Sistemas Operacionais

Prof. Dr. Helder Oliveira

Plano de Aula

- Sistemas monolíticos
- Sistemas em camadas
- Sistemas micro-núcleo
- Modelo Cliente-Servidor
- Máquinas virtuais
- Exonúcleo

Estrutura de Sistemas Operacionais

- Vimos como os sistemas operacionais parecem por fora (isto é, a interface do programador), é hora de darmos uma olhada por dentro.
- Estudaremos seis estruturas diferentes que foram tentadas, a fim de termos alguma ideia do espectro de possibilidades.

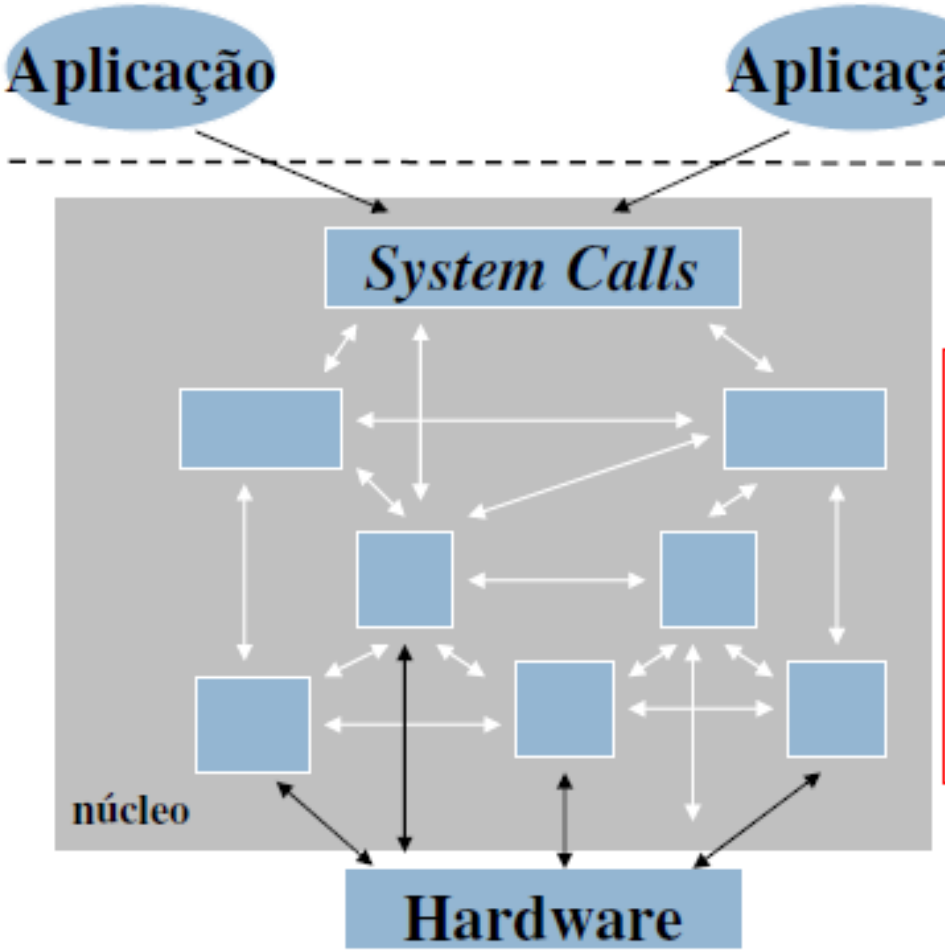
Sistemas monolíticos

- Organização mais comum.
- Nesta abordagem o SO inteiro é executado como um único programa no modo núcleo.
- O sistema operacional é escrito como uma coleção de rotinas, ligadas a um único grande programa binário executável.
- Cada procedimento no sistema é livre para chamar qualquer outro, se este oferecer alguma computação útil de que o primeiro precisa.
  - Ser capaz de chamar qualquer procedimento que você quer é muito eficiente, mas ter milhares de procedimentos que podem chamar um ao outro sem restrições pode também levar a um sistema difícil de lidar e compreender.
  - Uma quebra em qualquer uma dessas rotinas derrubará todo o sistema operacional.

Sistemas monolíticos

- Para construir o programa objeto real do sistema operacional quando essa abordagem é usada, é preciso primeiro compilar todas as rotinas individuais (ou os arquivos contendo as rotinas) e então juntá-las em um único arquivo executável usando o ligador (linker) do sistema.
- Em termos de ocultação de informações, essencialmente não há nenhuma — toda rotina é visível para toda outra rotina.
- É possível se ter alguma estrutura.
  - As chamadas de sistema providas pelo sistema operacional são requisitadas colocando-se os parâmetros em um local bem definido (por exemplo, em uma pilha) e então executando uma instrução de desvio de controle (trap).

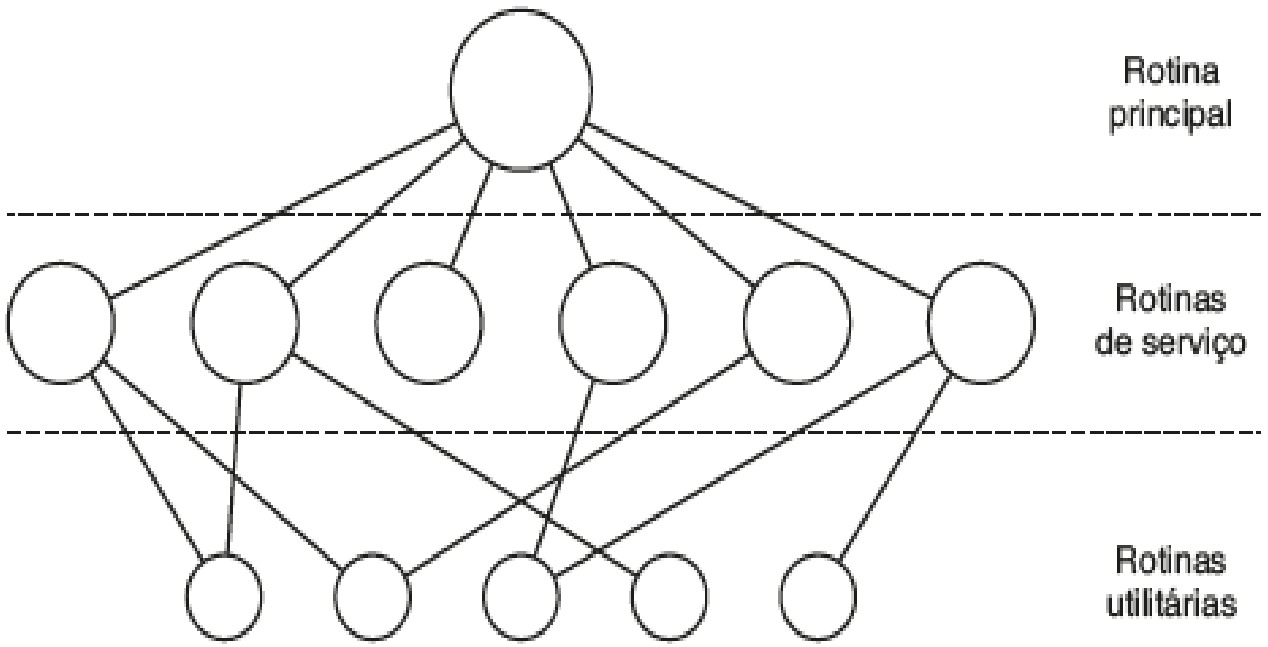
Sistemas monolíticos



1. Um programa principal que invoca a rotina do serviço requerido.
2. Um conjunto de rotinas de serviço que executam as chamadas de sistema.
3. Um conjunto de rotinas de utilidade que auxiliam as rotinas de serviço.

Sistemas monolíticos

FIGURA 1.24 Um modelo de estruturação simples para um sistema monolítico.



- Nesse modelo, para cada chamada de sistema há uma rotina de serviço que se encarrega dela e a executa.
- As rotinas utilitárias fazem coisas que são necessárias para várias rotinas de serviços, como buscar dados de programas dos usuários. Essa divisão em três camadas é mostrada na figura.

Sistemas monolíticos

- Pode ser comparada com uma aplicação formada por vários procedimentos que são compilados separadamente e depois linkados, formando um grande e único programa executável.
  - Grande desempenho
  - Uma falha pode paralisar todo o núcleo. O sistema pode parar por causa de um erro.
  - As interfaces e níveis de funcionalidade não são bem separados nem estão unificados. O excesso de liberdade torna o sistema vulnerável
  - Ex:
    - Linux e FreeBSD

Sistemas monolíticos

- Além do sistema operacional principal que é carregado quando o computador é inicializado, muitos sistemas operacionais dão suporte a extensões carregáveis, como drivers de dispositivos de E/S e sistemas de arquivos.
- Esses componentes são carregados conforme a demanda.
  - No UNIX eles são chamados de bibliotecas compartilhadas.
  - No Windows são chamados de DLLs (Dynamic Link Libraries — bibliotecas de ligação dinâmica).
    - Eles têm a extensão de arquivo .dll e o diretório C:\Windows\system32 nos sistemas Windows tem mais de 1.000 deles.

Sistema em Camadas

- Divide o sistema operacional em sistemas sobrepostos. Cada módulo oferece um conjunto de funções que pode ser usado por outros módulos.
- A vantagem da estruturação em camadas é isolar o sistema operacional, facilitando sua alteração e depuração, além de criar uma hierarquia de níveis de modos, protegendo as camadas mais internas.

Sistema em Camadas

- O empilhamento de várias camadas de software faz com que cada pedido de uma aplicação demore mais tempo para chegar até o dispositivo periférico ou recurso a ser acessado, prejudicando o desempenho do sistema.
- Não é óbvio dividir as funcionalidades de um núcleo de sistema operacional em camadas horizontais de abstração crescente, pois essas funcionalidades são inter-dependentes, embora tratem muitas vezes de recursos distintos.

Sistema em Camadas

- Primeiro S.O. Em camadas **THE**
  - Desenvolvido na Technische Hogeschool Eindhoven na Holanda por E. W. Dijkstra (1968) e seus estudantes.
  - Sistema em lote simples para um computador holandês, o Electrologica X8, que tinha 32 K de palavras de 27 bits (bits eram caros na época).
  - Sistema tinha seis camadas.

FIGURA 1.25 Estrutura do sistema operacional THE.

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

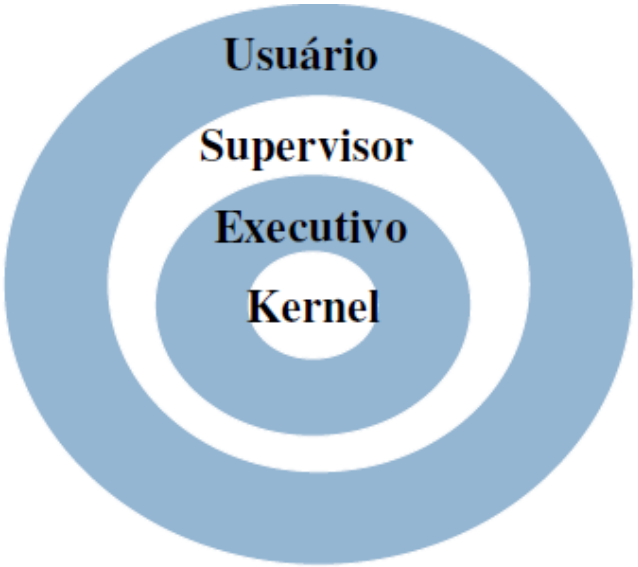
Sistema em Camadas

- O sistema THE era um sistema de lote simples para um computador holândes, o Electrologica X8.
  - Camada 0 – lidava com alocação do processador , alternando entre processos quando ocorriam interrupções ou quando os temporizadores expiravam. Em outras palavras, a camada 0 (zero) proporcionava a multiprogramação básica da CPU.
  - Camada 1 – fazia o gerenciamento da memória. Ela alocava espaço para os processos da memória principal e em um tambor (Antigo meio magnético de armazenamento de dados) utilizado para armazenar partes do processo (páginas) para os quais não havia lugar na memória principal.
  - Camada 2 – fazia a comunicação entre o console do operador e cada processo.
  - Camada 3 – gerenciava dispositivos de entrada e saída.
  - Na camada 4 – localizavam-se os programas de usuários. Eles não tinham de se preocupar com o gerenciamento de processo, memoria, console ou E/S.
  - Na camada 5 – estava localizado o processo operador do sistema.

Sistema em Camadas (anéis)

- Outra generalização do conceito de camadas estava presente no sistema MULTICS.
- Em vez de camadas, MULTICS foi descrito como tendo uma série de anéis concêntricos
  - Anéis mais internos são mais privilegiados que os externos;
  - Procedimentos de anéis externos executavam chamadas de sistema para utilizar os serviços dos anéis internos;
  - Proteção dos segmentos de memória.

Sistema em Camadas



Sistema Multics VMS

+No sistema MULTICS VMS as camadas inferiores são as mais privilegiadas.

- Quando um procedimento em um anel exterior queria chamar um procedimento em um anel interior, ele tinha de fazer o equivalente de uma chamada de sistema, isto é, uma instrução de desvio, TRAP, cujos parâmetros eram cuidadosamente conferidos por sua validade antes de a chamada ter permissão para prosseguir.

Sistema em Camadas

- O esquema de camadas THE era na realidade somente um suporte para o projeto, pois em última análise todas as partes do sistema estavam unidas em um único programa executável,
- No MULTICS o mecanismo de anéis estava bastante presente no momento de execução e imposto pelo hardware.
- A vantagem do mecanismo de anéis é que ele pode ser facilmente estendido para estruturar subsistemas de usuário.

Sistemas micro-núcleo (microkernel)

- Uma tendência dos sistemas operacionais é tornar o núcleo menor e mais simples possível e para implementar esta ideia o sistema é dividido em processos.
  - Erros no código do núcleo podem derrubar o sistema instantaneamente.
  - Processos de usuário podem ser configurados para ter menos poder, de maneira que um erro possa não ser fatal.
  - A densidade de erros depende do tamanho do módulo, idade do módulo etc., mas um número aproximado para sistemas industriais sérios fica entre dois e dez erros por mil linhas de código.
    - Sistema operacional monolítico de cinco milhões de linhas de código contenha entre 10.000 e 50.000 erros no núcleo.
      - Nem todos são fatais.

Sistemas micro-núcleo (microkernel)

- Ideia básica:
  - Atingir uma alta confiabilidade através da divisão do sistema operacional em módulos pequenos e bem definidos.
    - **O micronúcleo é executado em modo núcleo.**
    - **O resto é executado como processos de usuário comuns relativamente sem poder.**
- Ao se executar cada driver de dispositivo e sistema de arquivos como um processo de usuário em separado, um erro em um deles pode derrubar esse componente, mas não consegue derrubar o sistema inteiro.
  - Um erro no driver de áudio fará que o som fique truncado ou pare, mas não derrubará o computador.

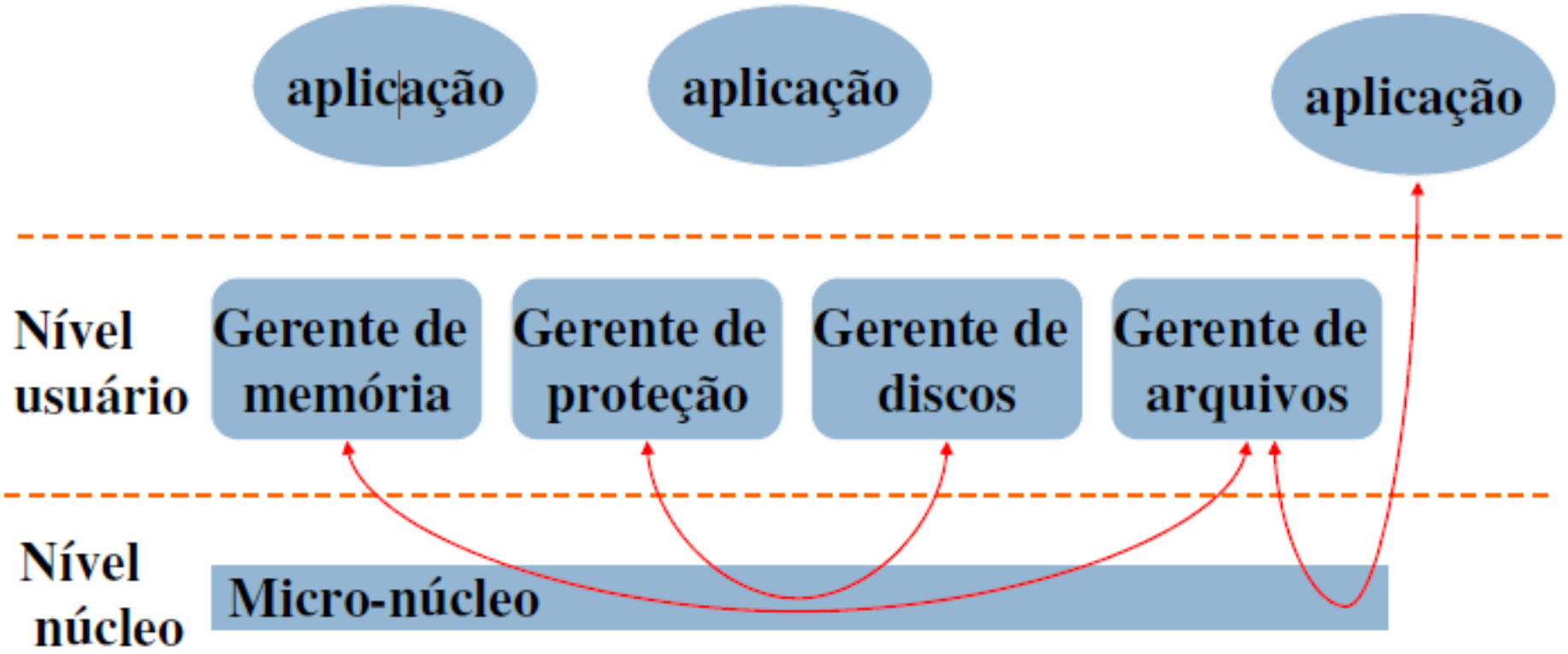
Sistemas micro-núcleo (microkernel)

- A utilização deste modelo permite que os servidores executem em modo usuário.
- Apenas o núcleo do sistema, responsável pela comunicação entre clientes e servidores, executa no modo kernel.
- O sistema operacional passa a ser de mais fácil manutenção.
- Não importa se o serviço esta sendo processado em um único processador, com múltiplos processadores (fortemente acoplado) ou em sistema distribuído (fracamente acoplado).

Sistemas micro-núcleo (microkernel)

- Um ambiente distribuído permite que um cliente solicite um serviço e a resposta seja processada remotamente.
- Sua implementação é difícil e mais usualmente é implantado uma combinação do modelo de camadas com o cliente-servidor.
- O núcleo do sistema passa a incorporar o escalonamento e gerência de memória além das funções de device drivers.

Micro-núcleo – Visão Geral



Micro-núcleo

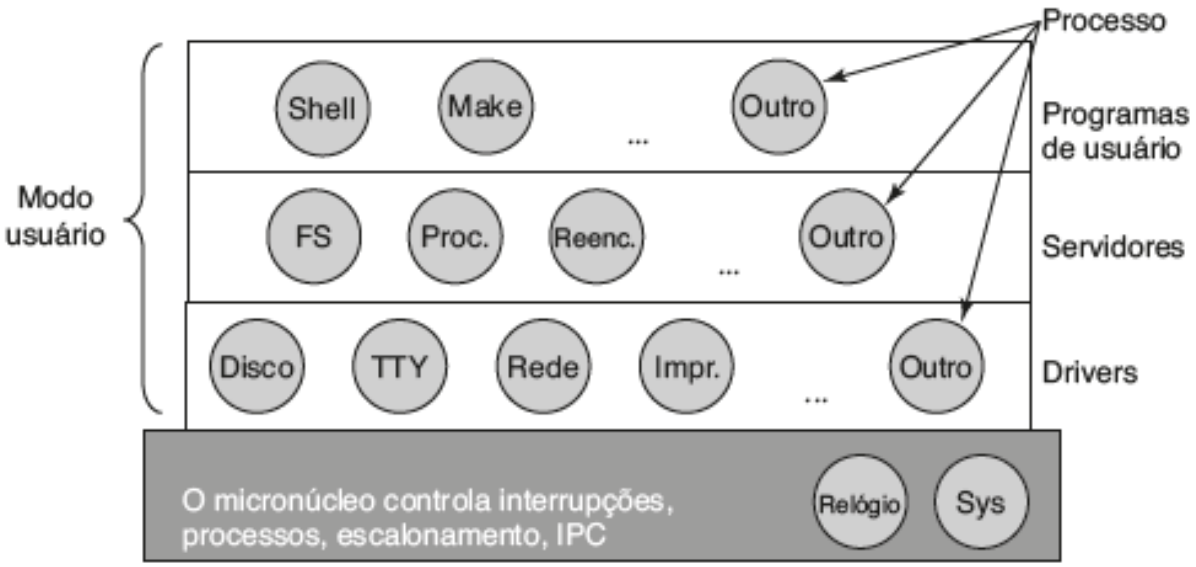
- A idéia básica por trás do projeto do micronúcleo é alcançar alta confiabilidade por meio da divisão do sistema operacional em módulos pequenos, bem definidos, e apenas um desses módulos – o micronúcleo – é executado no modo núcleo e o restante é executado como processos de usuário.
- Quando há execução de cada driver de dispositivo e cada sistema de arquivo como processo separado, um erro em um deles pode quebrar aquele componente, mas não pode quebrar o sistema inteiro.

Micro-núcleo

- Sistemas operacionais de computadores de mesa comuns não usam micronúcleos.
- Micronúcleos são comuns em aplicações de tempo real, industriais, avionica e militares, que são cruciais e tem requisitos de confiabilidade muito altos.
- Ex:
  - Integrity.
  - K42 PikeOS.
  - QNX.
  - Symbiam.
  - MINIX3.

Micro-núcleo

FIGURA 1.26 Estrutura simplificada do sistema MINIX.



- MINIX 3
  - 12.000 linhas de C e cerca de 1.400 linhas de assembler.
  - Núcleo:
    - Tratadores de chamada de núcleo rotulados Sys.
    - Driver de dispositivo para o relógio.
  - Processos de usuário
    - Outros drivers de dispositivos operam como processos de usuário em separado.
  - Fora do núcleo, o sistema é estruturado como três camadas de processos, todos sendo executados em modo usuário.
    - A camada mais baixa contém os drivers de dispositivos.
    - Os servidores, que fazem a maior parte do trabalho do sistema operacional.

Modelo Cliente Servidor

- Uma variação da ideia do micronúcleo é distinguir entre duas classes de processos.
  - **Servidores**, que prestam serviço.
  - **Clientes**, que usam serviços.
- Frequentemente a camada inferior é o micronúcleo mas não obrigatoriamente. A essência é a presença de processos clientes e servidores.

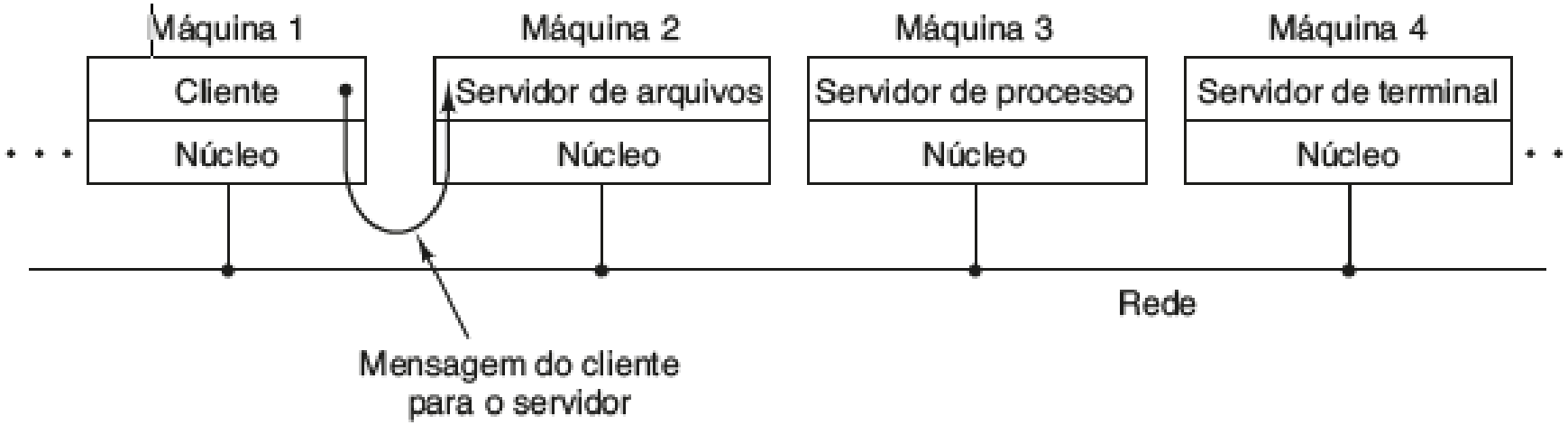
Modelo Cliente Servidor

- A comunicação entre clientes e servidores é realizada muitas vezes pela troca de mensagens.
  - Para obter um serviço, um processo cliente constrói uma mensagem dizendo o que ele quer e a envia ao serviço apropriado.
  - O serviço então realiza o trabalho e envia de volta a resposta.
  - Se acontecer de o cliente e o servidor serem executados na mesma máquina, determinadas otimizações são possíveis, mas conceitualmente, ainda estamos falando da troca de mensagens aqui.

Modelo Cliente Servidor

- Uma generalização óbvia dessa ideia é ter os clientes e servidores sendo executados em computadores diferentes, conectados por uma rede local ou de grande área, como descrito na Figura 1.27.
- Modelo cliente-servidor é uma abstração que pode ser usada para uma única máquina ou para uma rede de máquinas.

FIGURA 1.27 O modelo cliente-servidor em uma rede.



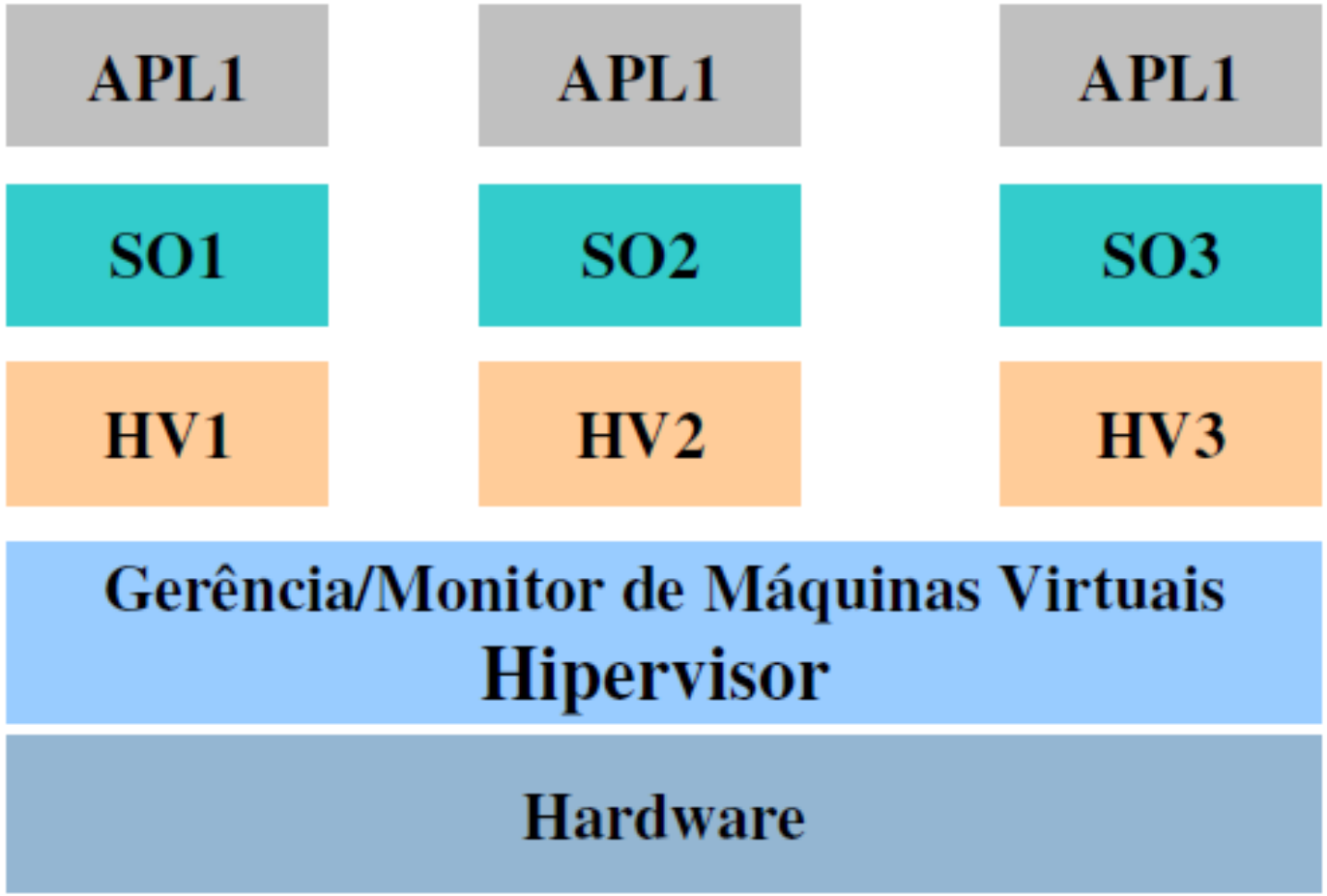
Modelo Cliente Servidor

- Grande parte da web opera dessa maneira.
- Um PC pede uma página na web para um servidor e ele a entrega. Esse é o uso típico do modelo cliente-servidor em uma rede.

Máquina Virtual

- Máquinas virtuais não são máquinas estendidas com arquivos e outras características convenientes.
- São cópias exatas do hardware, inclusive com modos núcleo/usuário, E/S, interrupções e tudo o que uma máquina real tem.
- Cada VM pode executar qualquer SO capaz de ser executado diretamente sobre o hardware.
- Diferentes VMs podem executar diferentes SOs.

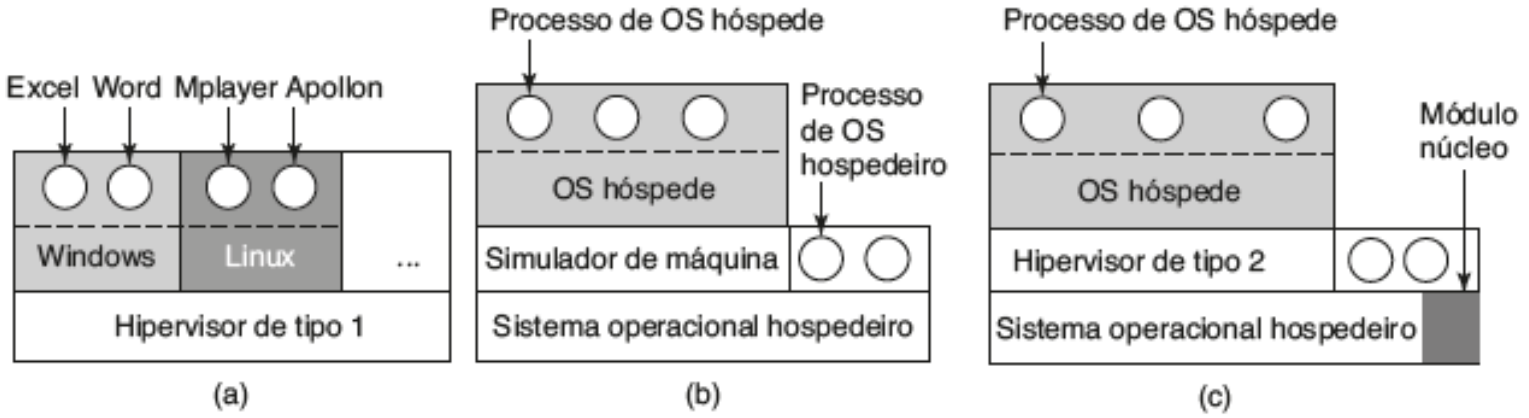
Máquina Virtual



Máquina Virtual

- Hipervisor do tipo 1(a) são executados diretamente nohardware.
  - Eucalyptus
- Hipervisor do tipo 2(b) são executados como aplicativos na camada superior do SO
  - Ex: VMware, VirtualBox

FIGURA 1.29 (a) Um hipervisor de tipo 1. (b) Um hipervisor de tipo 2 puro. (c) Um hipervisor de tipo 2 na prática.



Exonúcleo

- Em vez de clonar a máquina real, como nas VMs, outra estratégia é dividi-la ou dar a cada usuário um subconjunto de recursos.
- Assim uma VM pode ter os blocos de 0 a 1023 do disco e outra de 1024 a 2047 e assim por diante.
- Na camada inferior há o programa exonúcleo responsável por alocar recursos às VMs e verificar as tentativas de uso para assegurar que uma máquina não esteja utilizando os recursos de outra.
- Cada VM pode utilizar seu próprio SO mas somente com os recursos que pediu e que foram alocados.
- Este tem vantagem de separar, com menor custo, a multiprogramação (no exonúcleo) do código do SO do usuário, visto que o exonúcleo mantém as VMs umas fora do alcance das outras.

Leitura

- SISTEMAS OPERACIONAIS MODERNO 4ª edição
  - 1.7 Estrutura de sistemas operacionais
  - 1.8 O mundo de acordo com a linguagem C
  - 1.9 Pesquisa em sistemas operacionais

Dúvidas?