Sistema de Software Distribuído

- É composto por uma **sequência** de instruções, que é interpretada e executada por um processador
- É composto por instruções concorrentes ou paralelas, que são interpretadas e executadas por um ou mais processadores
  - Interconectados por uma rede
  - Geograficamente separados

- **Por que distribuir?**

Por que distribuir?

Manutenção

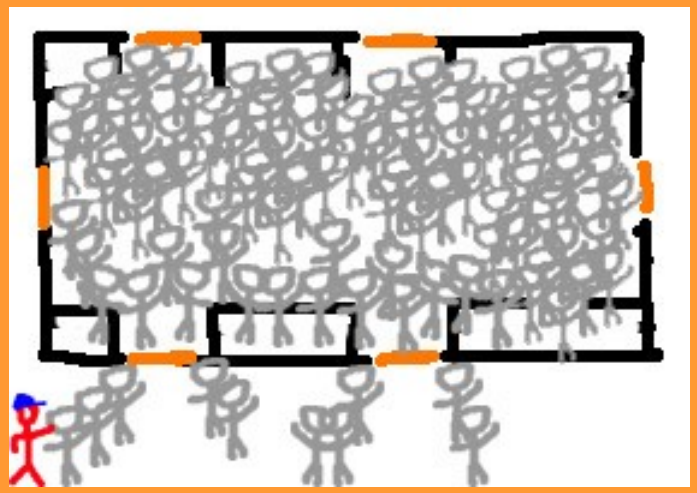
Dividir para conquistar

Por que distribuir?

Acomodação às capacidades das máquinas disponíveis

Pode ser “impossível” instalar e executar um sistema de grande porte em uma única máquina

Ou muito caro!



- Manutenção
  - Dividir para conquistar

Por que distribuir?

Melhor desempenho e Escalabilidade

Concorrência, paralelismo

- Manutenção
  - Dividir para conquistar
- Acomodação às capacidades das máquinas disponíveis
  - Pode ser "impossível" instalar e executar um sistema de grande porte em uma única máquina

Por que distribuir?

Tolerância a falhas

Analogia-exemplo: pesquisas na Guerra Fria ➡ Internet

- Manutenção
  - Dividir para conquistar
- Acomodação às capacidades das máquinas disponíveis
  - Pode ser "impossível" instalar e executar um sistema de grande porte em uma única máquina
- Melhor desempenho e Escalabilidade
  - Concorrência, paralelismo

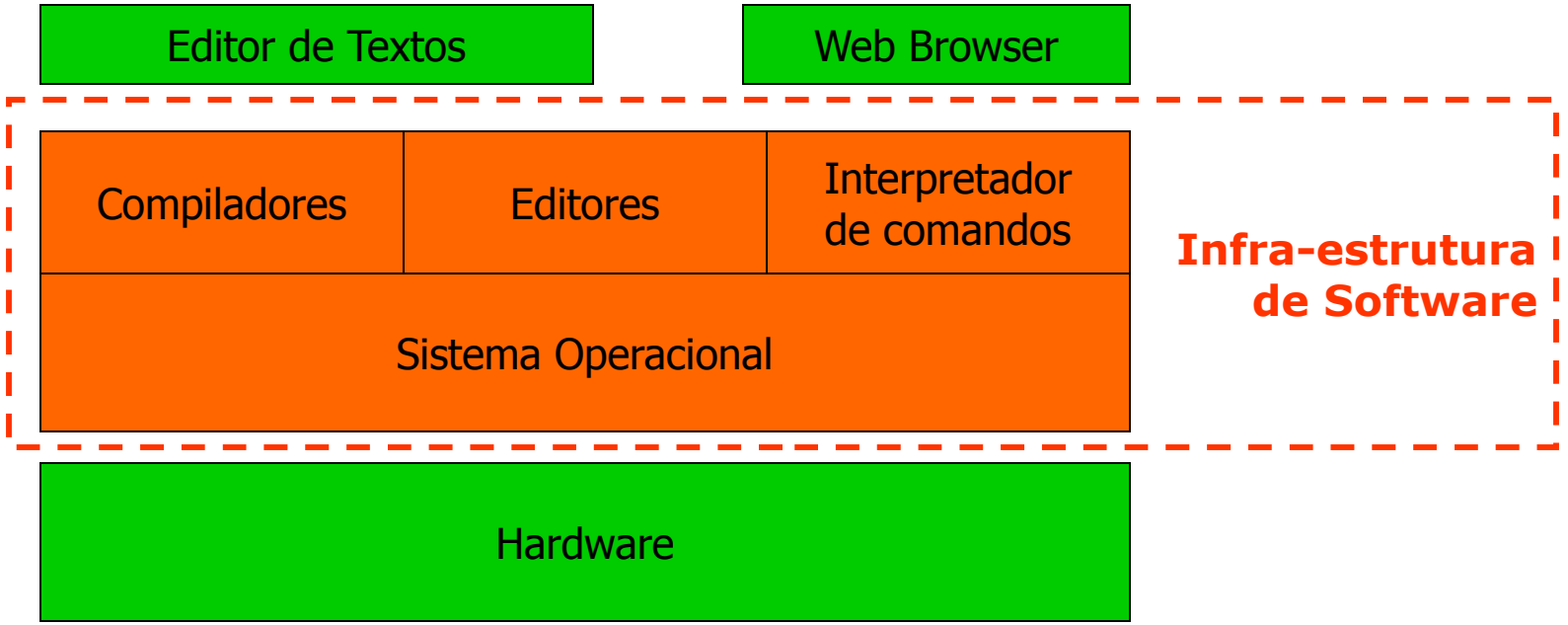
Requisitos Ligados a Sistemas Distribuídos

- **Comunicação em redes** – controle de sessões, ...
- **Coordenação** – sincronização, ...
- **Confiabilidade** – transmissões, replicação
- **Escalabilidade** – transparência de localização, replicação, ...
- **Heterogeneidade** – hardware, plataformas de SO, linguagens de programação

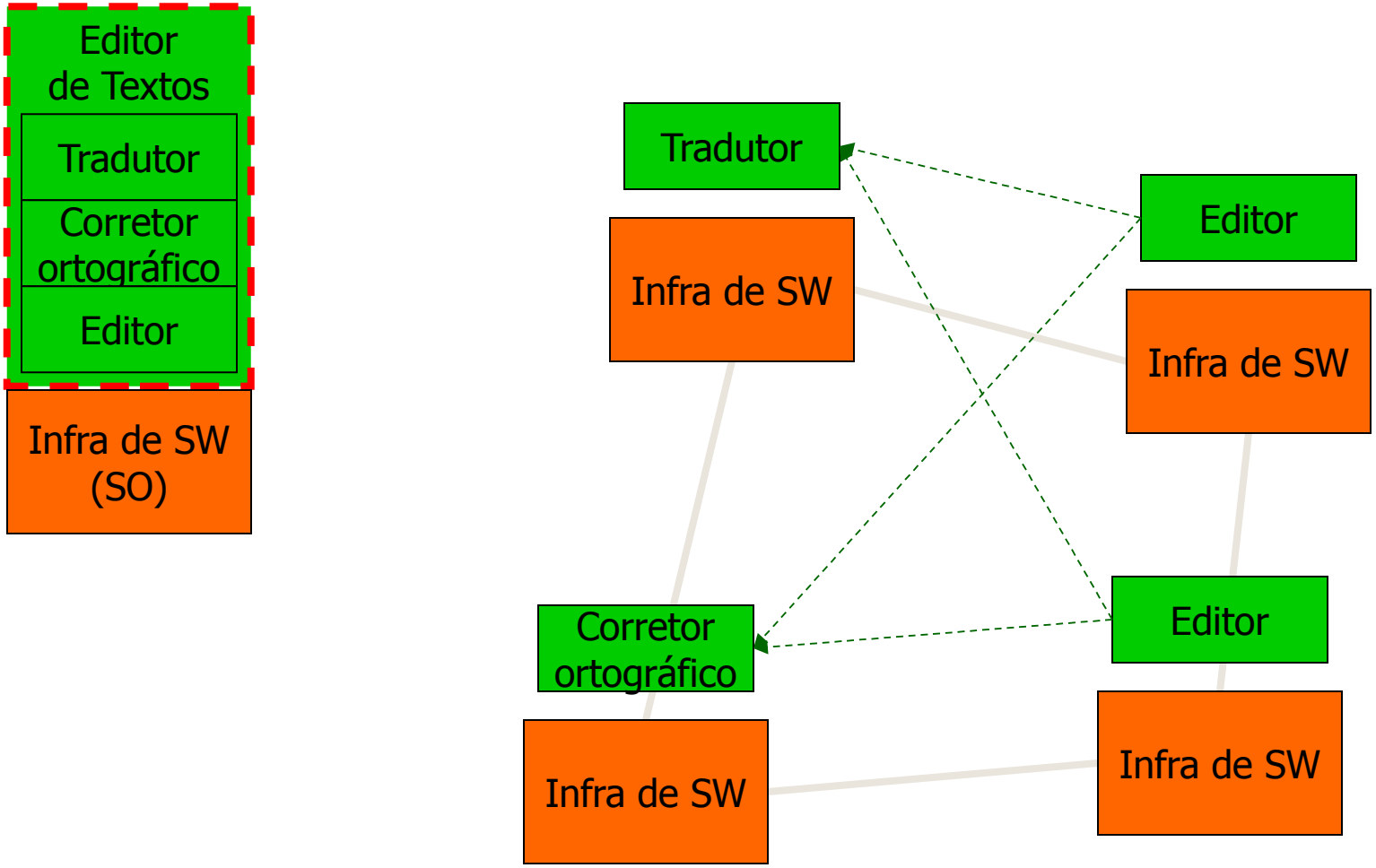
➢ Caro demais se os desenvolvedores de aplicação tiverem que resolver tudo isto!

Infra-estrutura de Software

- Software responsável pelo controle direto e gerência do hardware e operações básicas de sistema
- Fornece a base para a execução de software de aplicação



Infra-estrutura de Software para Aplicações Distribuídas



Sistemas Distribuídos

Conceitos e Arquiteturas

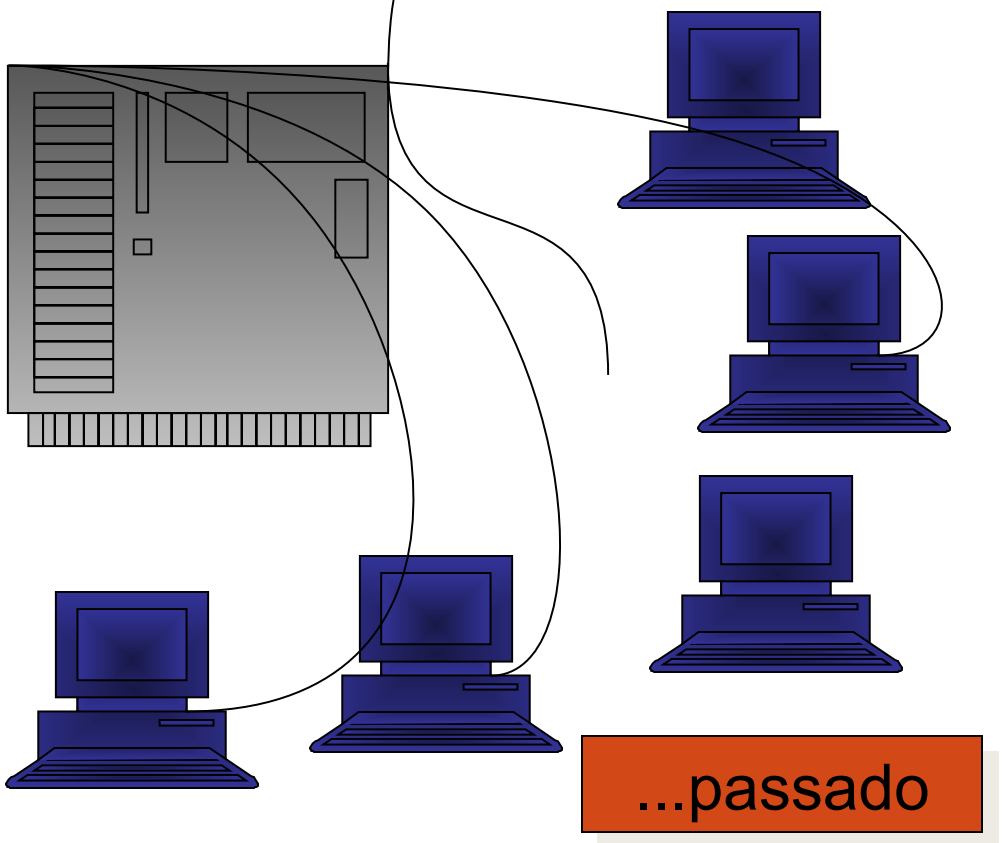
Sistemas Centralizados

Motivação

Computadores com grande capacidade de processamento *(mainframes)*

- Grande porte físico limitações para acomodação
- Grande consumo de energia sala especial, refrigeração
- SO único dependência do fabricante
- **Grande custo**

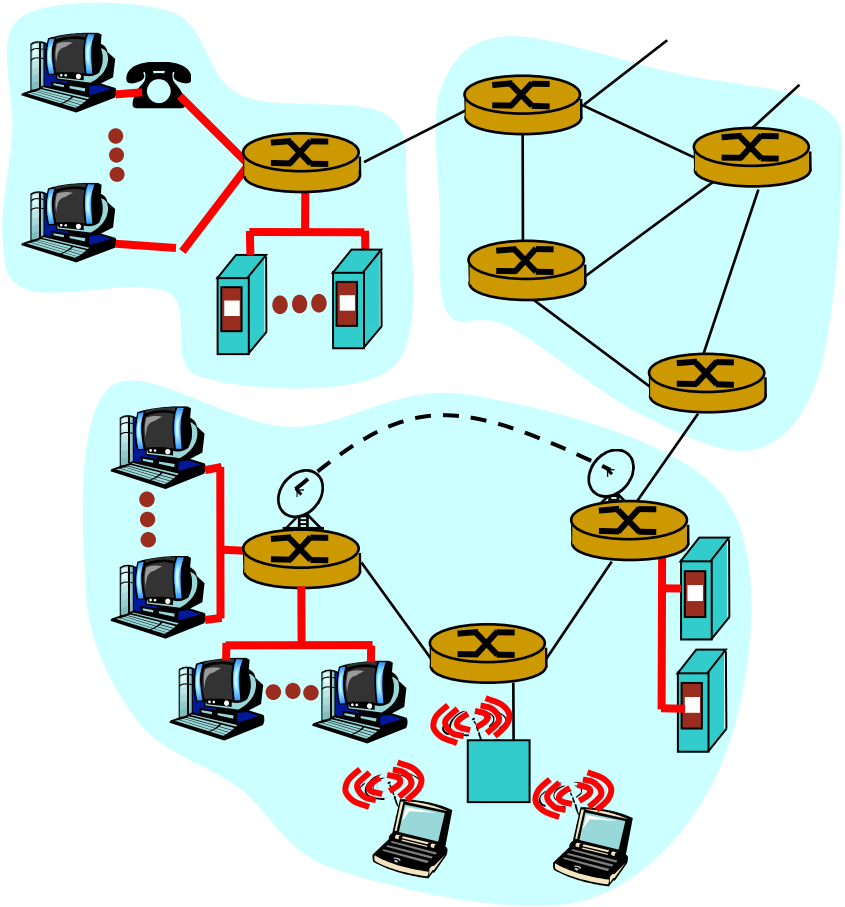
Terminais sem capacidade de processamento (“burros”)



Sistemas em Rede

A realidade dos últimos tempos...

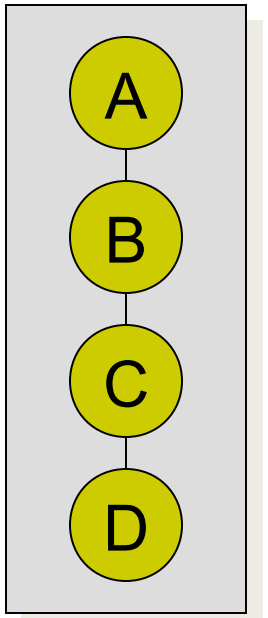
- Computadores diversos, todos com capacidade de processamento
  - Portes diversos
  - SOs diversos
  - Redes diversas (Ethernet, ATM, com fio, sem fio...)
  - Internet



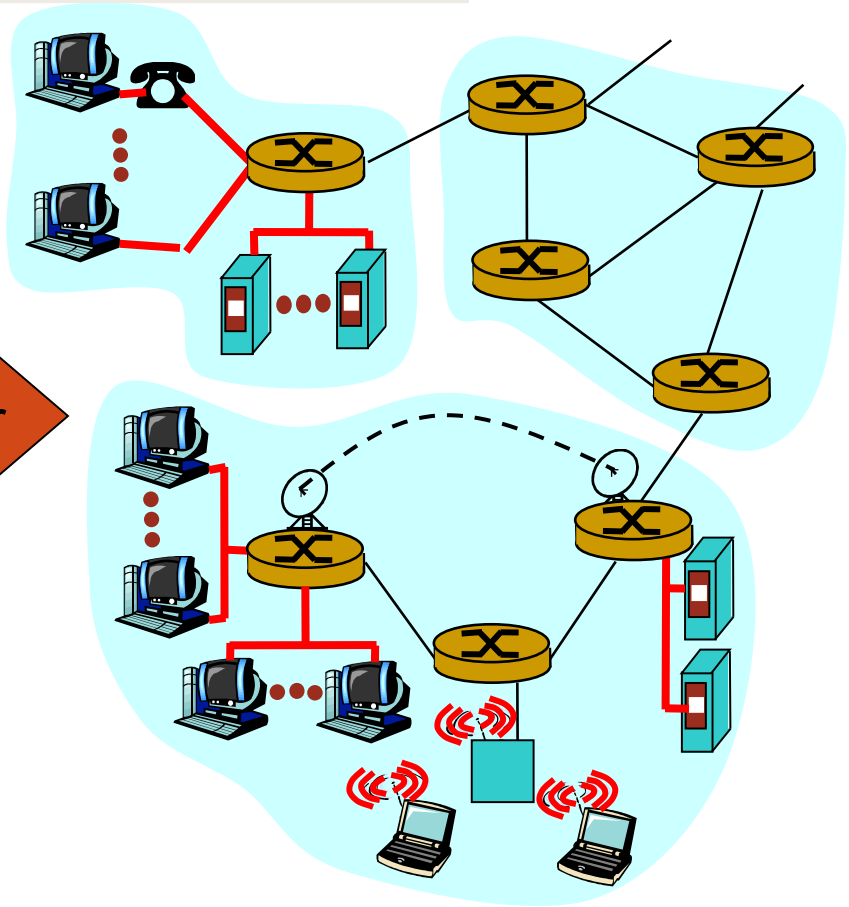
Distribuição de sistemas

Dividindo para conquistar!!!

Programa modularizado



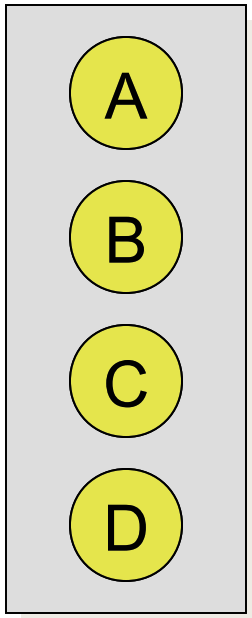
distribuir



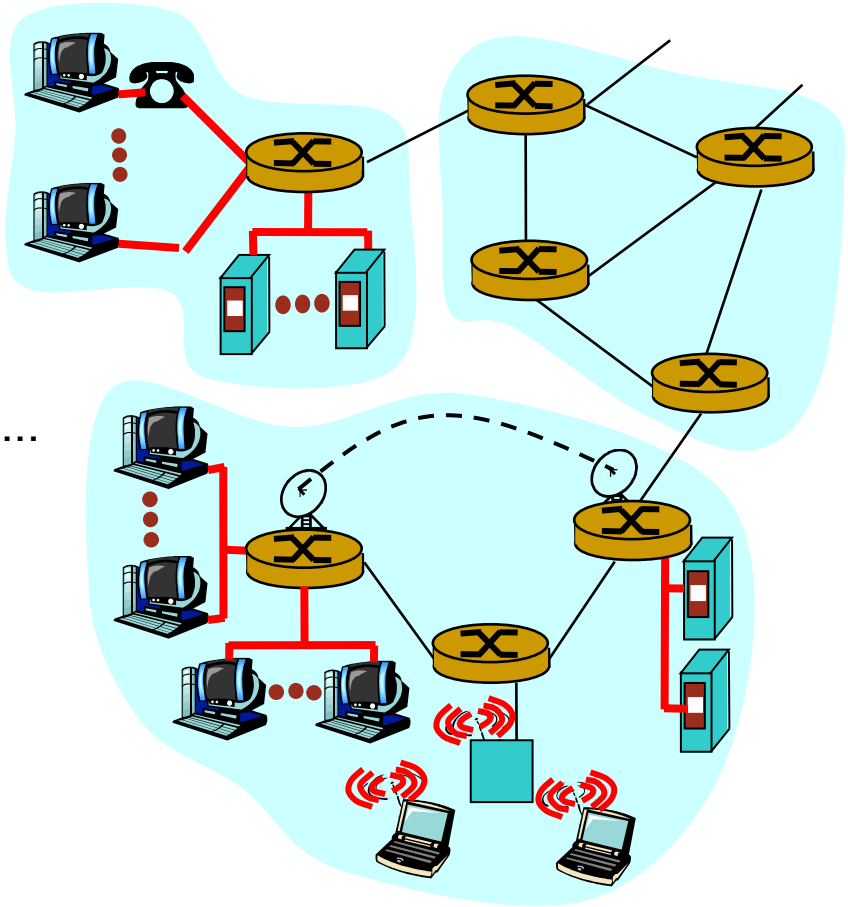
Execução sequencial ou concorrente (*threads*)

Dividindo para conquistar!!!

Programa modularizado



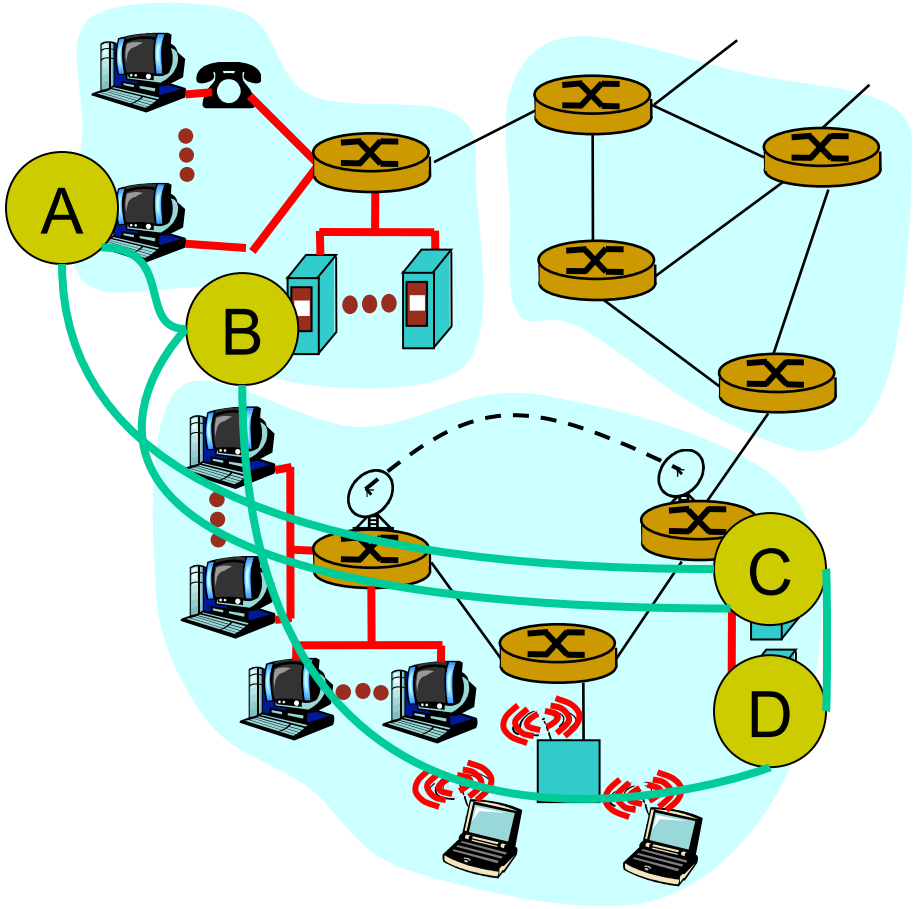
Distribuindo...



Dividindo para conquistar!!!

Programa distribuído

- Componentes interligados (**comunicação**)
- Processamento (**computação**) distribuído ou paralelo



Potencial para ser mais poderoso do que um sistema centralizado, convencional

- Pode ser mais **confiável**: toda função pode ser replicada
  - Quando um processador falha, outro pode continuar o trabalho
  - C*rash* não necessariamente resulta em perda de dados
- Várias computações podem ser realizadas em **paralelo**:

Pode-se considerar **tolerância a falha** e possibilidade de **paralelismos** como as propriedades fundamentais de um sistema distribuído

Definições

- Lamport: “Um sistema distribuído é aquele que faz você parar de ter o trabalho realizado quando uma máquina da qual você nunca ouviu falar falha”

- Mais seriamente, Tanenbaum e van Renesse (1985):

Um sistema (operacional) distribuído é aquele que aparece para os usuários como um sistema (operacional) centralizado ordinário, mas que executa em múltiplas CPUs independentes.
O conceito chave é *transparência*, ou seja, o uso de múltiplos processadores deve ser invisível (transparente) para o usuário.
Pode-se dizer que o sistema é visto como um “uniprocessador virtual”, e não como uma coleção de máquinas distintas.

Definições mais recentes

- "Coleção de **computadores independentes** que aparecem para os usuários do sistema como um **único computador**." (**Tanenbaum & van Steen**)

- "Um sistema em que componentes de *hardware* e *software* localizados em computadores em rede **se comunicam e coordenam suas ações** por passagem de mensagens." (**Coulouris** *et al*)

- "Uma coleção de elementos de processamento interconectados, tanto logicamente como fisicamente, **para execução cooperativa de programas** de aplicação com o controle geral dos recursos centralizado." (**M. Eckhouse**)

- **Vários componentes**
- **Conectados via uma rede**
- **Compartilhando recursos**
- **Transparência**

Por que construir SDs?

- **Desempenho/custo**
- **Modularidade**
  - Assunto bastante explorado recentemente
- **Expansibilidade/Abertura**
  - Sistemas distribuídos são capazes de crescimento incremental
- **Compartilhamento de informações e recursos**

Por que construir... (cont)

- **Escalabilidade**
  - Mais fácil aumentar a capacidade do sistema através da introdução de mais recursos
- **Confiabilidade e Disponibilidade**
  - Redundância e Replicação
  - O sistema deve ser capaz de se recuperar de falhas

Complexidade

- Limita o que pode ser construído
- Schroeder chama os problemas causados pela complexidade de **problemas de sistema**:
  - **Interconexão**: devido a componentes que não foram feitos para funcionar de forma combinada
  - **Interferência**: dois componentes de um sistema podem exibir comportamento indesejável ou incompatível quando combinados
  - **Propagação de efeito**: “efeito cascata” de falhas pode derrubar um sistema inteiro se isso

Problemas de sistema (cont)

- **Efeitos de escala**:
  - Um sistema que funciona bem com 10 nós pode falhar se crescer para centenas de nós
- **Falha parcial**
  - Um grande diferencial de sistemas distribuídos em relação a sistemas centralizados
- **Ausência de um relógio global**
  - Poucas ou nenhuma garantia(s) de tempo
  - Outro grande diferencial dos sistemas distribuídos

Requisitos não-/funcionais como fonte de complexidade

- Sistemas distribuídos são complexos porque o que eles têm que fazer é complexo
- Exemplos:

1. **Gerenciamento do escalonamento de trens**
   - passageiros têm que trocar de trens para chegar em seus destinos – o problema da sincronização

1. **Sistema de arquivos distribuídos**
   - Autenticação, controle de acesso, controle de concorrência etc.
   - Complexidade ainda maior quando há os requisitos de alta disponibilidade e tolerância a falhas

Necessidade econômica como fonte de complexidade

- A solução simples nem sempre pode ser usada – às vezes é cara demais!
- Exemplo:
  - Em uma rede de longa distância, **interconectar todos os pontos** (nós) seria a solução mais simples, porém extremamente cara!
  - A solução mais barata é fazer uma rede em que **todos os nós são alcançáveis**, porém não necessariamente de forma direta
    - O custo dessa solução é mais baixo, porém a **complexidade aumenta consideravelmente**

Conceito-chave

- **Transparência**

- Distribuição
- Comunicação
- Complexidade
- Heterogeneidade

Sistemas Distribuídos

Características, Objetivos

e

Modelos Arquiteturais

Características

- Conjunto de máquinas autônomas
- Interconectadas por canais de comunicação
- Comunicando-se por troca de mensagens
- Independência de falhas (falhas parciais)

- Ausência de relógio global
- Ausência de estado global
- Estado compartilhado da aplicação (através de comunicação)

Objetivos de um Projeto de SD

- Eficiência – alto desempenho (latência, *throughput*, etc.)
- Robustez – resistência a falhas
  - Disponibilidade: o sistema está no ar quando preciso (*instante de tempo*)
  - Confiabilidade: o sistema não falha por um longo *período de tempo*
    - E quando falha, a falha não provoca uma "catástrofe"
- Consistência – mesma visão de dados
- **Transparência**
- **Abertura**
- **Escalabiliade**

Tipos de Transparência

- **Localização**: esconde onde o recurso está localizado
- **Acesso**: operações idênticas (ou muito similares) para acesso local e remoto
- **Falha**: esconde a falha e recuperação de um recurso
- **Replicação**: esconde de usuários ou programadores de aplicação a existência de réplicas de recursos
- **Migração**: esconde que um recurso pode se mover para outra localização
- **Temporal**: comunicação não necessariamente sincronizada
- **Concorrência**: compartilhamento de recursos sem interferência entre processos concorrentes

Obstáculos à Transparência

- Latência
- Modelo diferente de acesso à memória
- Concorrência
- Falhas parciais

Abertura

- Capacidade de um sistema de ser estendido (hw, sw)
  - e de interoperar com outros sistemas
- Resulta da especificação de interfaces padrão e de torná-las públicas
- Especificações podem ser
  - padrões estabelecidos por organização de padronização
  - padrões estabelecidos pelo uso (de fato)

Escalabilidade

- Mais recursos => mais capacidade
  - Mais ou menos na mesma proporção
- Exemplos de limites de escalabilidade

Conceito	**Exemplo**
Serviços centralizados	Um único servidor para todos os usuários
Dados centralizados	Uma única lista telefônica on-line
Algoritmos centralizados	Roteamento baseado em informação completa

Modelos Arquiteturais

Cliente-Servidor

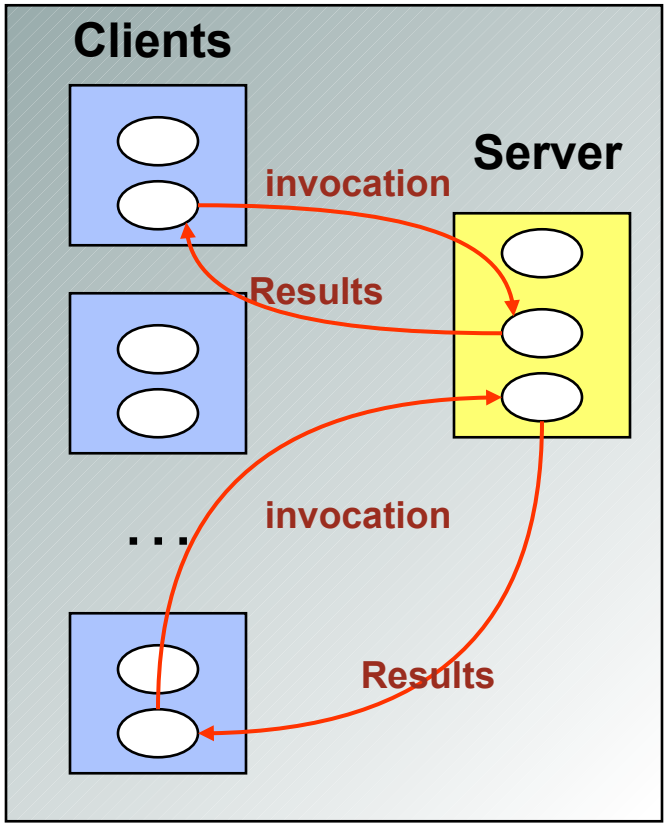
Peer-to-Peer

Objetos Distribuídos

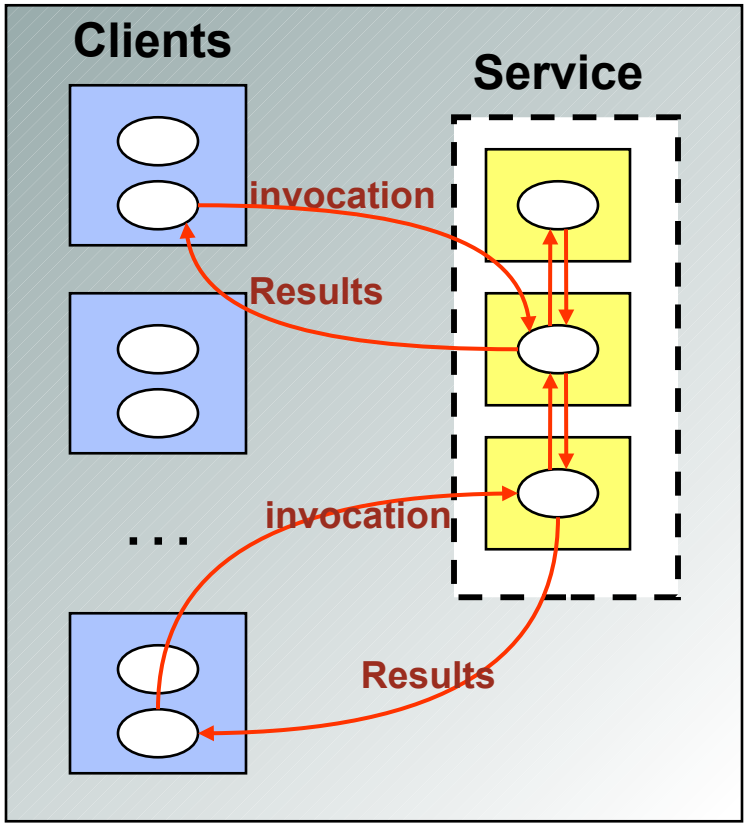
O que é um modelo arquitetural?

- Estrutura em termos de componentes especificados separadamente
- Inter-relações de componentes
- Divisão de responsabilidades entre componentes

Arquitetura Cliente-Servidor



Servidor Único



Múltiplos Servidores

C/S: Aspectos positivos

- Arquiteturas cliente-servidor fornecem uma infra-estrutura versátil que suporta a inserção de novas tecnologias mais rapidamente

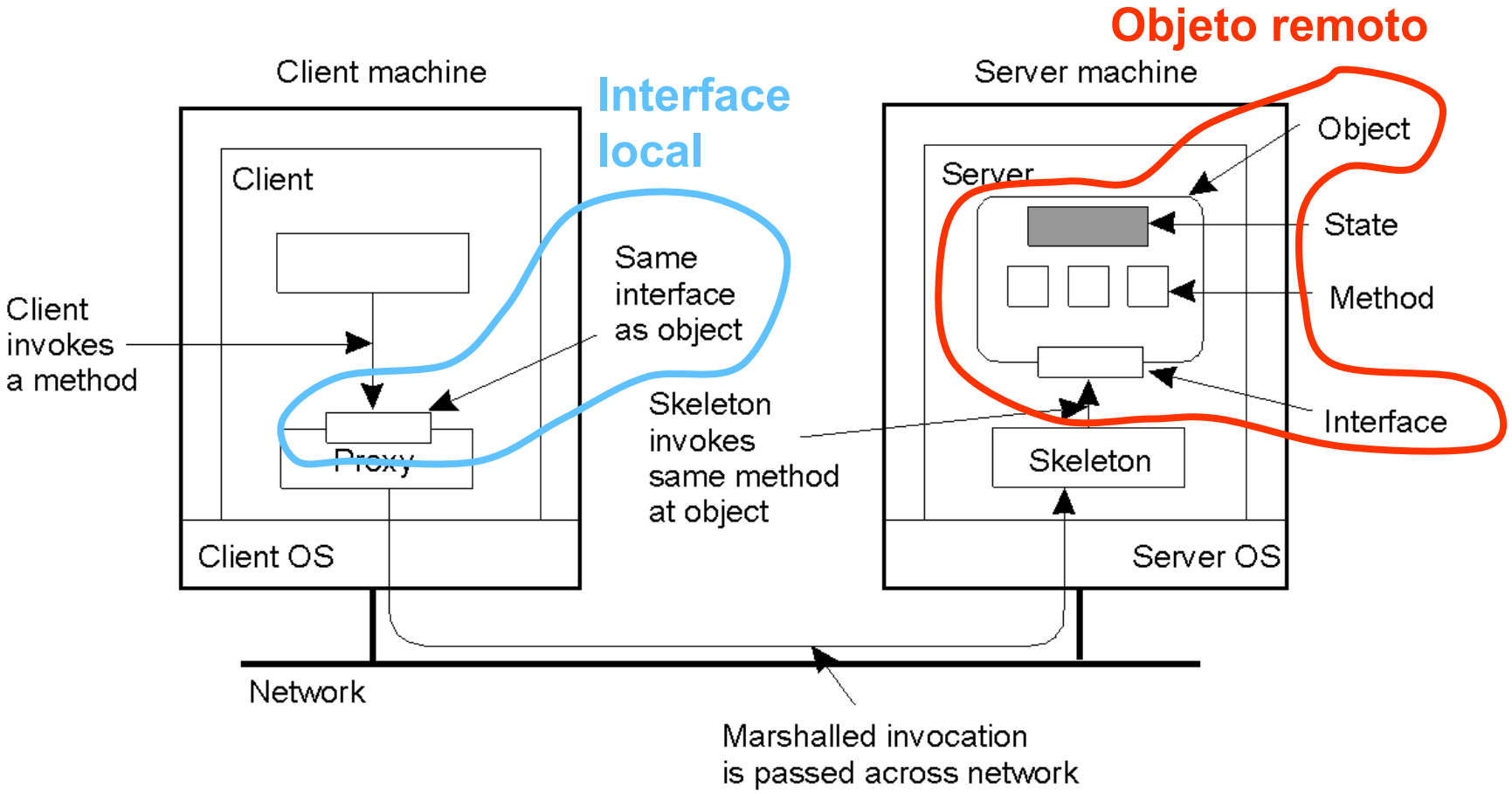
- Arquiteturas de software cliente-servidor têm sido usadas desde os anos 80 maturidade

Objetos Distribuídos

- Uma aplicação distribuída pode ser vista como um conjunto de objetos

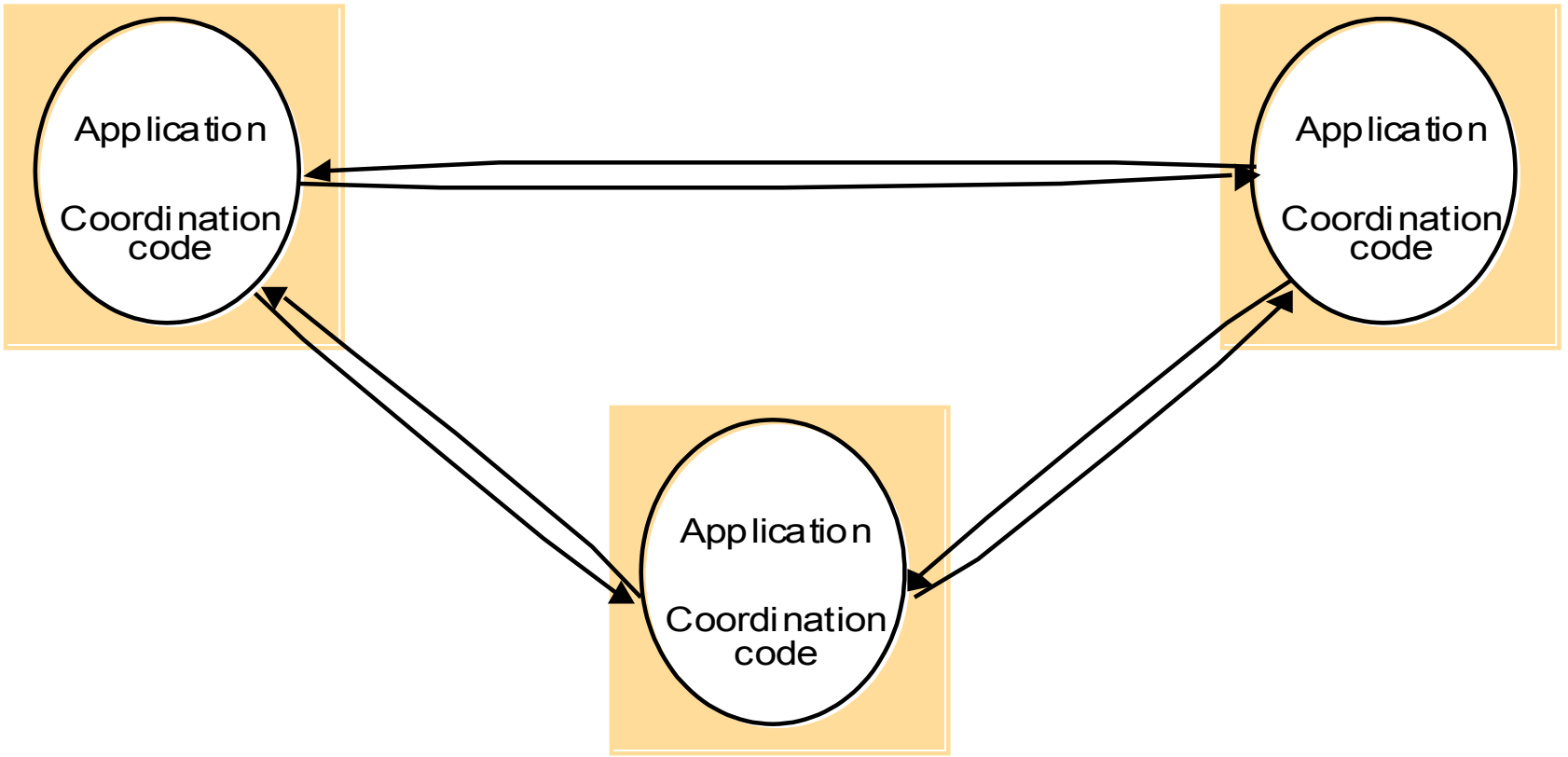
- Objetos:
  - Consistem em dados + código
  - Podem ser clientes, servidores ou ambos
  - Interface esconde detalhes de implementação
  - Modelar com objetos não implica no uso de programação orientada a objetos

Objetos Distribuídos



- Observe a separação entre interface e objeto

Peer processes (P2P)



Escolhendo arquitetura...

- Alguns *tradeoffs* devem ser considerados para selecionar a arquitetura mais apropriada, incluindo:
  - O crescimento potencial do número de usuários,
  - Custo e
  - Homogeneidade do ambiente computacional futuro e do momento
- Cliente-servidor é um modelo-base
  - Objetos distribuídos são uma evolução
  - Há várias interpretações de P2P: “semi-P2P”, “P2P puro”, ...

Plataformas de Software

Infra-estruturas para
Sistemas Distribuídos

Infra-estruturas de Software

- Sistemas Operacionais Distribuídos
- Sistemas Operacionais de Rede
- Middleware

Introdução

- Hardware é importante para sistemas distribuídos...
- Mas software é o que melhor diferencia sistemas distribuídos

- Infra-estruturas para a construção de sistemas distribuídos são como sistemas operacionais
  - **gerenciam recursos**
    - permitem compartilhamento de recursos (CPUs, memórias, periféricos, rede e dados)
  - Tentam **esconder a heterogeneidade e complexidade do hardware** (remoto, principalmente)
    - **máquina virtual** (software) onde aplicações podem ser executadas mais facilmente

Sistemas operacionais para ambientes distribuídos

- **Fortemente acoplados**
  - Tentam manter visão única e global dos recursos gerenciados
- **Fracamente acoplados**
  - Coleção de computadores, cada um executando seu próprio sistema operacional
  - Trabalham juntos para tornar os serviços e recursos de uns disponíveis aos outros

SOD, SOR e Middleware

- Fortemente acoplados => **sistemas operacionais distribuídos (SODs)**
  - visão única e global dos recursos
- Fracamente acoplados => **sistemas operacionais de rede (SORs)**
  - cada computador executando seu próprio SO
- Para melhor suporte à *transparência de distribuição* são necessários melhoramentos ou serviços adicionais
  - uma camada adicional: **middleware**

SOD, SOR e Middleware (cont.)

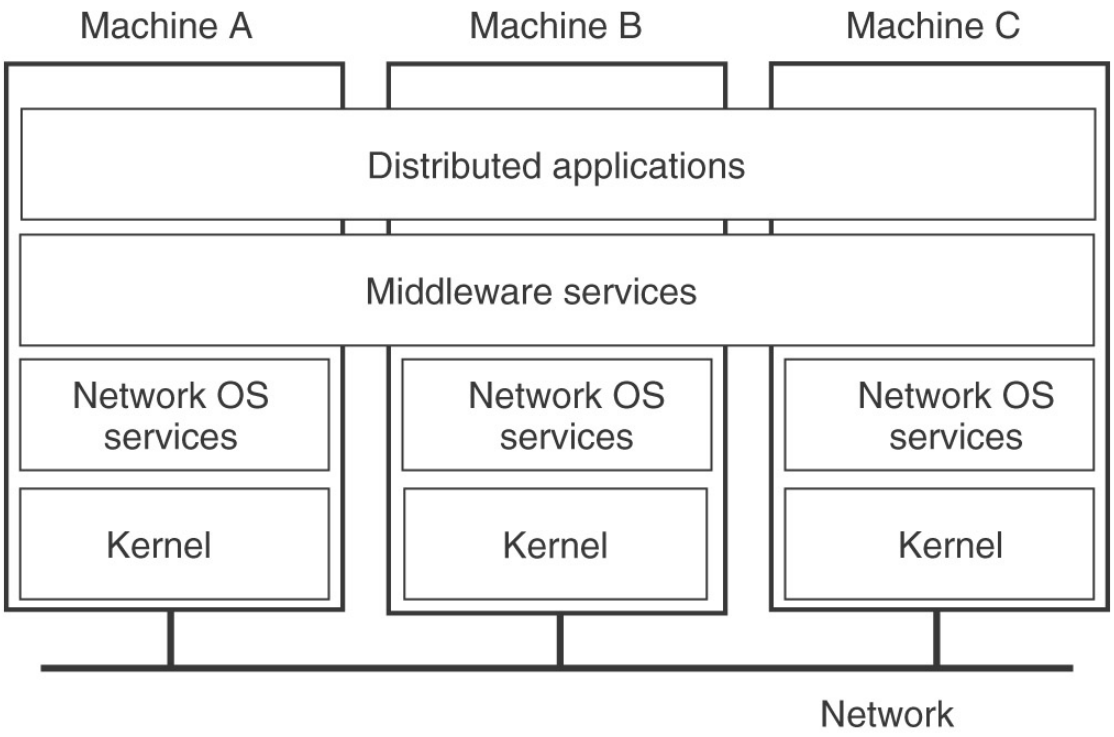
Sistema	**Descrição**	**Principal objetivo**
SOD	SO fortemente acoplado para multi-processadores e multicomputadores homogêneos	Esconder e gerenciar recursos de hardware
SOR	SO fracamente acoplado para multicomputadores heterogêneos (LAN/WAN)	Oferecer serviços locais para clientes remotos
Middleware	Camada adicional sobre um SOR	Prover transparência de distribuição e serviços adicionais

Middleware

Ideal

- Escalabilidade e abertura – SORs
- Transparência e uso mais fácil – SODs

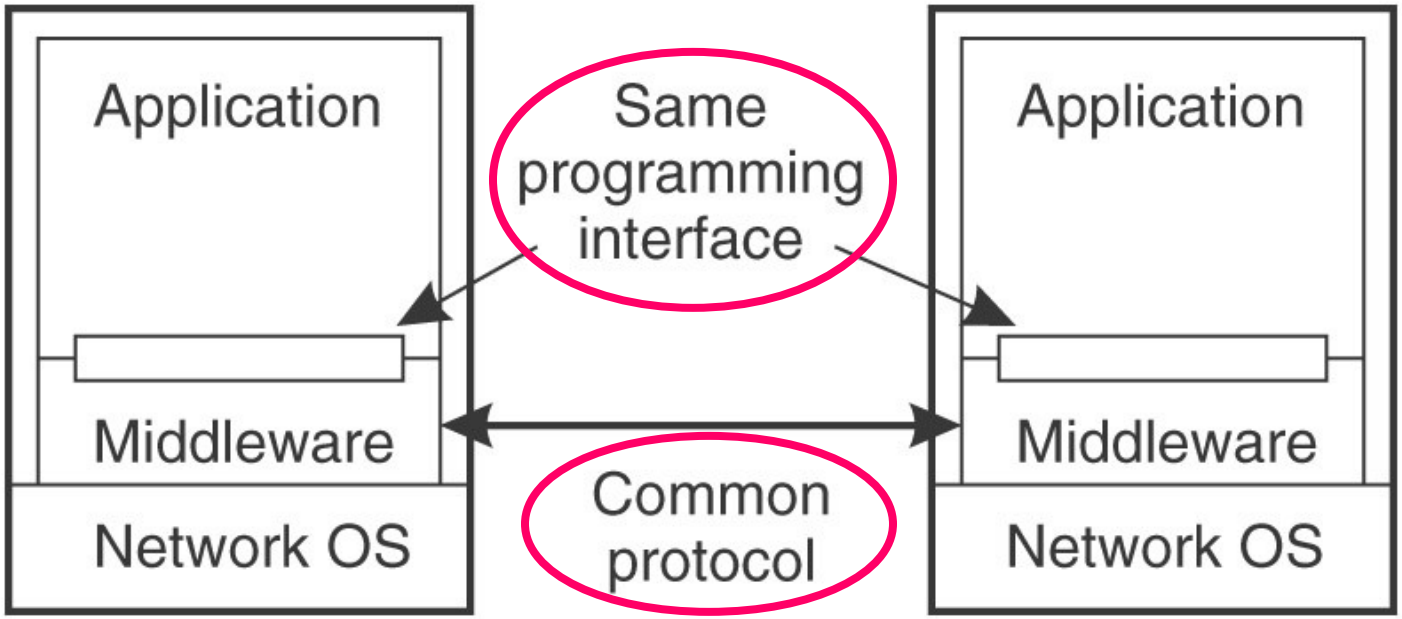
Estrutura


Machine A
Machine B
Machine C
Distributed applications
Middleware services
Network OS services
Network OS services
Network OS services
Kernel
Kernel
Kernel
Network

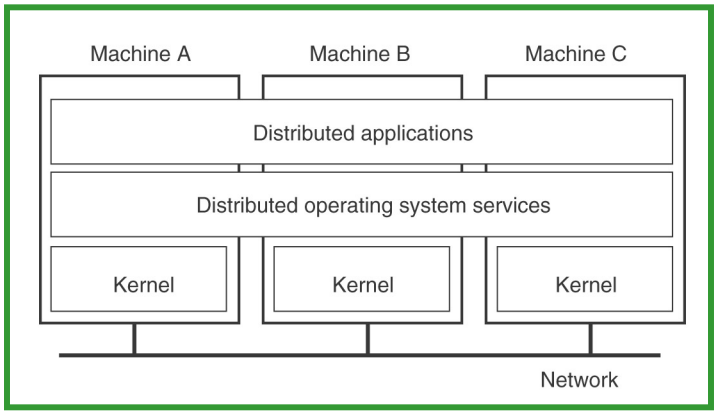
Serviços Agregados

- Nomeação
- Persistência
- Transações
- Segurança
- Tolerância a Falhas
- ....

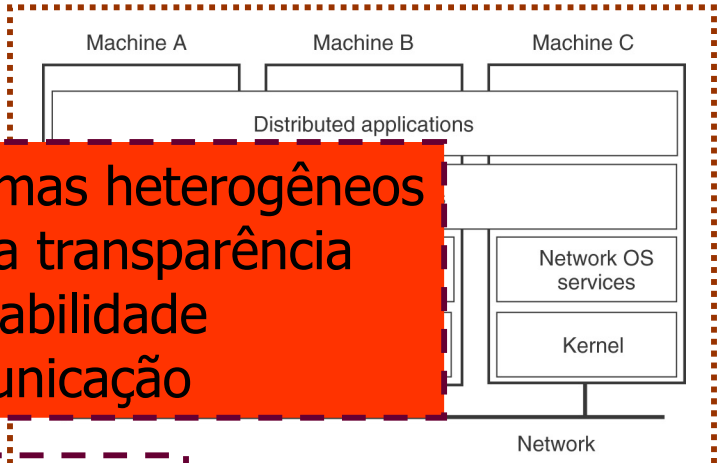
Abertura


Application
Same
programming
interface
Application
Middleware
Middleware
Common
protocol
Network OS
Network OS

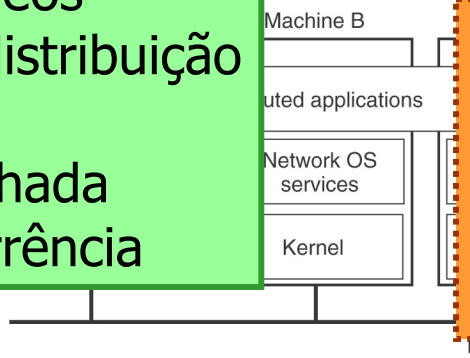
Conclusões



- Sistemas homogêneos
- Transparência de distribuição
- Alto desempenho
- Memória compartilhada
- Controle de concorrência



- Sistemas heterogêneos
- Pouca transparência
- Escalabilidade
- Comunicação



- Sistemas heterogêneos
- Transparência de distribuição e comunicação
- Serviços
- Abertura