GBC074 – Sistemas Distribuídos

Introdução

Baseado no material disponível pelo author em
https://www.distributed-systems.net

Sistemas Distribuídos

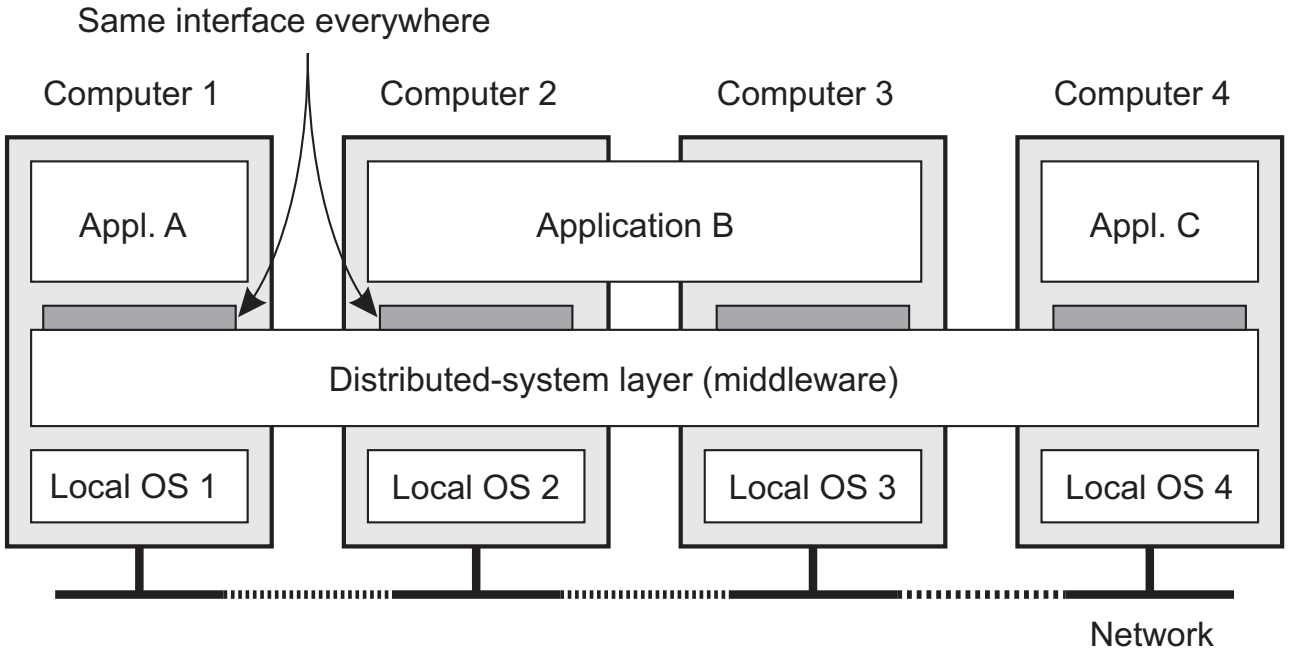
- Definição
  - Um Sistema distribuído é uma coleção de **sistemas computacionais autônomos** (independentes) que se apresentam ao usuário com um **sistema único coerente**
- Características
  - Sistemas computacionais autônomos ou **nós**:
    - dispositivos de hardware ou processos de software
  - Sistema único coerente:
    - percebido como um único sistema ⇒ nós colaboram na execução da tarefa.

Nós autônomos

- Comportamento independente
- Cada nó tem sua própria noção de tempo:
  - **Sem relógio global**
  - Problemas fundamentais de sincronização e coordenação
- Comunicação:
  - memória compartilhada
  - mensagens
- Em uma coleção de nós:
  - Como se faz o gerenciamento de membros do grupo (*group membership*)?
  - Como se garante comunicação com (não) membro autorizado?

Sistemas Distribuídos e o Middleware

- Equivalente ao “sistema operacional” de um sistema distribuído
- Oferece diversos serviços:
  - Comunicação
  - Segurança
  - Métricas
  - Recuperação de falhas
  - Transações
  - Composição de serviços

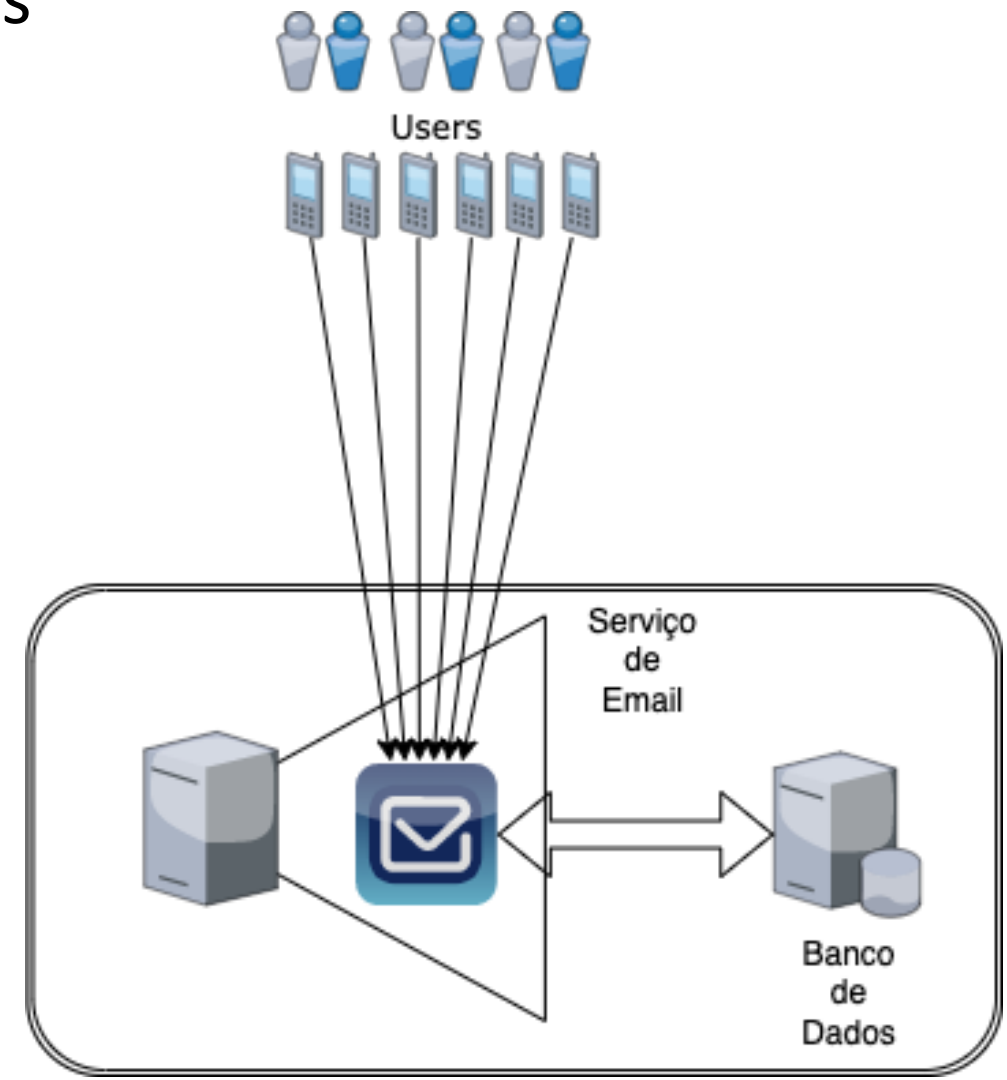


Sistemas Distribuídos

- Exemplos
  - Entregue este e-mail para fulano@knowhere.uni
  - Autorize a transferência de D dinheiros da conta C para a conta C'.
  - Movimente o braço mecânico que está segurando um bisturi, 3cm à direita, então abaixe-o 3mm, e movimente-o 4cm para a esquerda
  - Leia o valor do sensor de temperatura T e, caso seu valor supere V, emita alarme luminoso vermelho intermitente e alarme sonoro

Sistemas Distribuídos

- Exemplos



Sistemas Distribuídos

Um outro ponto de vista...

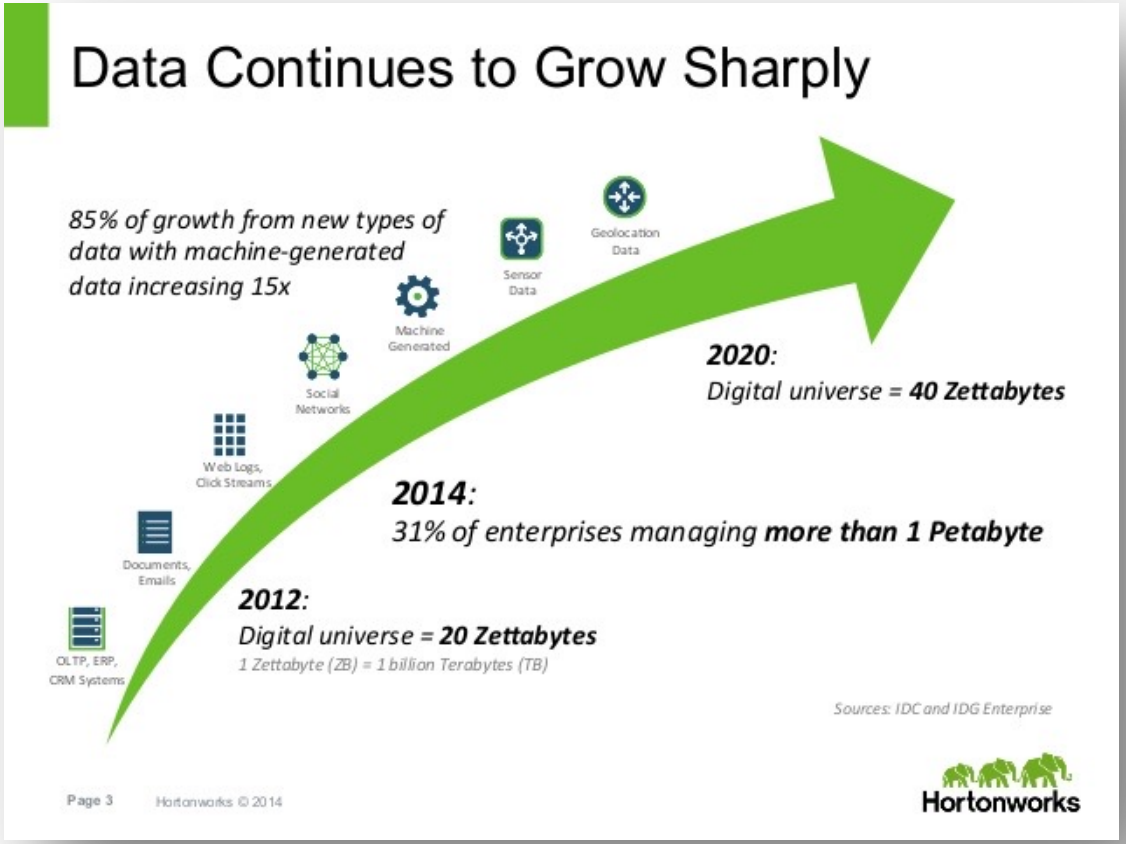
- Segundo Leslie Lamport:

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

- Um "bom" sistema distribuído:
  - Disponível:
    - Sempre funcional
  - Poder computacional:
    - Capacidade de processamento
    - Capacidade de armazenamento
    - Baixa latência
  - Baixo custo
    - Tamanho apropriado à tarefa

Por que utilizar sistemas distribuídos?

Computadores individuais tem capacidade reduzida de processamento e armazenamento, mas …



… necessidade de poder computacional cresce exponencialmente

Transparência de um sistema distribuído

Transparência	Esconde...
Acesso	...diferenças na representação dos dados e o modo como eles são acessados
Localização	...onde está o objeto
Realocação	...uma possível movimentação do objeto enquanto sendo usado
Migração	...que o objeto pode ser movido
Replicação	...que o objeto é replicado
Concorrência	...que o objeto pode ser compartilhado por várias usuários
Falha	...falhas e recuperação

Abertura de um sistema distribuído

- Capacidade de interagir com serviços de outros sistemas abertos, independente do ambiente sobre o qual foram construídos:
- Requisitos:
  - Interfaces bem definidas
  - Interoperabilidade
  - Portabilidade de aplicações
  - Extensibilidade

- Implementado por meio de
  - Políticas
    - Exemplos: nível de consistência para cache, mecanismo de QoS, nível mínimo de segurança
  - Mecanismos
    - Exemplos: lista de mecanismos de cache para seleção, listas de parâmetros de QoS e de algoritmos de criptografia para seleção

Escalabilidade em Sistemas Distribuídos

- Pelo menos 3 componentes a considerar:

1. Número de usuários ou processos (escalabilidade de tamanho)
2. Distância máxima entre nós (escalabilidade geográfica)
3. Número de domínios administrativos (escalabilidade administrativa)

- Maioria dos sistemas se preocupam apenas com o primeiro caso

Escalabilidade em Sistemas Distribuídos

- Soluções:

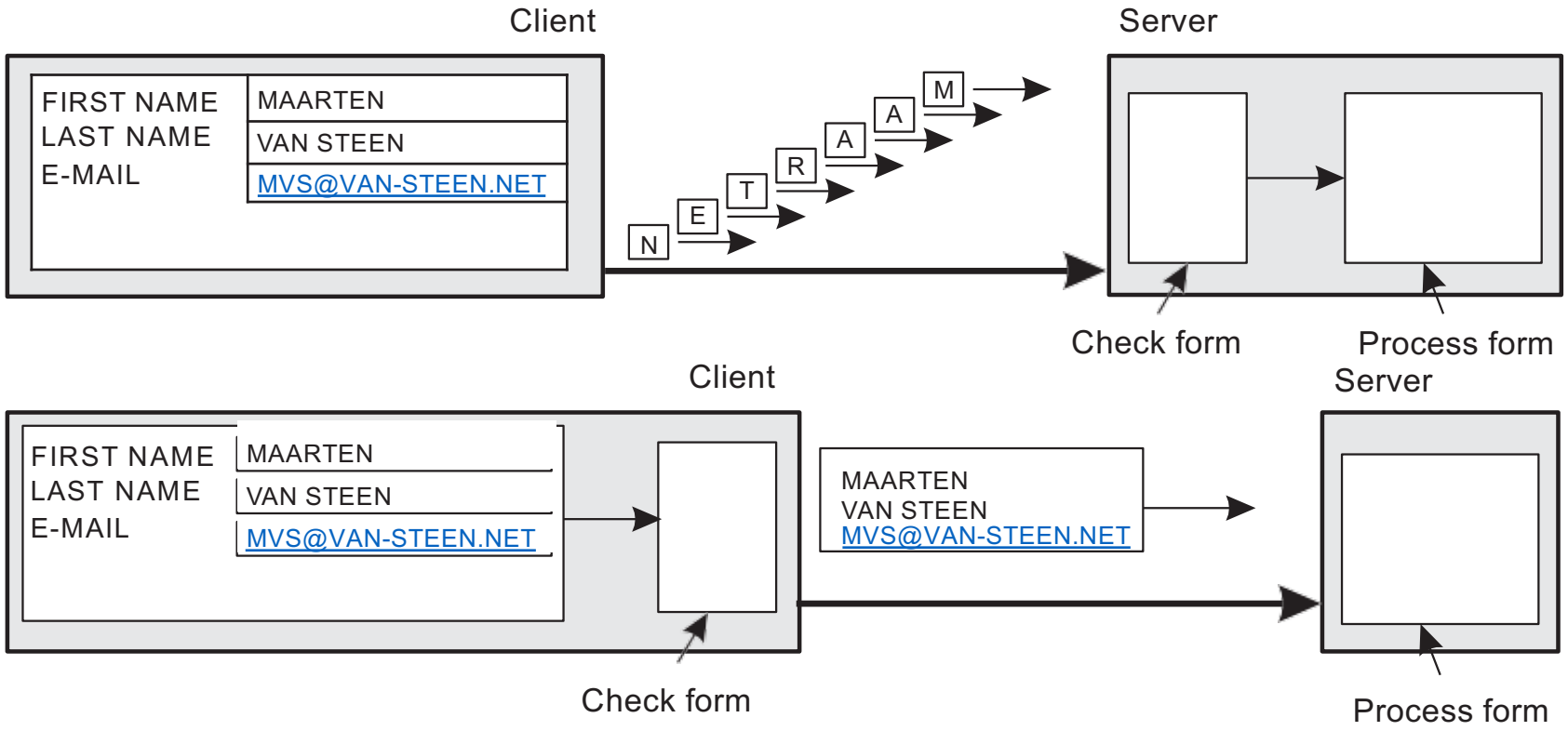
1. Scale up (vertical scaling): aumentar a capacidade de um dispositivo
   - Custo exponencial
   - Limitado
2. Scale Out: agregar o poder computacional de diversos computadores "baratos”
   - Capacidade ilimitada de crescimento (em teoria)
   - Diversos desafios teóricos

Escalabilidade em Sistemas Distribuídos

- Scale Out - desafios:
  - Escalabilidade:
    - Necessidade de aplicar as técnicas de computação distribuída e superar as barreiras para conseguir atender a número crescente de clientes
  - Tolerância a falhas:
    - Capacidade de um sistema se manter no ar a despeito de problemas, isto é, de ser tolerante a faltas
    - Implica em redundância, o que fatidicamente implica em **distribuição** e em Sistemas Distribuídos
- Conclusão:
  - principais razões para se desenvolver sistemas distribuídos são alcançar **escalabilidade** e **tolerância a falhas**, ambas resultantes da **agregação** do poder computacional de múltiplos componentes

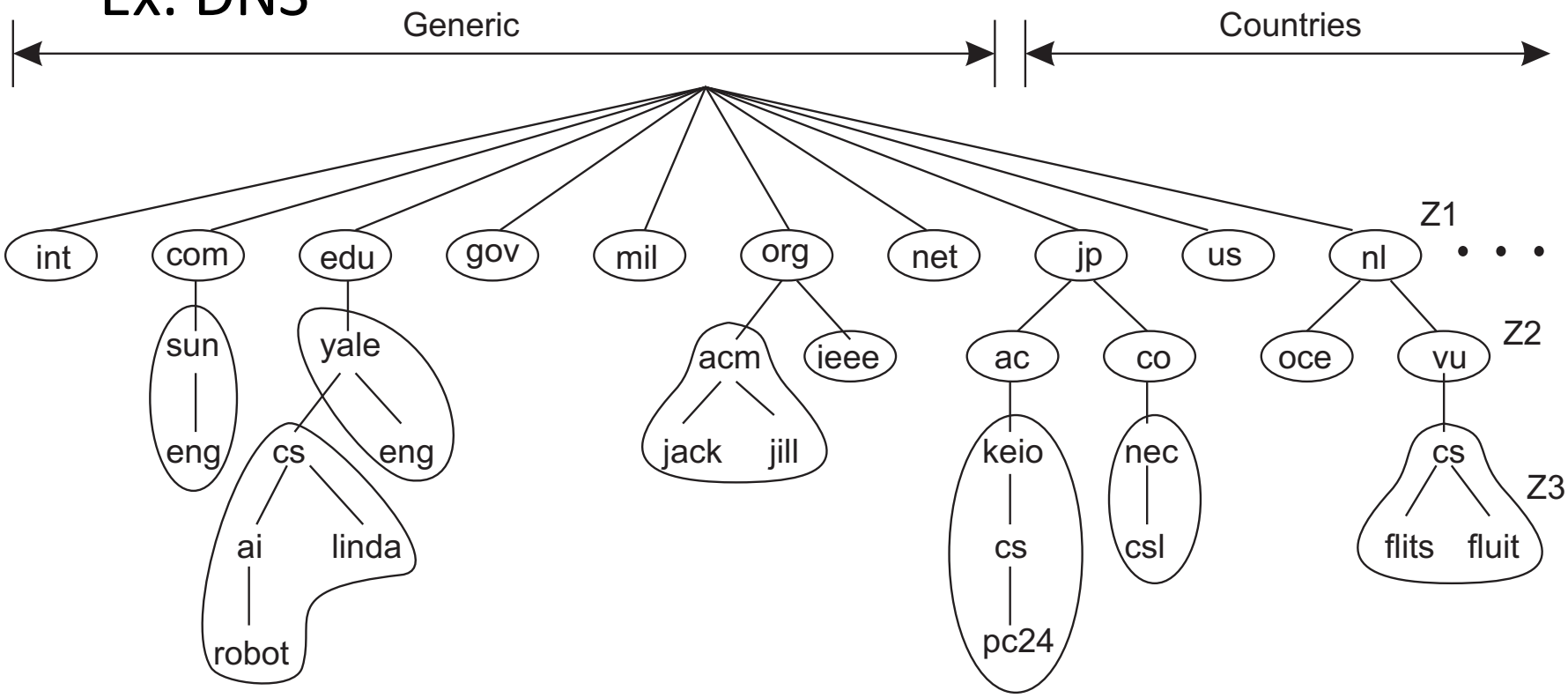
Técnicas para prover escalabilidade

- Mover a computação para o cliente:



Técnicas para prover escalabilidade

- Particionar dados e computação em diversas máquinas
- Ex. DNS



Técnicas para prover escalabilidade

- Outros exemplos de técnicas:
  - Bancos de dados e servidores de arquivos replicados
  - Web sites espelhados
  - Caches em navegadores e proxies
  - Caches de arquivos (servidores e clientes)

Desafios relacionados à replicação

- Principais dificuldades:
  - Múltiplas cópias levam a inconsistências: como lidar com escritas e caches?
  - Manter cópias consistentes requer sincronização global a cada modificação do estado
  - Sincronização global previne soluções de larga escala

- Observação
  - Se um sistema pode tolerar inconsistências, a necessidade de sincronização global pode ser reduzida:
    - Depende da aplicação

Desafios no desenvolvimento de sistemas distribuídos

- Muitos sistemas se tornam complexos sem necessidade:
  - Muitas vezes devido a “consertos” de erros detectados
- Algumas suposições falsas durante o desenvolvimento:
  - A rede é confiável
  - A rede é segura
  - A rede é homogênea
  - A topologia não muda
  - A latência é zero
  - A banda é infinita
  - Existe apenas um administrador

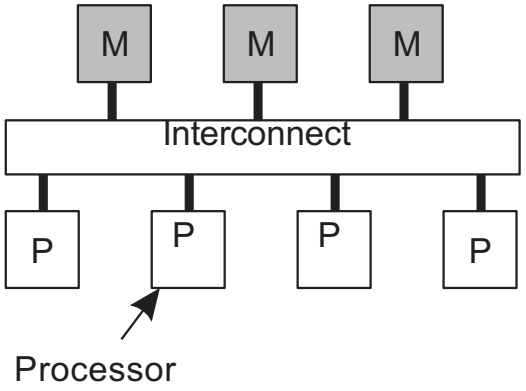
Tipos de Sistemas Distribuídos

- Sistemas de computação (de alta performance)
- Sistemas de informação
- Sistemas de computação pervasiva

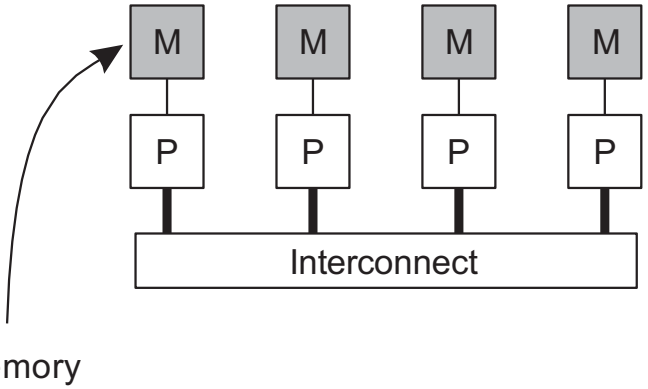
Sistemas de Computação (de alta perf.)

- Teve início com a computação paralela:
  - Computadores multiprocessadores

- Memória Compartilhada

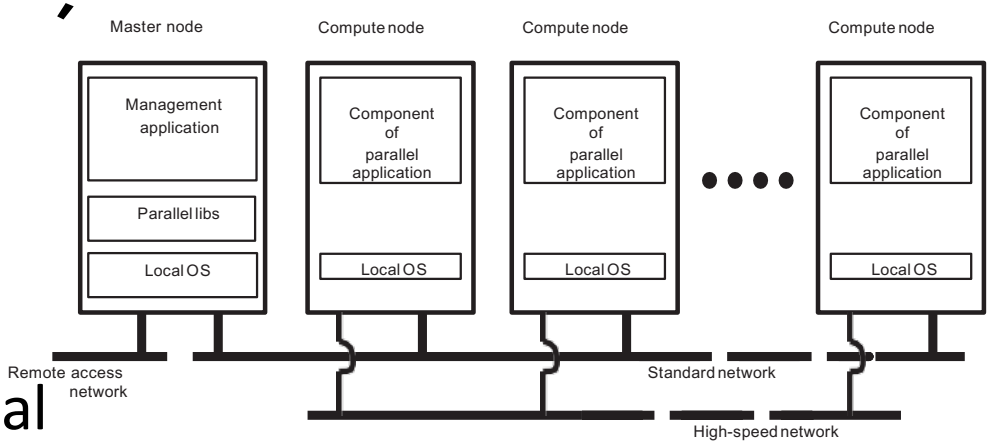


Memória Privada

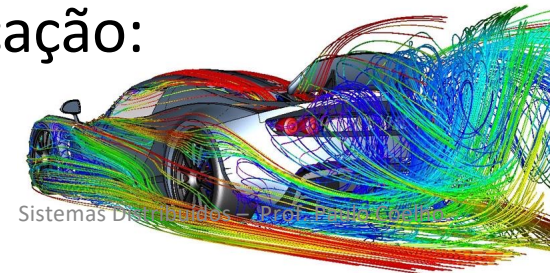


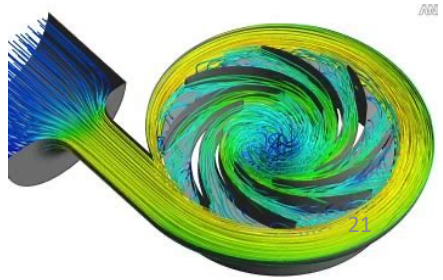
- Nunca atingiu expectativa dos programadores:
  - pouco usada hoje em dia

Sistemas de Computação



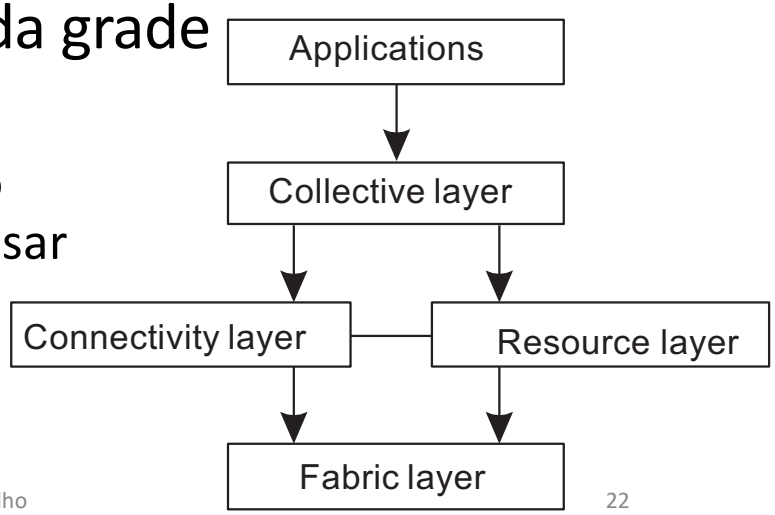
- ***Clusters:***
  - Coleção de nós similares
  - Mesmo sistema operacional
  - Conectados por rede de alta velocidade
  - Geralmente compartilhados por pesquisadores resolvendo problemas de áreas como bioinformática, engenharia, economia e inteligência artificial
  - **Fortemente acoplados:**
    - grande dependência dos componentes uns nos outros, tanto na administração quanto na aplicação, e se um dos componentes para de funcionar, normalmente os outros também param
  - Exemplos de aplicação:





Sistemas de Computação

- **Grids**:
  - Federação de sistemas computacionais
  - Diferente domínios administrativos
  - Componentes **fracamente acoplados**
  - Muito usadas até meados da década passada
  - Membros de uma associação disponibilizam capacidade computacional a um *pool*.
  - Hardware, software e tecnologias de rede podem ser diferente entre os componentes da grade
  - Exemplo:
    - SETI@home: pessoas doavam tempo ocioso do seu computador para analisar sinais de rádio recebidos do espaço.

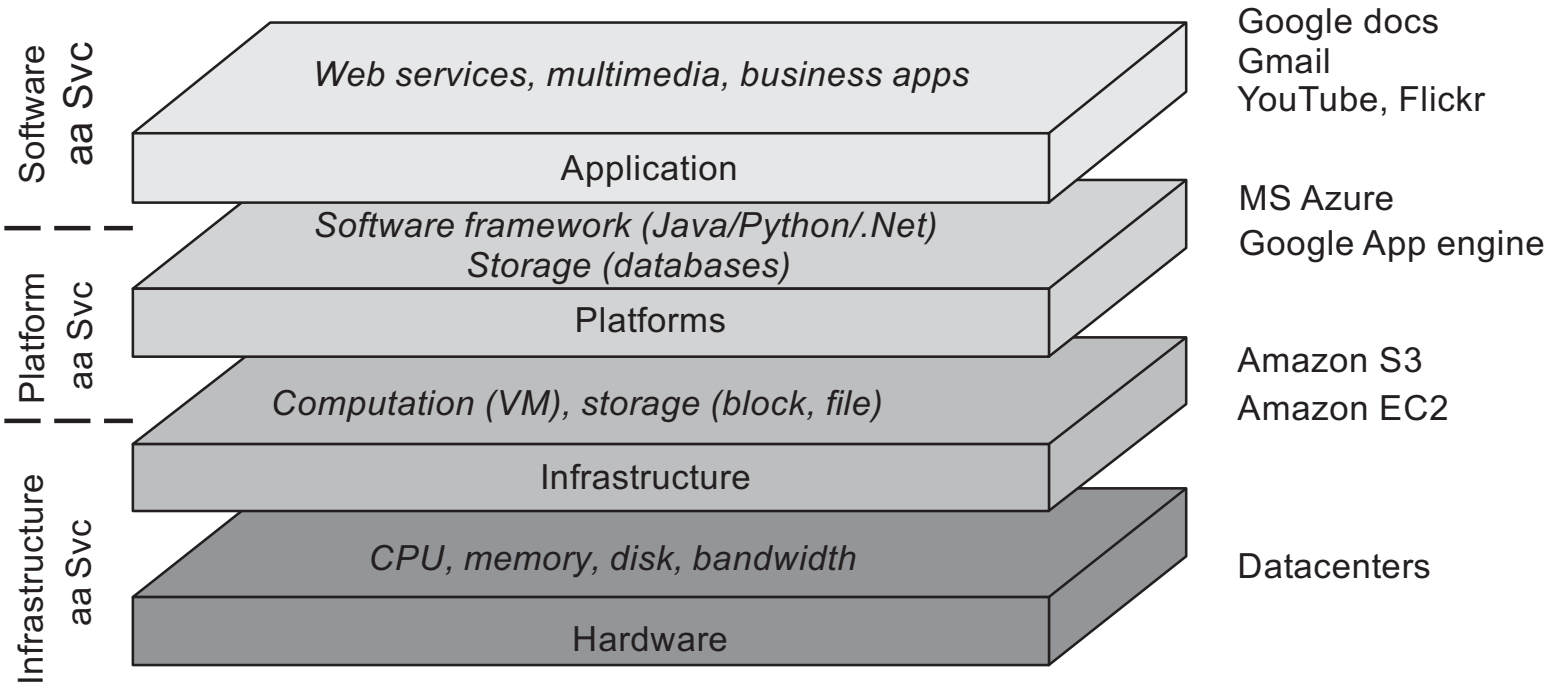


Sistemas de Computação

- **Cloud**:
  - Modelo de **computação utilitária:**
    - Fornecimento de recursos computacionais por provedores em troca de um pagamento **proporcional** à quantidade de recursos utilizados
    - Similar a fornecimento de água ou eletricidade
    - Facilidade para construir infraestrutura
  - Infraestrutura para outros sistemas distribuídos
  - Complexas peças de engenharia com diversos subsistemas:
    - sincronização de relógios
    - monitoração de falhas, tolerância a falhas e coleta de logs
    - roteamento eficiente
    - movimentação de recursos virtualizados para consolidação de recursos físicos
    - armazenamento redundante de dados

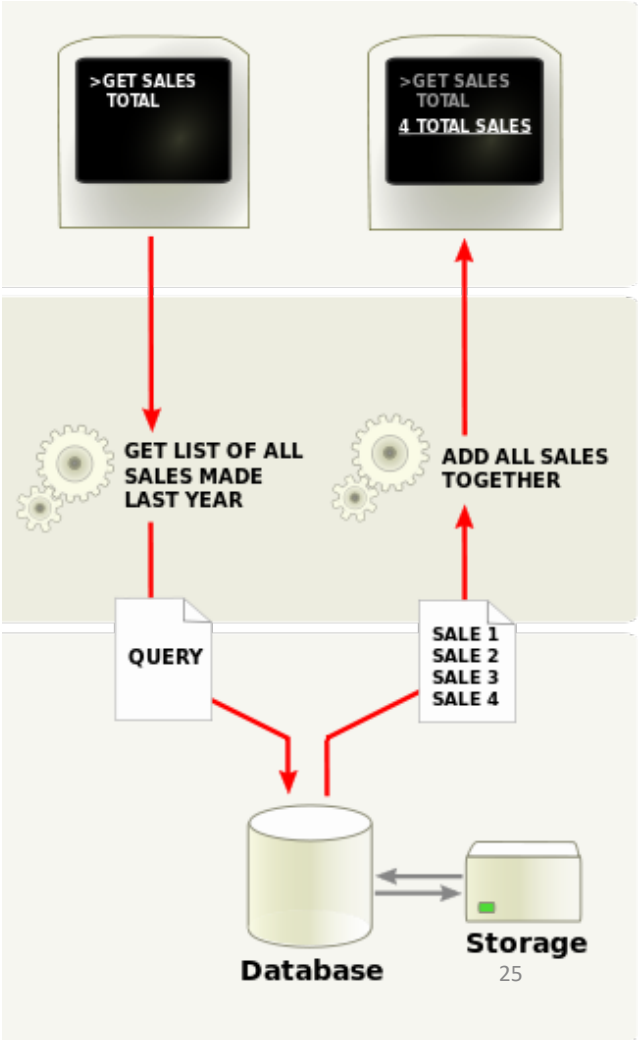
Sistemas de Computação

- **Cloud**:



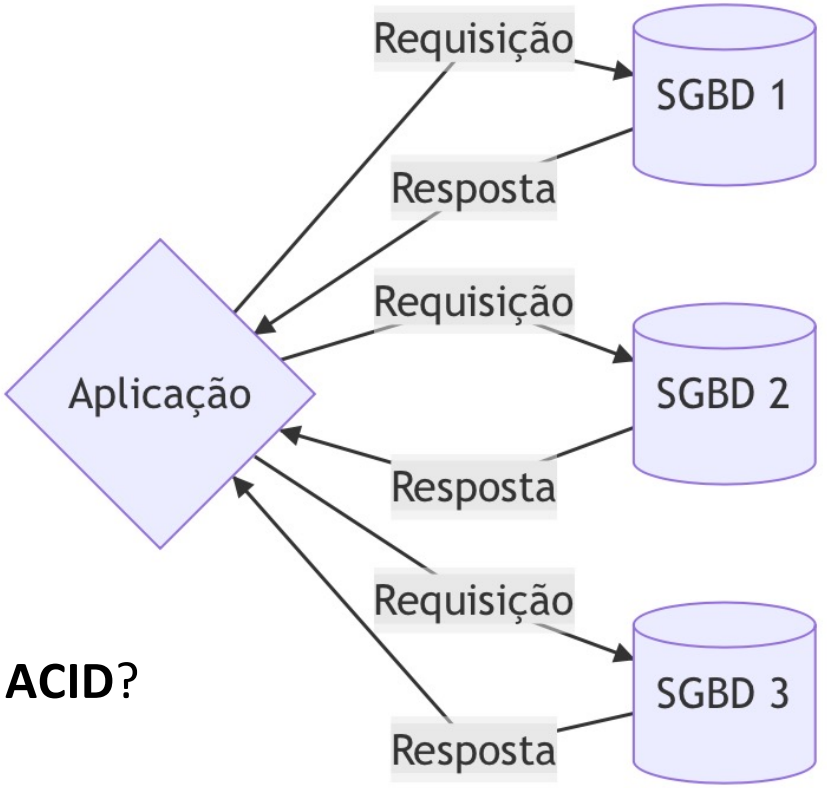
Sistemas de Informação

- Encontrados em diversas formas
- Termo é muito abrangente:
  - dificilmente um sistema distribuído não estaria nesta classe.
- Exemplo:
  - arquitetura em 3 camadas:
    1. implementa a interface com o usuário
    2. contém a lógica do negócio
    3. mantem os dados da aplicação



Sistemas de Informação

- Bancos de dados na terceira camada são frequentemente distribuídos:
  - Necessidade de coordenação
- Exemplo:
  - SGBD1: dados dos clientes
  - SGBD2: dados do estoque
  - SGBD3: ordens de compras
    - O que significa fazer uma compra?
    - O que pode dar errado?
    - Como implementar propriedades **ACID**?



S.I. - Integração de Aplicações

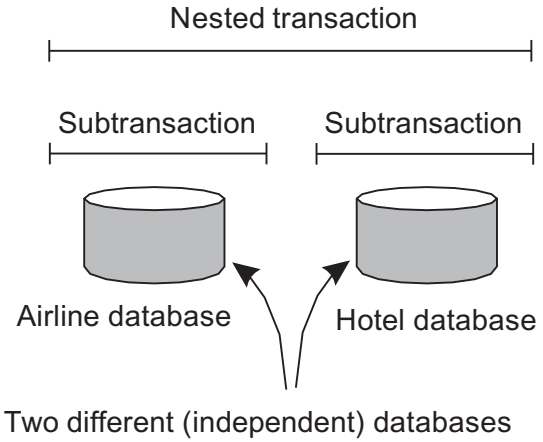
- Frequentemente é necessário integrar aplicações:
  - sistemas de informação legados com sistemas mais modernos
  - Expor sistemas usando uma interface mais moderna
- O ***middleware*** é o element capaz de encapsular esta integração

S.I. - Integração de Aplicações: exemplo

- Transações aninhadas (nested transaction)

Primitiva	Descrição
BEGIN TRANSACTION	Marca o início de uma transação
END TRANSACTION	Termina a transação e tenta fazer o *commit*
ABORT TRANSACTION	Encerra a transação e restaura valores antigos
READ	Lê de arquivo, tabela, ou similar
WRITE	Escreve dados em arquivo, tabela, ou similar

Problema: tudo-ou-nada

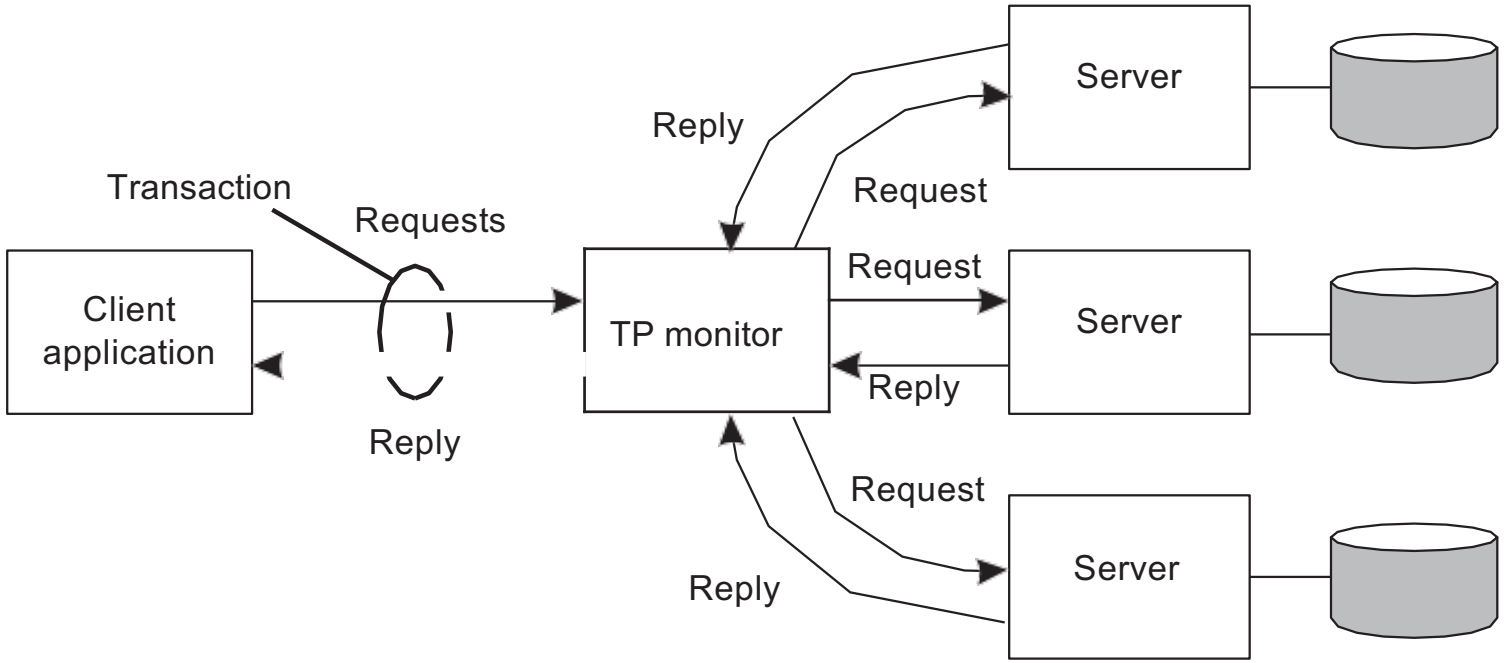


- **Atômico**: indivisível
- **Consistente**: não viola invariantes
- **Isolado**: sem interferência mútua
- **Durável**: commit significa mudanças permantes

Como garantir estas propriedades?

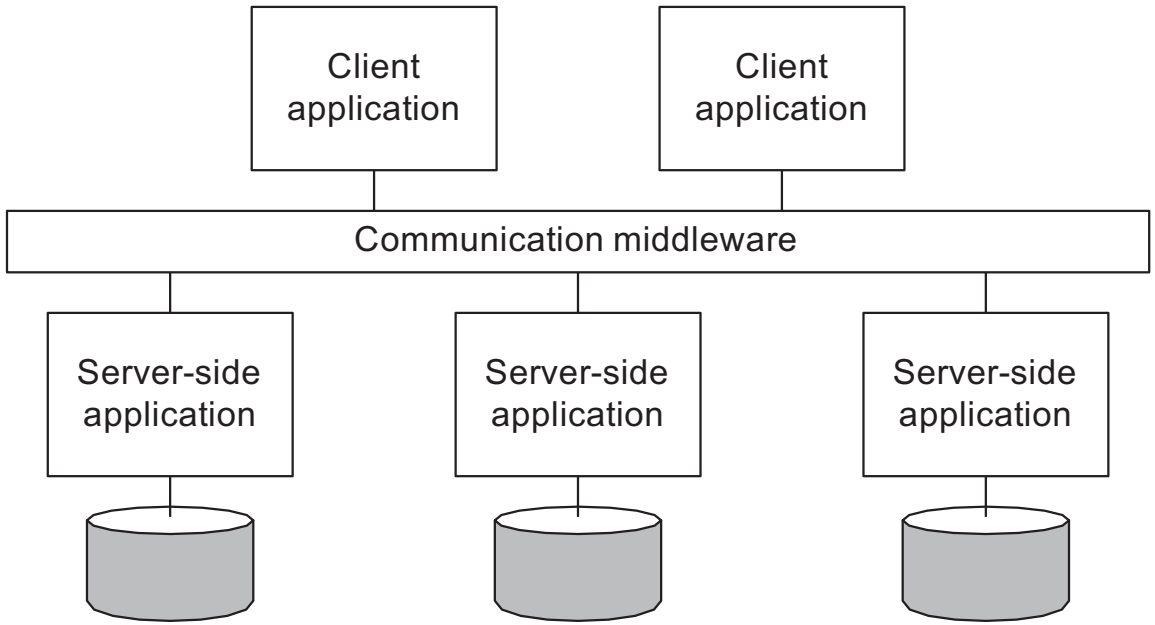
S.I. - Integração de Aplicações: exemplo

- TPM: Transaction Processing Monitor



S.I. - Integr. de Aplicações: Middleware

- Uma camada de software que se interpõe entre os clientes e um serviço oferecido.
  - Pode se expor via interface REST para os clientes, mas consultar o sistema legado em um padrão antigo
  - Pode agregar subsistemas de diversos departamentos de uma empresa via troca de mensagens.



S.I. - Integr. de Aplicações: Middleware

- Oferece facilidades de comunicação:
  - *Remote Procedure Call* (RPC):
    - Chamada local, empacotamento, envio, processamento, recebimento do retorno, desempacotamento e retorno da chamada "como se fosse local"
- *Message Oriented Middleware* (MOM):
  - Mensagens enviadas (**publicadas**) para **pontos de contato lógico** e encaminhadas para aplicações **subscritas**
  - Aplicações que publicam e se subscrevem não precisam saber da existência do outro, nem estar disponível ao mesmo tempo
  - Aumento de escalabilidade e capacidade de tolerar falhas.

Sistemas Pervasivos

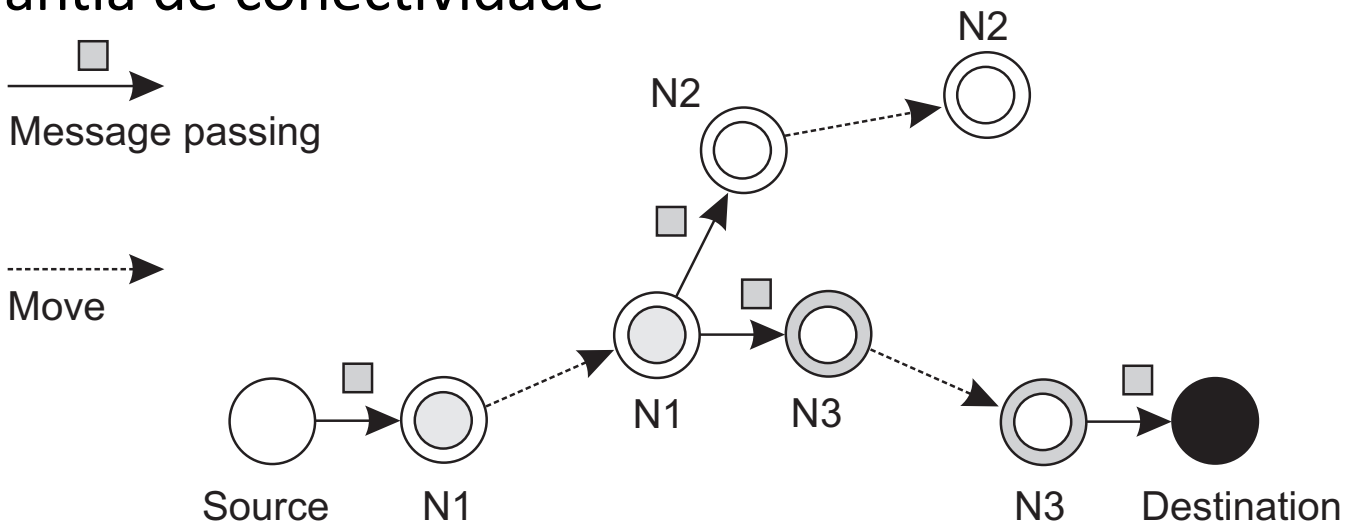
- Sistemas distribuídos emergentes em que os nós geralmente são pequenos, móveis e embarcados em um sistema maior
- Mistura-se ao ambiente e dia-a-dia do usuário
- 3 subtipos:
  - Sistemas Ubíquos:
    - Pervasivo e presente: contínuo interação entre sistema e usuário
  - Sistemas Móveis:
    - Pervasivo, mas ênfase na mobilidade
  - Redes de sensores:
    - Pervasivo, com ênfase no monitoramento (e atuação) do ambiente

Sistemas Ubíquos

- Características principais:
  - Distribuição: dispositivos com acesso à rede, distribuídos, acessíveis e forma transparente
  - Interação: não obstrusiva (usuários e dispositivos)
  - Sensível ao contexto: otimiza interação com base no contexto do usuário
  - Autonomia: dispositivos operam de maneira autônoma e autogerenciável
  - Inteligência: o sistema como um todo é capaz de manipular ações e interações de maneira dinâmica

Sistemas Móveis

- Características principais:
  - Grande conjunto de dispositivos móveis: smartphones, tablets, GPS, controles remotos, …
  - Localização do dispositivo muda ao longo do tempo ⇒ mudança de serviços, alcançabilidade, etc.
    Palavra-chave: **descoberta**
  - Rede tolerante a interrupção: sem rota estável ou garantia de conectividade



Redes de sensores

- Nós geralmente são:
  - Muitos: dezenas a milhares
  - Simples:
    - Pouca memória
    - Baixa capacidade de processamento
    - Baixa capacidade de comunicação
  - Alimentados por baterias
    ou até sem baterias em alguns casos

Redes de sensores

- Dois extremos: