Faculdade de Informática - PUCRS

Sistemas Distribuídos

material baseado em slides dos Profs.
Avelino Zorzo, Celso Costa, Fernando Dotti
e Luiz Gustavo Fernandes

e no livro: Distributed Operating Systems - Concepts and Design - Pradeep Sinha

INTRODUÇÃO

Introdução

- Fatores Motivadores
  - Avanços em microeletrônica
    - processadores rápidos, baratos
  - Avanços em comunicações
    - redes eficientes, confiáveis

  - Relação preço/performance
    - melhor usar múltiplos processadores interconectados

Introdução

- O que é um sistema distribuído (SD)?
  - *Um sistema distribuído é uma coleção de computadores independentes que parecem um sistema único para o usuário* [Tanenbaum].
- Dois aspectos:
  - Hardware: autonomia
  - Software: sistema único
- Exemplos:
  - Fábrica com robôs
  - Banco e agências

Introdução

- Vantagens de SD sobre Sistemas Centralizados (SC)

- Economia
  - *Revogação da lei de Grosh's:* performance é proporcional ao custo^2
    - Se o computador B custa o dobro de A, o seu desempenho será 4 vezes maior
  - *Mainframes*
- Velocidade
  - 10.000 CPUs x 50MIPS = 500.000 MIPS
  - Uma CPU (??) para isto deve executar 1 instrução a cada 0.002 nanoseg (2 picoseg). (Velocidade da luz 0.6mm em 2 picoseg)
- Algumas aplicações são naturalmente distribuídas
  - CSCW (*computer supported cooperative work*)
    - Ex: Google Docs

Introdução

- Vantagens de SD sobre SC

- Confiabilidade (*reliability*)
  - 5% for a do ar - 5% em perda de performance
  - Aviação, reatores nucleares, …
- Expansão
  - Aumentar poder de processamento sem se desfazer daquilo que já possui
  - De maneira gradativa

Introdução

- Vantagens de SD sobre Estações isoladas

- Compartilhamento de dados
- Compartilhamento de periféricos
  - Economia
- Comunicação
  - Correio eletrônico
- Flexibilidade
  - Melhor aproveitamento dos recursos

Introdução

- Desvantagens de SD

- Pouco software disponível
  - Nunca se conseguiu atingir a visão de um SO distribuído
    - Ainda hoje SD usa vários Sos locais
- Rede pode causar problemas
- Riscos ligados a segurança
  - Transferência de dados
  - Execução remota

INTRODUÇÃO
Histórico

Introdução (histórico)

- Computadores iniciais: Caros e grandes
- Anos 50 e 60: Spooling, multiprogramação
  - otimizar utilização da CPU
- Início dos anos 60: Sistemas Time Sharing
  - Primeiro passo na direção dos Sistemas Distribuídos
  - Incorpora dois conceitos fundamentais:
    - Compartilhamento de recursos
    - Acesso remoto
  - Terminais como máquina de acesso (terminal burro) – thin node
  - Tarefas executadas no comp. principal

Introdução (histórico)

- Final dos anos 60 e início dos anos 70: Surgimento das redes
  - Ethernet - Xerox Palo Alto: 73 - LAN
  - ARPANet - DoD: 69 – WAN
- Evolução do hardware: redução do tamanho, do preço, aumento da velocidade
- Comunicação: velocidades e distâncias maiores, maior confiabilidade
- final dos anos 70: Protocolo TCP/IP
- Final dos anos 60 e início dos anos 70: Unix
- Início dos anos 80: Estações de trabalho

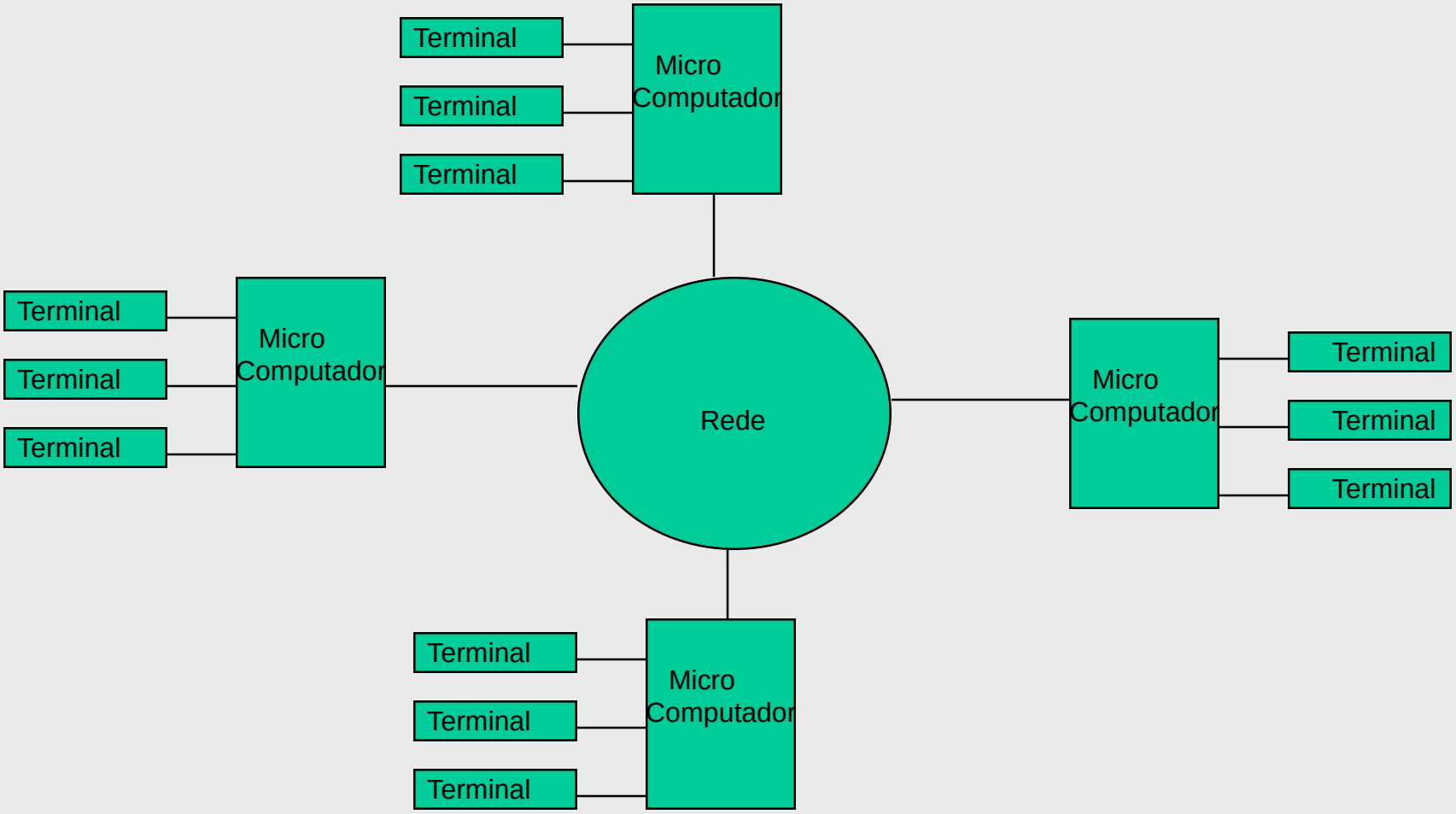
INTRODUÇÃO

Modelos de *Distributed Computing Systems* -
MIMD/NORMA (Pradeep)

Modelos de *Distributed Computing Systems* - MIMD/NORMA (Pradeep)

- Redes de Minicomputadores
- Redes de Estações de Trabalho
- Redes de Estações de Trabalho com Estações Servidoras (Modelo Cliente/Servidor)
- Pool de Processadores
- Cliente/Servidor com um Pool de Processadores

Modelo Rede de Minicomputadores



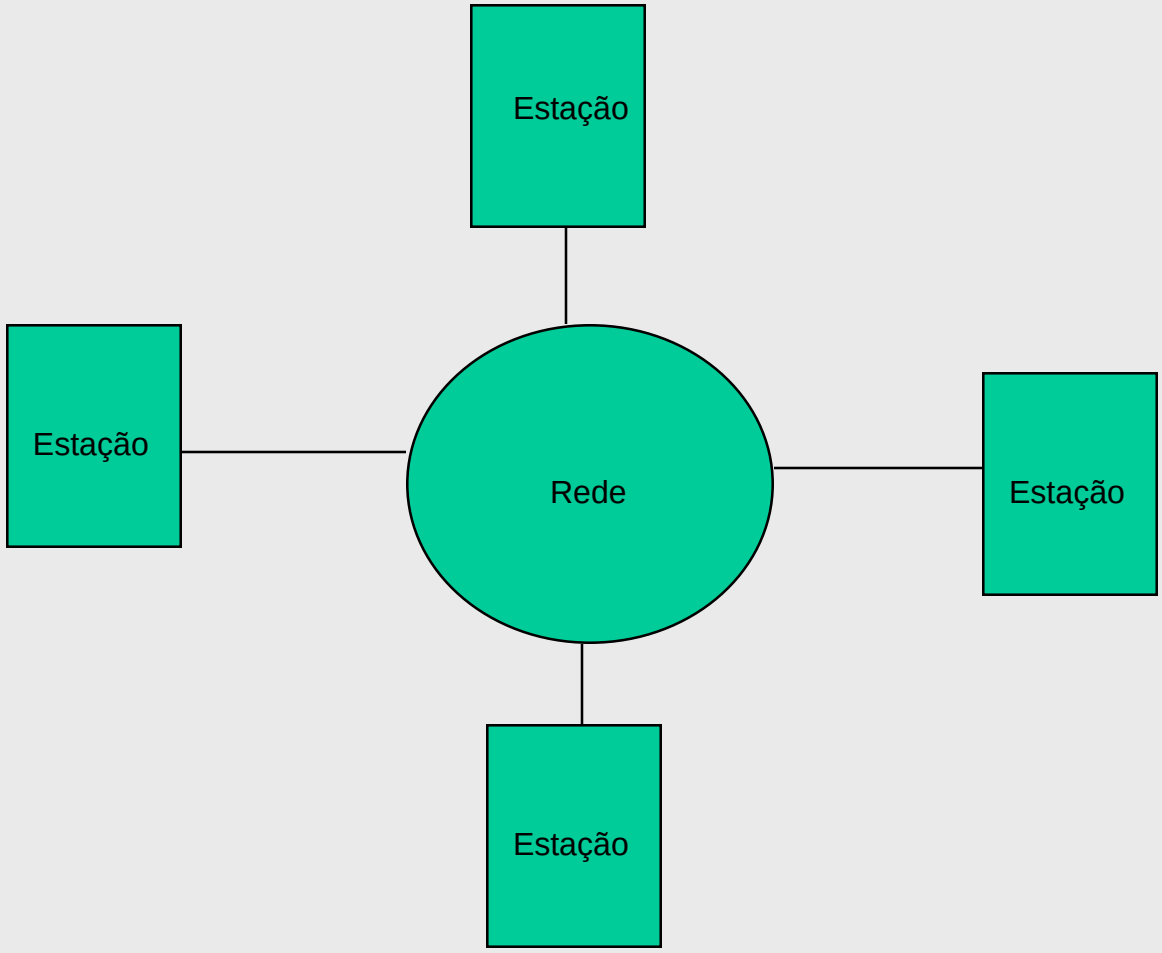
Modelo Rede de Minicomputadores

- Extensão do modelo time-sharing
- cada minicomputador tem usuários conectados via terminais interativos
- rede permite a usuário acessar recursos de outros minicomputadores
  - Cada minicomputador tem SO local
- ex.: ARPANet

Modelo Rede de Minicomputadores

- Objetivo:
  - Compartilhamento de recursos
- Software
  - Telnet
  - Ftp
  - Acesso remoto à bases de dados

Modelo Rede de Estações



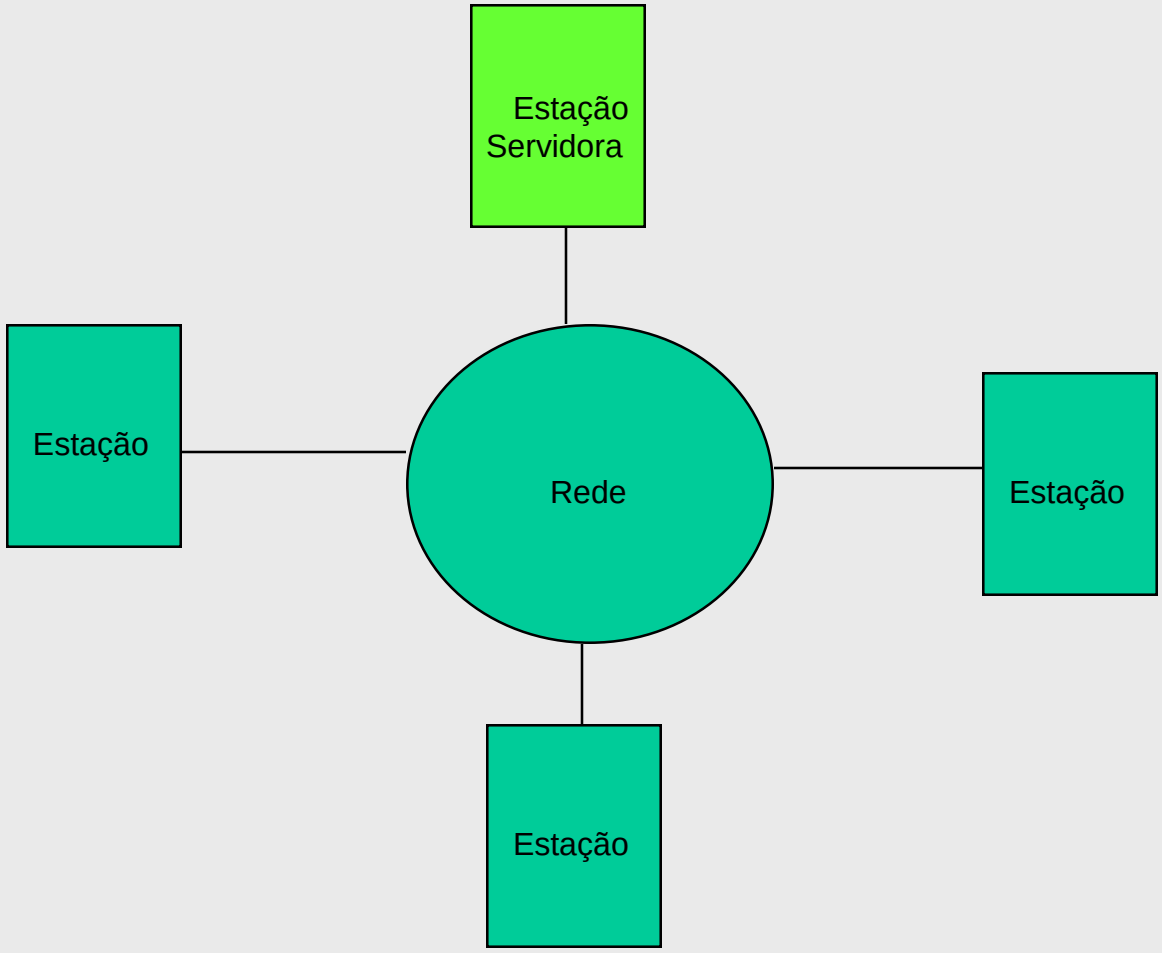
Modelo Rede de Estações

- Objetivo
  - Interconectar estações de maneira a otimizar o seu uso
- Cada estação possui seu próprio sistema operacional, seu próprio disco
- Usuário se conecta a uma estação
- O sistema distribui a carga de processamento na rede de estações (Distribuição de carga)

Modelo Rede de Estações

- Como achar estação livre ?
- Como transferir processo para outra estação ?
- O que fazer com um processo remoto em uma estação livre, se um usuário se *loga* na estação ?
- Ex.: Sprite - Xerox PARC

Modelo Cliente/Servidor

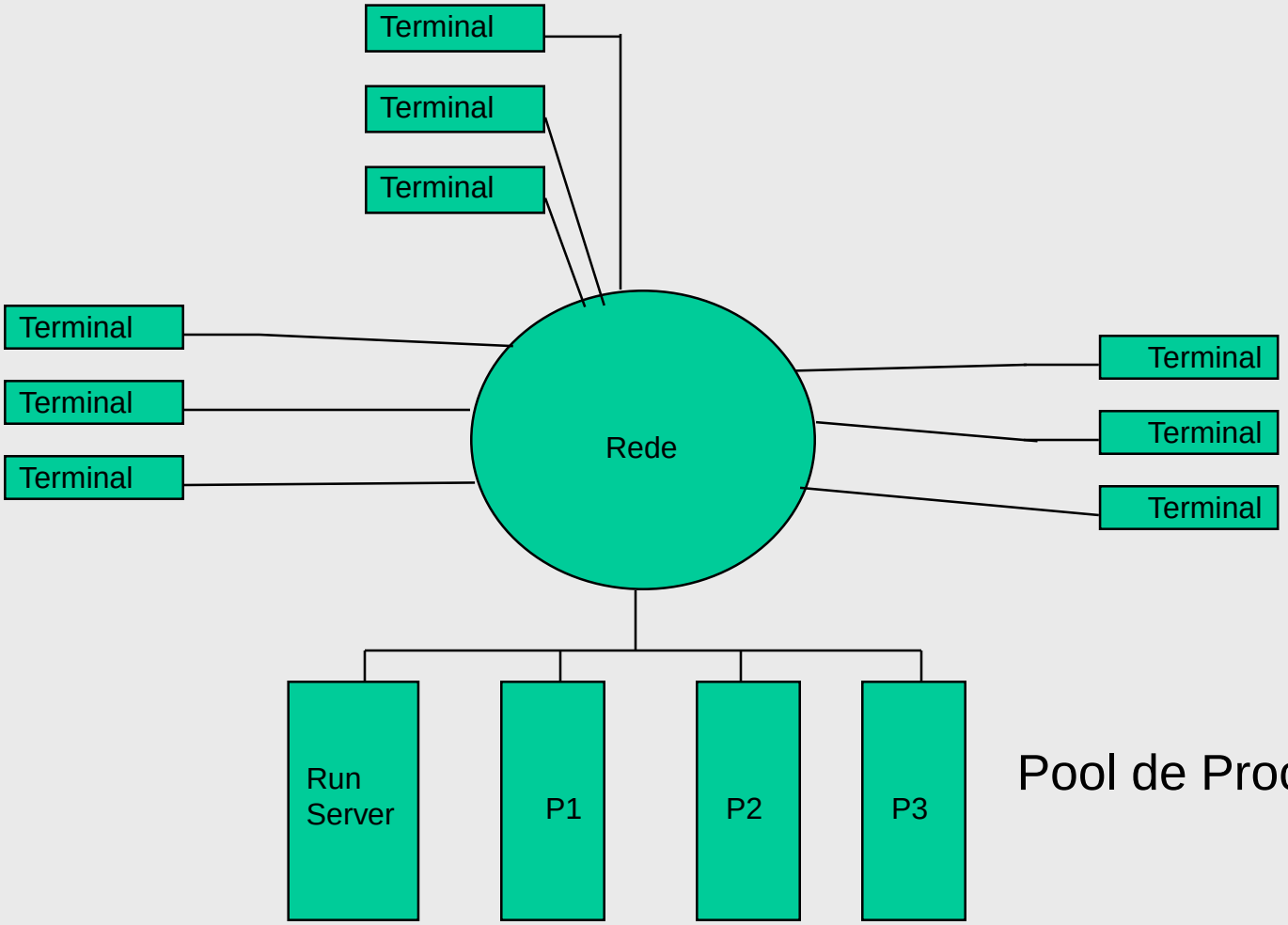


Modelo Cliente/Servidor

- Podem existir estações sem disco
- Estações servidoras oferecem os serviços (servidor de arquivos, servidor de impressão, servidor de Base de Dados)
- Usuário se conecta a uma estação
- O sistema implementa o acesso remoto (transparente) aos serviços EX.: NFS da Sun

- Serviço: funcionalidade provida por uma ou mais máquinas servidoras cooperantes
- Servidor: máquina servidora responsável por um serviço ou parte dele

Modelo Pool de Processadores



Pool de Processadores

Modelo Pool de Processadores

- Os processadores são gerenciados globalmente
  - Alocação de um grupo de processadores a um usuário
  - Liberação do grupo de processadores ao término da execução
- EX.: Sistema Operacional Amoeba
  - Tentativa de sistema operacional distribuído

INTRODUÇÃO

Terminologia

Introdução (terminologia)

- Sistema Operacional
  - programa que controla os recursos de um computador e oferece ao usuário uma interface mais conveniente para o uso do que a máquina

Introdução (terminologia)

- Em um Sistema Computacional Distribuído (DCS) podemos utilizar um:
  - Sistema Operacional de Rede
  - Sistema Operacional Distribuído

Introdução (terminologia)

- Sistema Operacional de Rede (NOS)
  - visão do sistema não é única, usuário conhece as várias máquinas
  - computadores funcionam de maneira autonoma
  - não há cooperação das máquinas para toleância a falhas

  - DCS com um NOS é dito um Networked System ou “Sistema em Rede”

Introdução (terminologia)

- Sistema Operacional Distribuído (DOS)
  - um SOD parece aos seus usuários como um único sistema operacional, centralizado, mas rodando em diversas CPUs independentes. O conceito chave é transparência. O uso de múltiplos processadores deve ser transparente ao usuário.
  - Máquinas não são autônomas
  - tolerância a falhas
  - um DCS com um DOS é dito um Distributed System ou True Distributed System
    - Ex: amoeba

INTRODUÇÃO
Considerações de Projeto de um SOD

Introdução (considerações de projeto)

- Objetivo
  - usuários devem ver um sistema distribuído como um sistema centralizado virtual que é flexível, eficiente, confiável, seguro e fácil de usar (Pradeep)

Introdução (considerações de projeto)

- Principal fonte de complexidade: informação completa sobre o sistema não está disponível
  - recursos separados fisicamente
  - não existe relógio comum
  - mensagens podem ser entregues com atraso (variante)
  - mensagens podem ser perdidas(!)

  - inexistência de informação “up-to-date” consistente
  - maior dificuldade/complexidade para realizar tarefas
    - ex.: como escalonar processadores se não se sabe exatamente quais estão rodando e qual sua carga em determinado momento

Introdução (considerações de projeto)

- Transparência
  - de acesso
    - usuário não distingue entre acesso a recurso local ou remoto
    - interface do usuário - conjunto de chamadas de sistema
      tem que ser projetadas para fazer sentido em sistemas centralizados ou distribuídos - acessar recursos locais ou distantes
    - facilidade de esquema de nomeação global
  - de localização
    - movimentação de recursos e usuários a vontade
    - recursos mantém mesmo nome, nome independente de localização
    - usuários acessam mesmos recursos a partir de qualquer nodo

Introdução (considerações de projeto)

- Transparência
  - de replicação
    - replicação: aumento de desempenho e confiabilidade
    - gerência de réplicas: nomeação de réplicas, mapeamento de nome dado pelo usuário para réplica apropriada do recurso, etc.
  - de falhas
    - mascara dos usuários as falhas parciais do sistema
    - continuidade do funcionamento, talvez de maneira degradada, em presença de falhas
  - de migração
    - razões: manter desempenho, confiabilidade e segurança
    - escolha de objeto (processo) a migrar
    - comunicação deve continuar de maneira transparente

Introdução (considerações de projeto)

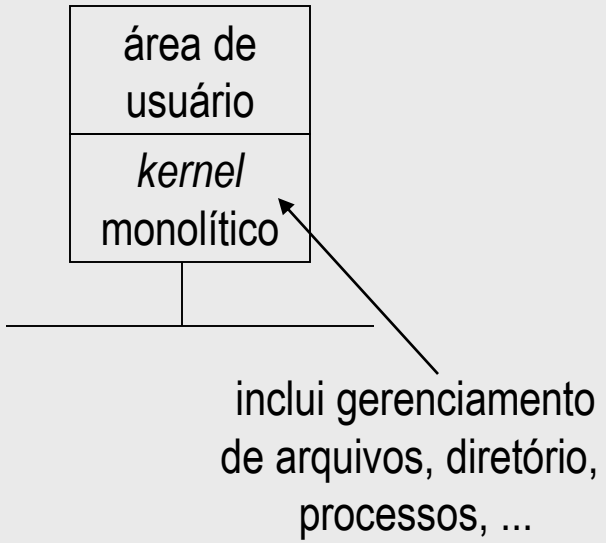
- Transparência
  - de concorrência
    - competição por recursos - necessidade de:
      - ordenação de eventos
      - exclusão mútua
      - no-starvation
      - no dead-lock
  - de desempenho
    - reconfiguração da carga do sistema para aumentar desempenho
    - facilidades de alocação de recursos e migração de processos para homogeneizar a carga dos nodos

Introdução (considerações de projeto)

- Transparência
  - de crescimento/expansão
    - permitir sistema ser estendido sem interromper atividades dos usuários

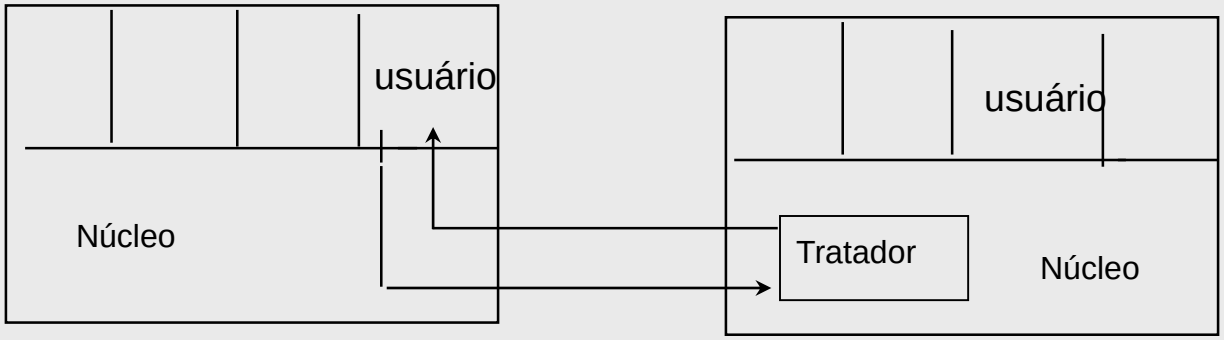
Introdução (considerações)

- Flexibilidade: kernel monolítico vs. Microkernel
  - ***Kernel* monolítico**
  - gerência de processos, memória, dispositivos, arquivos, nomes, e comunicação entre processos provida pelo kernel
  - kernel grande
  - modificação e adição de serviços dificultadas



Introdução (considerações)

- Flexibilidade: kernel monolítico vs. Microkernel
  - ***Kernel* monolítico**

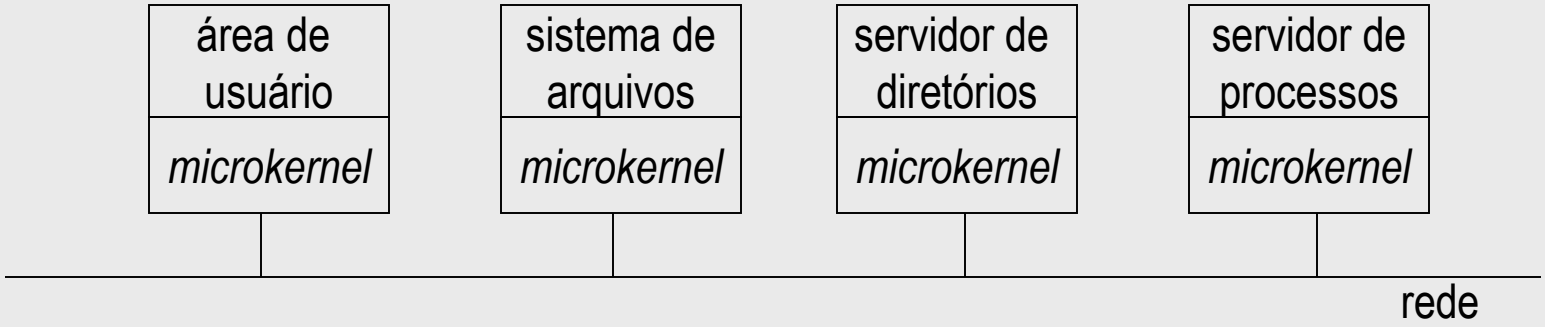


NÚCLEO MONOLÍTICO

Introdução (considerações)

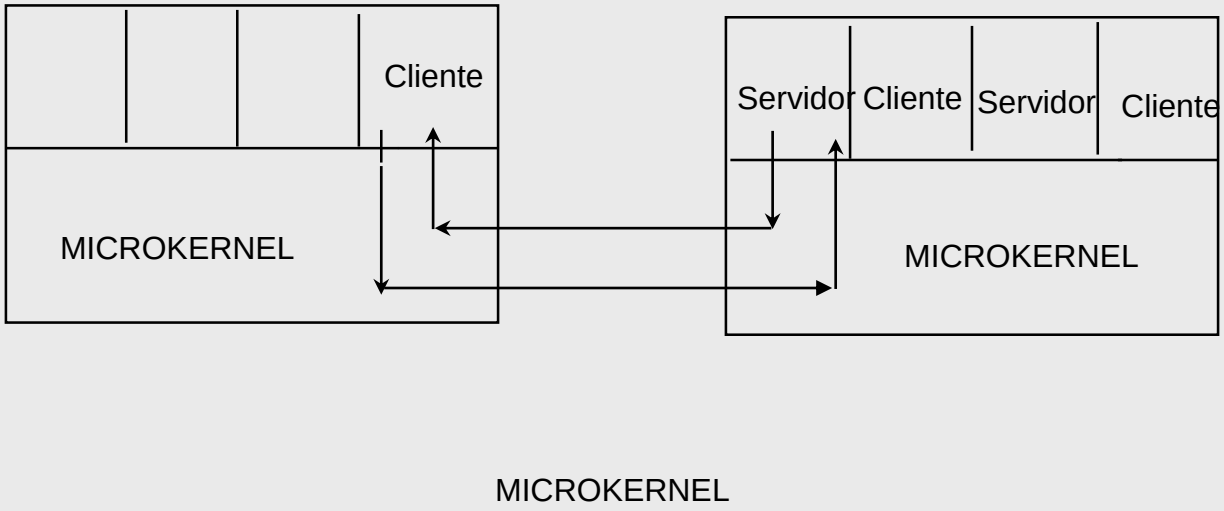
- Flexibilidade: kernel monolítico vs. Microkernel
  - ***Microkernel***
  - kernel pequeno, facilidades mínimas para implementar demais serviços do sistema
  - Funções Básicas em cada nodo:
    - *Comunicação entre processos*
    - Gerência de memória (básica)
    - Gerência de processos (básica)
    - Operações de E/S de baixo nível

- Funções Comuns: Servidores
  gerência de arquivos, nomes, gerência adicional de processos, etc.
- Modo usuário.



Introdução (considerações)

- Flexibilidade: kernel monolítico vs. Microkernel
  - ***Microkernel***



Introdução (considerações)

- Confiabilidade (*reliability*)
  - e.g. 4 servidores 95%
    - probabilidade de os 4 estarem fora do ar = 0,00000625
  - Outras considerações: perda de informações, segurança, tolerância a falhas, …
- Performance
  - Aplicação distribuída não deve ser mais lenta que aplicação centralizada
  - comunicação:
    - agrupar dados e outras informações para mandar através da rede
    - minimizar tráfego (algoritmos distribuídos - quanto se comunicam ?)
  - cache quando possível
  - minimizar cópias de dados
  - tirar vantagem de multiprocessamento

Introdução (considerações)

- "Escalabilidade" (*scalability*)
  - funciona para 200 - funciona para 200.000 ??
  - Princípios a serem evitados:
    - componentes centralizados
    - tabelas centralizadas
    - algoritmos centralizados
  - Algoritmos distribuídos
    - nenhum computador deve conter todas informações sobre o estado do sistema
    - falha em um computador não deve prejudicar outro
    - não assumir que exista relógio global