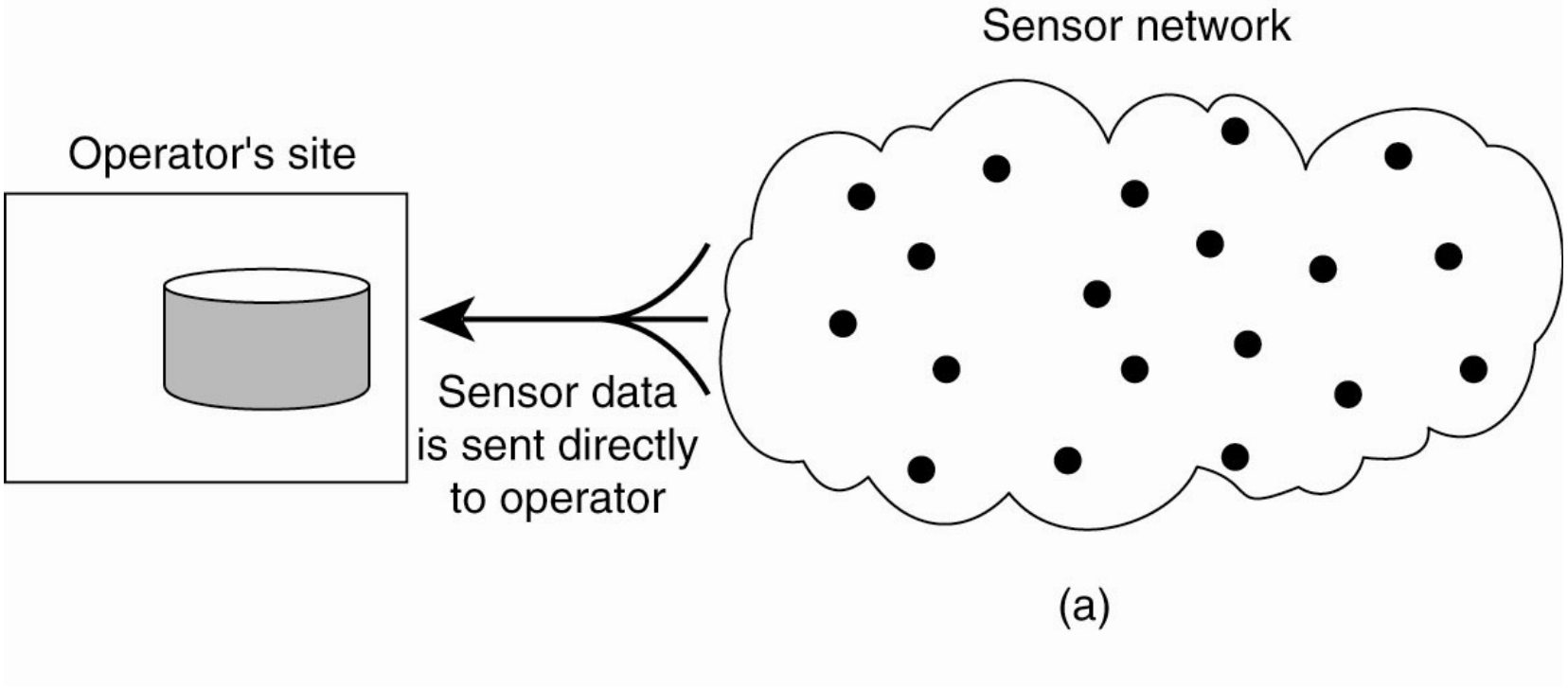
- Abranger mudanças de contexto
- Tendência a estrutura *ad hoc*
- Reconhecer compartilhamento (dados, recursos) como padrão
- Por exemplo: sistemas eletrônicos de *health care*



Obs: GPRS (General Packet Radio Service)
UMTS (Universal Mobile Telecommunications System)

Sistemas Pervasivos

- Redes de sensores sem fio
- Configuração automática com mudança de topologia
- Como controlar e agregar dados em um lugar monitorado
- Rede deve ser autonômica



Sistemas Pervasivos

- Redes de sensores sem fio