Sistemas Distribuídos

Professor Marco Câmara

2023-2

V.1.11 - 2021.10

Programa

- Fundamentos;
- Comunicação e Sincronismo;
- Grupos, Processos e *Threads*;
- Segurança
- Serviços
- Tolerância a Falhas

Sistema Distribuído?

Conjunto de dispositivos computacionais independentes que se apresenta a seus usuários como um sistema único e coerente. [Tanenbaum]

Sistema Distribuído?

Sistema no qual os componentes de hardware ou software, localizados em computadores interligados em rede, comunicam-se e coordenam suas ações apenas enviando mensagens entre si. [Coulouris]

Sistema Distribuído?

Sistema onde diversos processadores autônomos e repositórios de dados suportam a interação entre processos e bancos de dados colaborando com um objetivo comum. Esses processos coordenam suas atividades e troca de informações através de mensagens enviadas sobre uma rede de comunicação. [Sloman and Kramer]

Sistema Distribuído?

Objetivos

- Conectar usuários;
- Compartilhar recursos;
- Dividir tarefas computacionais.

Vantagens

Vantagem	Detalhamento
Economia	Microprocessadores oferecem melhor relação custo-benefício do que computadores de grande porte.
Performance	Um sistema distribuído por diversos dispositivos pode, na maior parte das situações, oferecer performance superior a um sistema centralizado.
Natureza Distribuída	Algumas aplicações, por sua natureza, exigem dispositivos separados até para fazerem sentido.
Escalabilidade	O poder computacional e os recursos podem ser acrescentados aos poucos.
Confiabilidade (X microcomputadores)	Graças aos recursos de redundância e ao balanceamento de carga, um sistema distribuído tipicamente é muito mais seguro do que um sistema centralizado de pequeno porte. O memso é válido para computadores de maior porte, em algumas situações.

Desvantagens

Desvantagem	Detalhamento
Experiência e Ferramentas	São poucos os exemplos de projetos e implementações de Sistemas Distribuídos; qual o Sistema Operacional, Linguagem e Ambiente de Desenvolvimento apropriado para Sistemas
Complexidade	Maiores exigências e requisitos, inclusive quanto aos recursos de hardware e rede; mecanismos de tolerância a falhas; manutenção de consistência e integridade.
Segurança	A distribuição das informações dificulta a proteção efetiva dos dados, e exige política cuidadosa de segurança.
Rede de Comunicação	Performance, estabilidade e segurança são fundamentais.

Conceitos de *hardware*

- Um sistema distribuído pressupõe diversos processadores, que podem estar organizados de diferentes formas:
  - Como opera a comunicação entre eles?
  - Como é realizada a interconexão entre eles?
- Classifica-se a organização desses processadores tipicamente usando a **Taxonomia de Flynn**, que leva em consideração:
  - Fluxo de instruções
  - Fluxo de dados

Taxonomia de Flynn

Instruções \ Fluxo de Dados	Único	Múltiplo
Único	SISD Single Instruction, Single Data	SIMD Single Instruction, Multiple Data
Múltiplo	MISD Multiple Instruction, Single Data	MIMD Multiple Instruction, Multiple Data

V.1.11 - 2021.10

Taxonomia de Flynn

- *Single Instruction Single Data*:
  - **Computadores de Von Neumann**: microprocessadores tradicionais e Computadores de Grande Porte;
- *Single Instruction Multiple Data*
- *Multiple Instruction Single Data*
- *Multiple Instruction Multiple Data*

Taxonomia de Flynn

- *Single Instruction Single Data*
- *Single Instruction Multiple Data*:
  - Uma Unidade de Controle, múltiplas ULA (Unidade Lógica e Aritmética);
  - Processadores Vetoriais, Super-computadores;
  - Utilizadas em Computadores convencionais para processamento multimídia (Intel Core i7 possui a extensão SSE - *Streaming SIMD Extensions*);
- *Multiple Instruction Single Data*
- *Multiple Instruction Multiple Data*

Taxonomia de Flynn

- *Single Instruction Single Data*
- *Single Instruction Multiple Data*
- *Multiple Instruction Single Data:*
  - Não existem computadores que operem nesse modelo;
  - *Pipelined Machines* seriam um exemplo? (pesquisar)
- *Multiple Instruction Multiple Data*

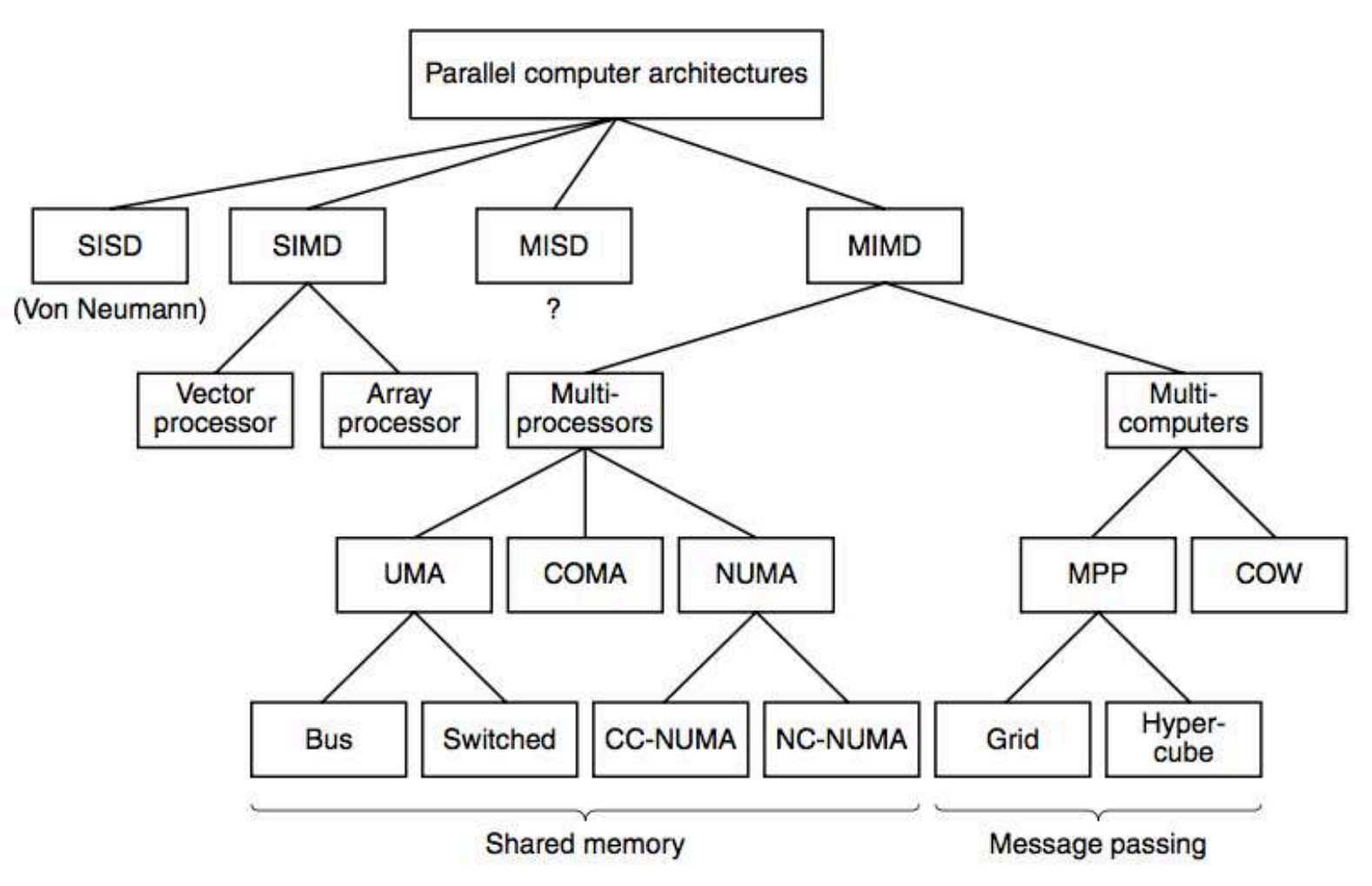
Taxonomia de Flynn

- *Single Instruction Single Data*
- *Single Instruction Multiple Data*
- *Multiple Instruction Single Data*
- *Multiple Instruction Multiple Data:*
  - Múltiplos processadores e Múltiplos Computadores;
  - Categoria mais comum no estudo dos Sistemas Distribuídos.

Aula 02

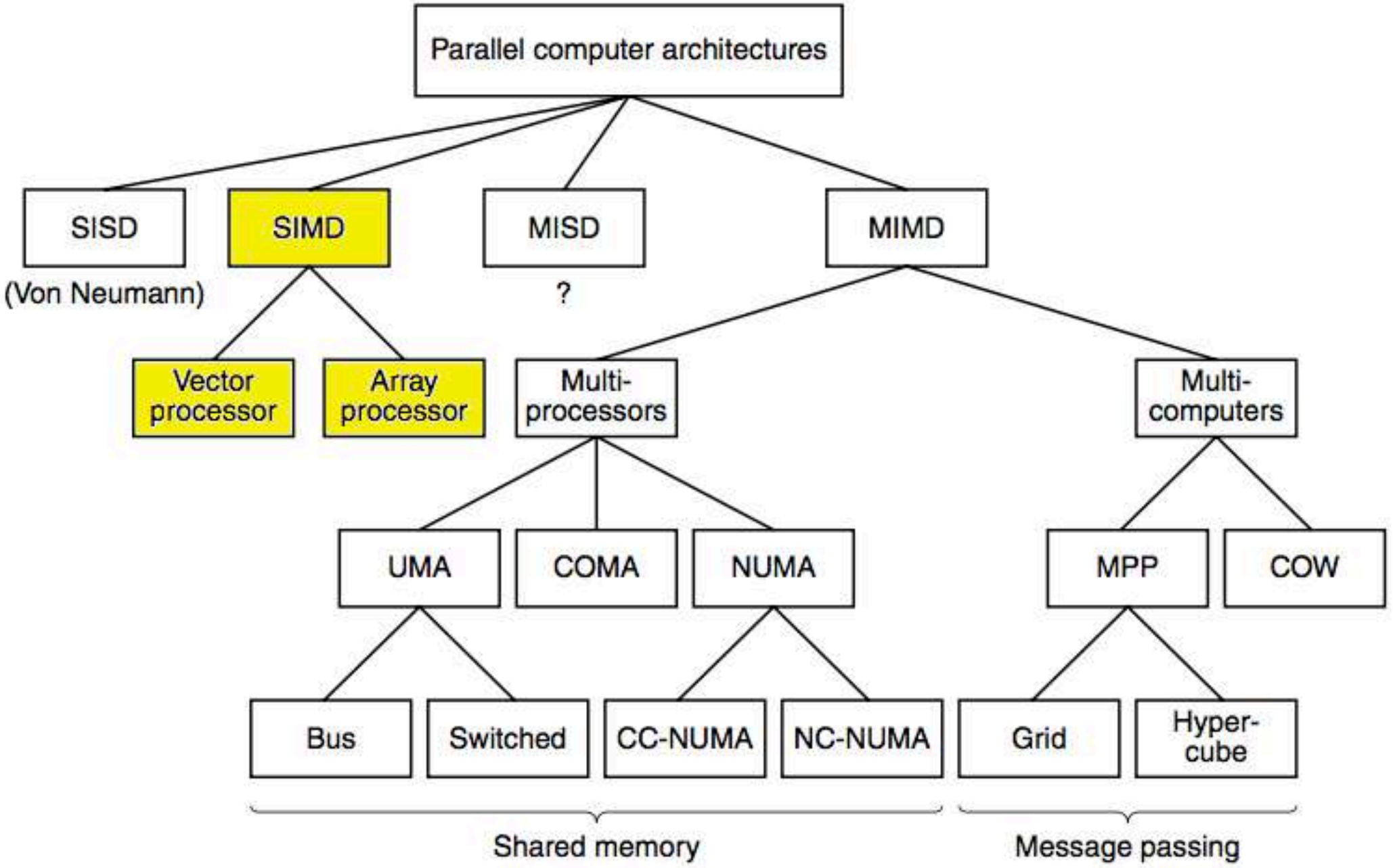
V.1.11 - 2021.10

Taxonomia de Flynn ampliada



SIMD se divide em duas categorias:

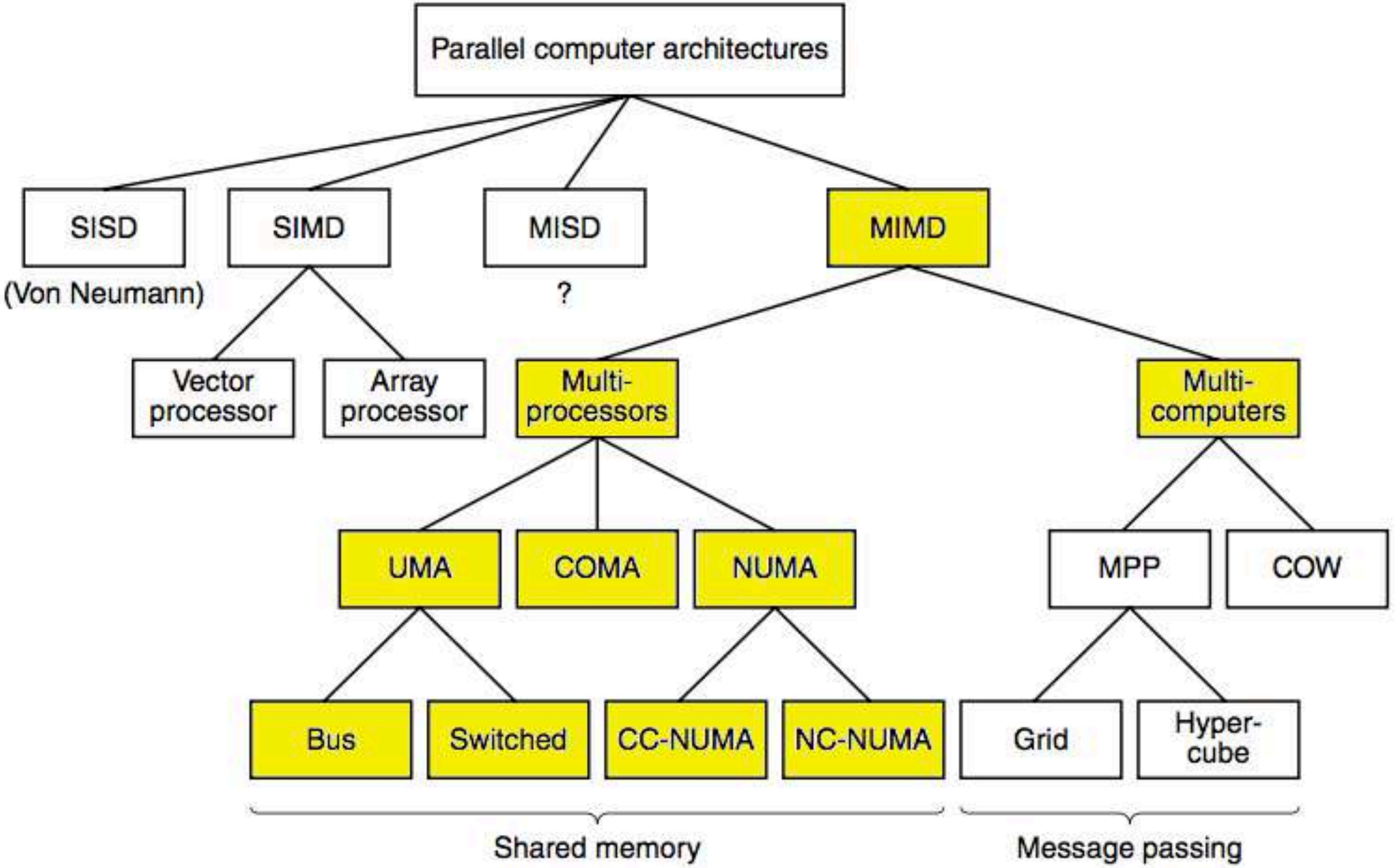
- **Processadores Vetoriais**: supercomputadores numéricos e máquinas que aplicam a mesma instrução em diferentes elementos de um vetor;
- **Vetor de Processadores**: unidade controladora envia a mesma instrução para diversas ULAs independentes.



MIMD baseado em máquinas multi-processadas se divide em três categorias:

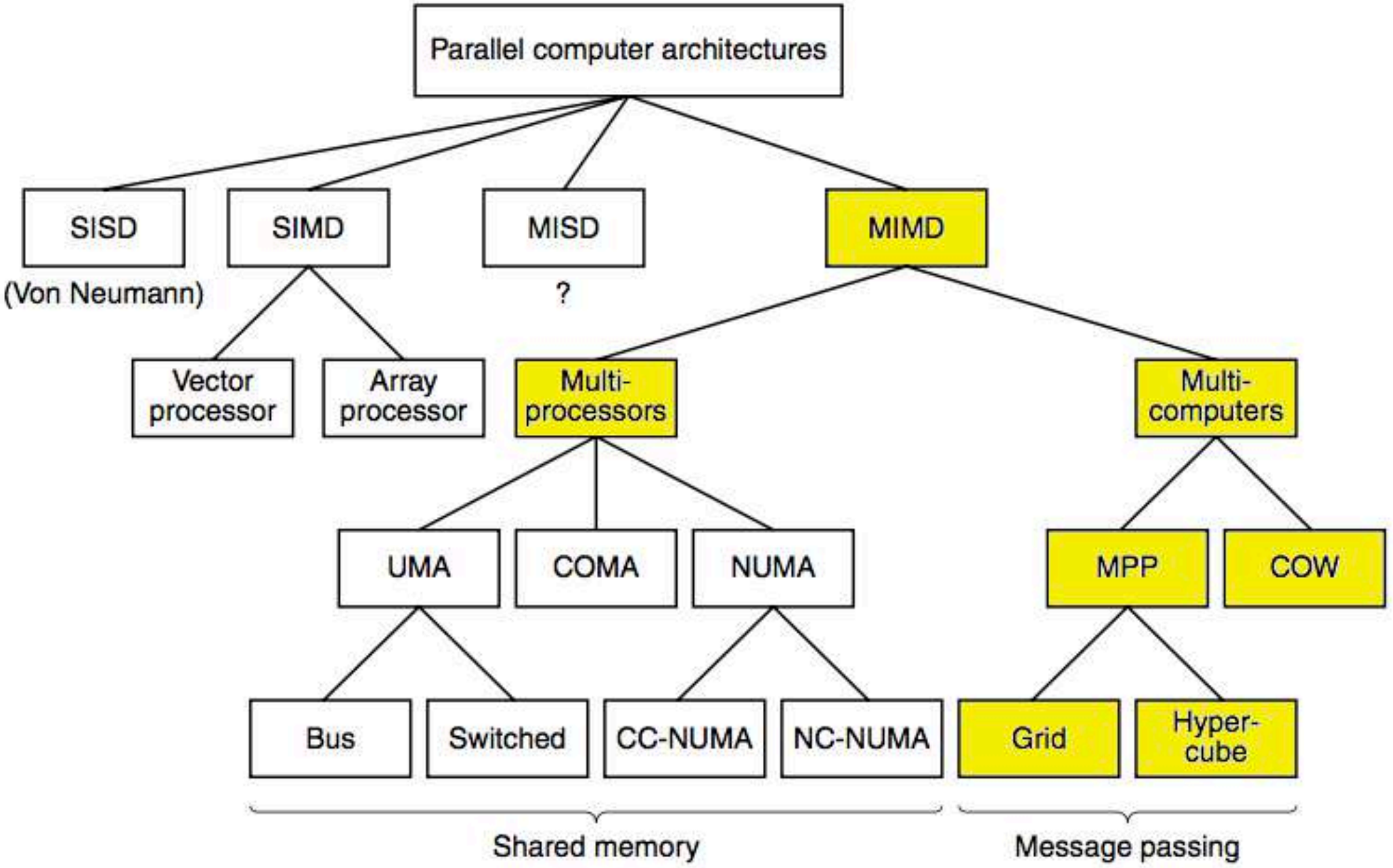
- **UMA**: Uniform Memory Access;
- **NUMA**: NonUniform Memory Access
- **COMA**: Cache Only Memory Access.

} Classificação quanto ao tempo de acesso à memória.

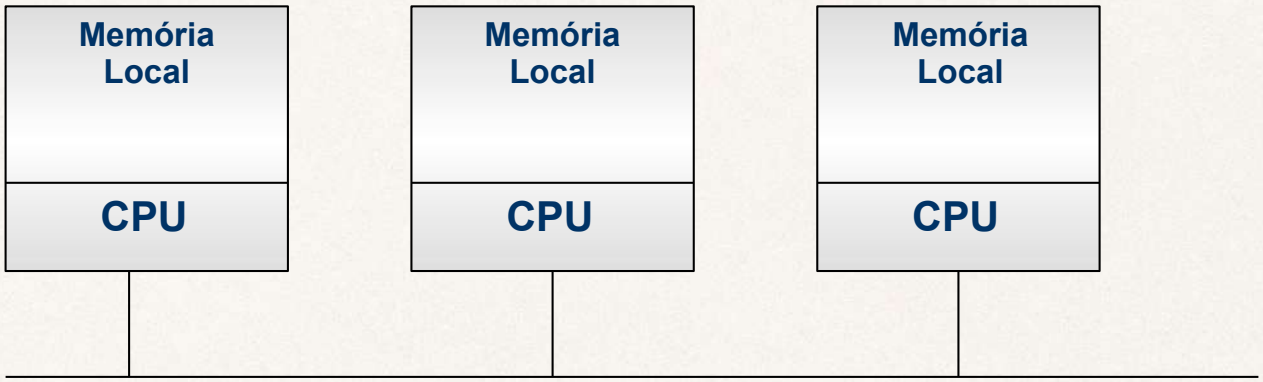


MIMD baseado em múltiplos computadores se divide em duas categorias:

- **MPP**: *Massive Parallel Processors*, constituídos de *hardware* específico com diversas CPUs acopladas;
- **Cluster Of Workstations**: computadores convencionais e servidores conectados em rede.



Multiple Instruction, Multiple Data



- Topologia “Lógica" em Barramento
- Memória Privada
- Comunicação CPU-CPU
  - Tráfego bem reduzido comparado a CPU-Memória
  - Requisitos típicos de um rede local convencional
    10 Mbps, 100 Mbps, 1 Gbps

Requisitos de Projeto

- Flexibilidade
- Confiabilidade / Disponibilidade
- Desempenho
- Escalabilidade

Requisitos de Projeto

- Transparência
  - Visão de um sistema centralizado
  - Encapsular Detalhes, como:
    - Localização
    - Migração
    - Replicação
    - Concorrência
    - Paralelismo

Conceitos de Software para SD

- *Software* fracamente acoplado

X

- *Software* fortemente acoplado

Software fortemente acoplado

- Rede é um “sistema único”;
- Uniprocessador Virtual;
- Kernel igual em todos os dispositivos
  - Processos se comunicam de forma “global”, independente de onde estiverem;
  - Esquema de proteção também global;
  - Interfaces para manipulação de processos precisa ser similar aos ambientes convencionais;
  - “System Calls” também são globais;

Software fracamente acoplado

- Máquinas e usuários independentes entre si
- Interação limitada a mensagens entre dispositivos
- Sistema Distribuído sobre rede local
- Compartilhamento de recursos

Software fracamente acoplado

- Em caso de falhas na comunicação
  - Computadores individuais mantém operação baseada em recursos locais;
- SW fracamente acoplado em HW fracamente acoplado
  - Sistemas operacionais de rede;
  - Heterogeneidade (estações e servidores conectados);
  - Cada estação tem seu próprio Sistema Operacional;
  - Execução local de aplicações e conexão remota a outras máquinas.

Software fracamente acoplado

- Aplicações Típicas
  - Compartilhamento de arquivos e dados
  - Compartilhamento de Recursos (ex.Impressão)
  - Serviços de Aplicação (ex.Banco de Dados)
  - Comunicação associado à aplicação
    - Compartilhamento restrito à informação;
    - "Visão" do arquivo compatíveis com o SO local.

Estilos básicos de arquitetura

V.1.11 - 2021.10

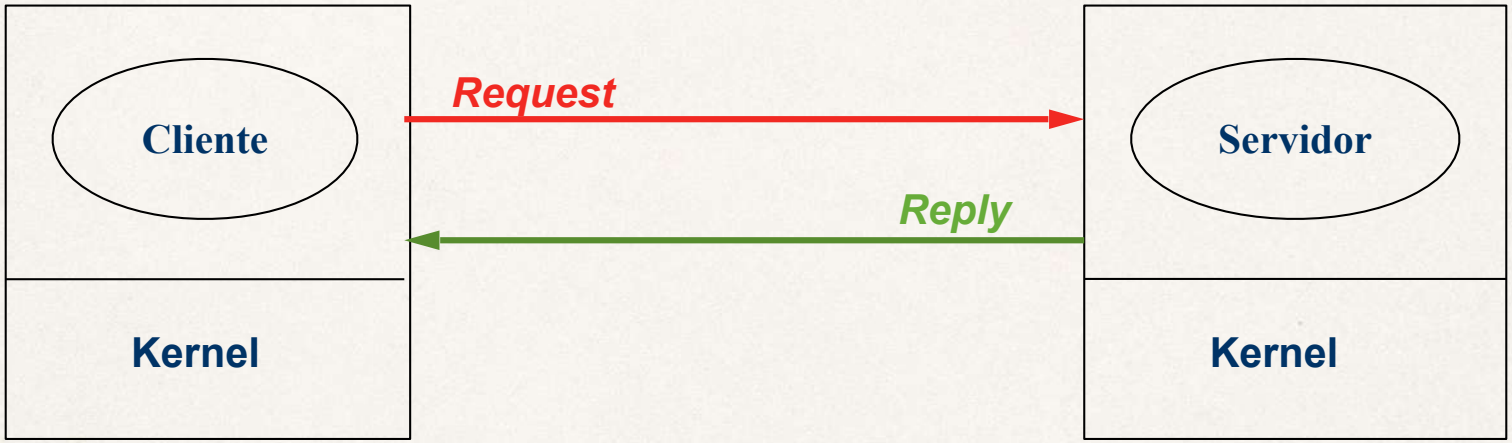
Estilos básicos de arquitetura

- Cliente-Servidor
  - Clientes interagem com servidores tipicamente hospedados em computadores distintos;
  - Servidores podem ser “clientes” de outros servidores;
  - Problema da escalabilidade (capacidade e largura de banda).
- *Peer-to-Peer*
  - Todos os processos envolvidos têm funções semelhantes;
  - Interação cooperativa, sem distinção entre “servidores” e “clientes”;
  - Tipicamente todos rodam aplicações iguais ou muito semelhantes, adaptadas para o SO local;

Desafios para os Sistemas Distribuídos Modernos

- Os nós não são mais estáticos, completamente separados e autonômos;
- Computação Móvel
  - Descoberta de serviços;
  - Capilaridade da Infraestrutura.
- IoT (*Internet of Things*);
- Computação e Serviços em Nuvem
- Segurança;

Arquitetura Cliente-Servidor



Arquitetura Cliente-Servidor

- Em redes locais de alta qualidade, pode ser baseada em protocolo simples (*Request-Reply*)
  - Evita complexidade e *overhead* dos modelos OSI ou TCP/IP;
  - Não orientado à conexão (conexão consome recursos);
  - Somente 3 camadas do OSI.
- Com protocolo simples, operações precisam ser "idempotentes"
  - Duas execuções consecutivas não causam problema.

Arquitetura *Peer-to-Peer*

- Todos os processos que constituem o sistema são iguais;
  - Funções a serem realizadas são representadas por todo processo que constitui o SD -> interação simétrica;
  - Cada processo agirá como cliente e servidor ao mesmo tempo.

Sistemas Híbridos

- Cliente-Servidor + Peer-to-Peer
  - Servidores de Borda;
  - Sistemas Distribuídos Colaborativos (*torrent?*)

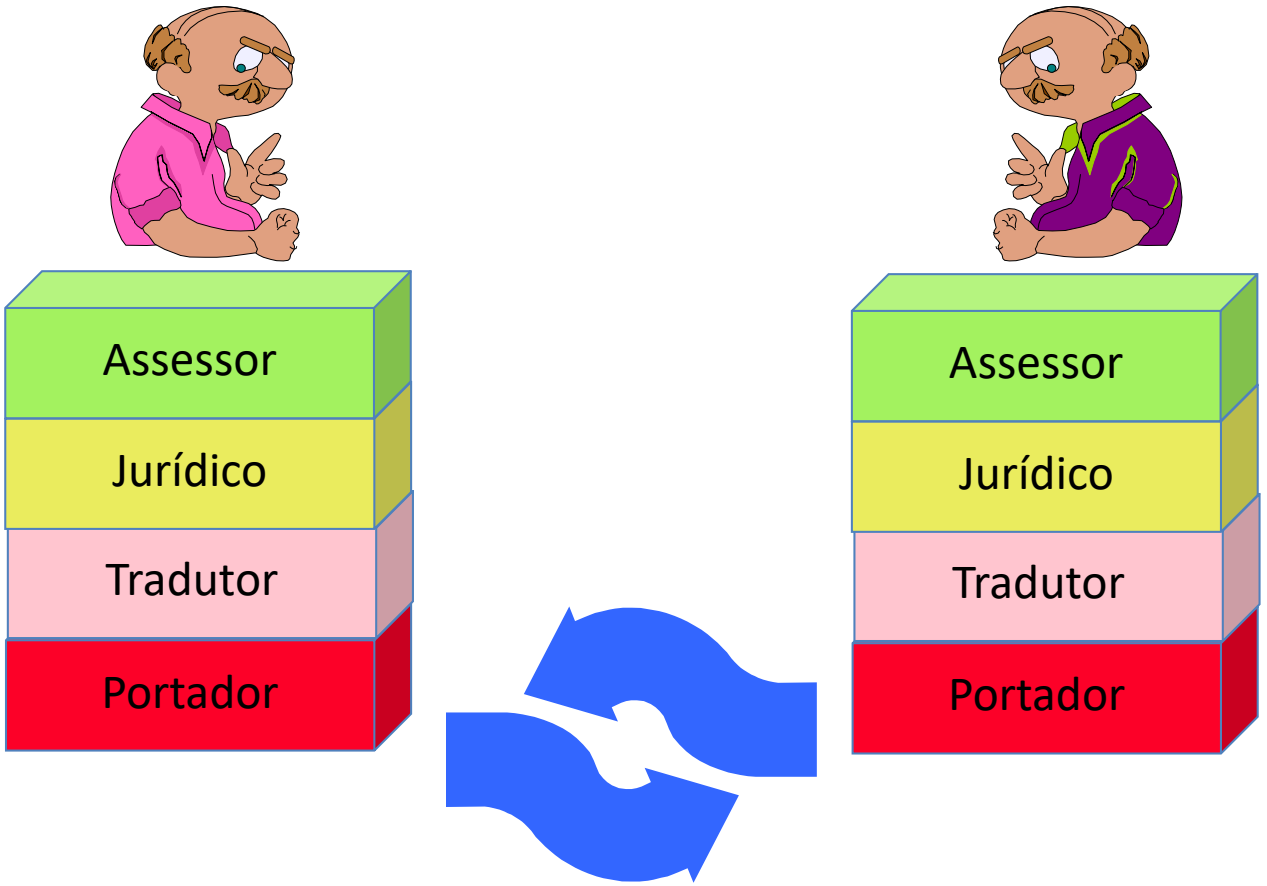
Comunicação entre Processos

- Multi-Computadores
  - Não há dispositivos nem memória compartilhada
  - Comunicação opera por troca de Mensagens
  - Protocolos de comunicação
    - Regras para a correta troca de mensagens
    - Interpretação das mensagens

Protocolos

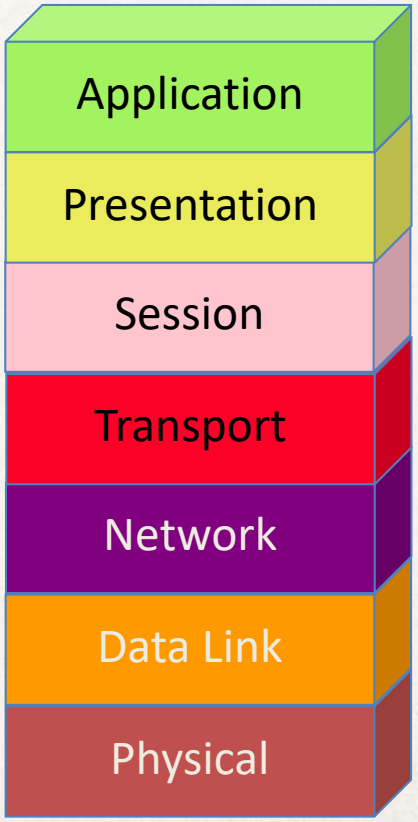
- A complexidade típica dos sistemas de comunicação exige a divisão das tarefas de comunicação entre diversos protocolos;
- Classificação Básica
  - Orientados a Conexão
  - Não orientados a Conexão
- Protocolos de diversos "níveis"
  - Camadas
  - Responsabilidades específicas na troca de mensagens
- Modelos de Referência em Camada
  - OSI - *Open Systems Interconnection*
  - TCP/IP - Padrão de mercado

Modelos em Camadas : Exemplo Clássico



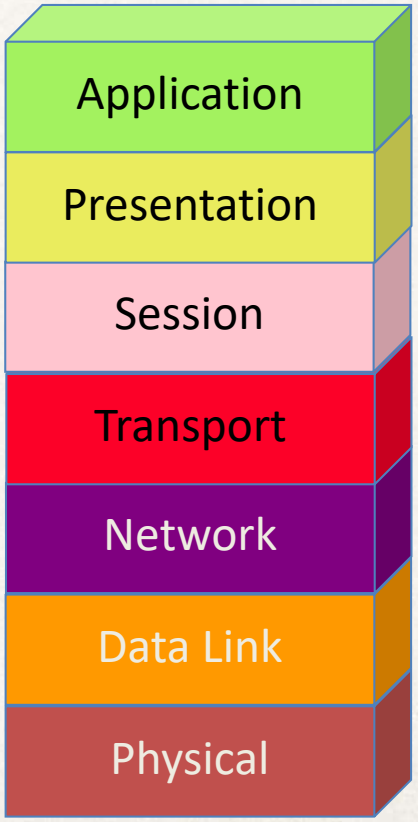
O modelo OSI

- Sete camadas. Por quê ?
  - Redução do tráfego entre as camadas;
  - Funções inequívocas;
  - Compatibilidade com os padrões de mercado.
- Comunicação Virtual entre camadas semelhantes;
- Inserção de Cabeçalhos;
- Funções de cada camada ?



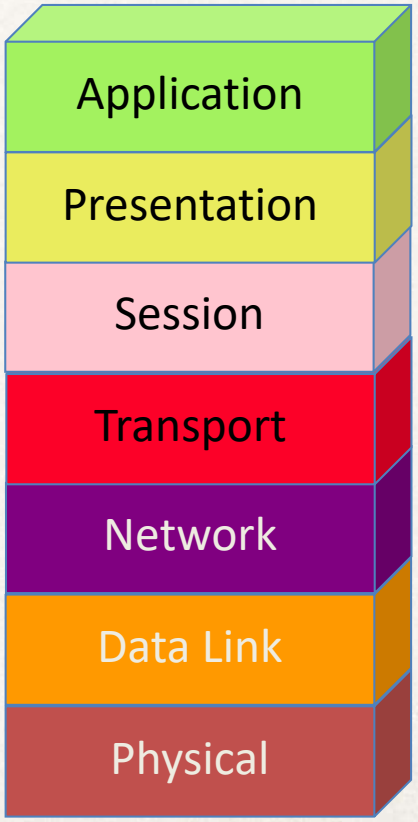
O modelo OSI - Funções Camadas

- Física (*Physical*)
  - Determina interfaces mecânica, elétrica e tempos;
  - É a camada onde efetivamente ocorre a comunicação entre emissor e receptor;
  - Domínio do cabeamento estruturado, engenharia elétrica.



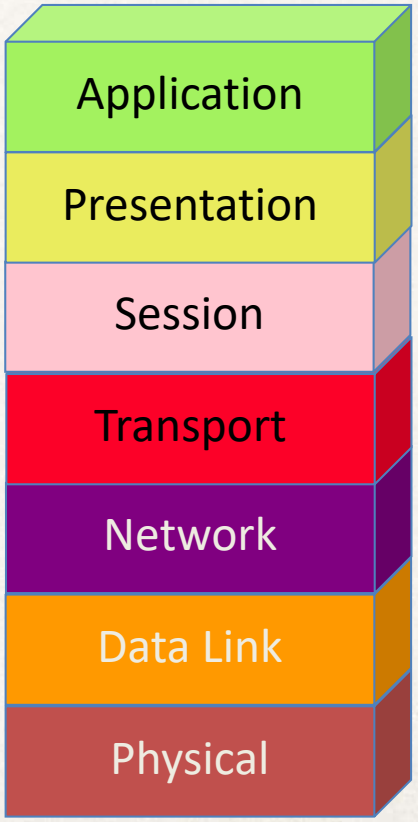
O modelo OSI - Funções Camadas

- Enlace (*Data Link*)
  - Transforma a camada física em um ambiente livre de erros;
  - Delimita e estabelece campos
    - Delimitadores por padrão físico, tamanho ou codificação (c/ misturadores)
  - Delgada nas redes mais modernas;
  - Subdividida nas redes IEEE802 (LLC e MAC);
  - Controle de fluxo.



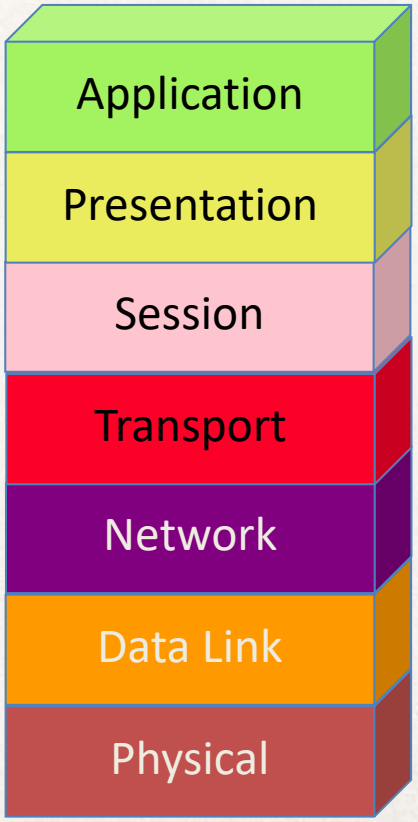
O modelo OSI - Funções Camadas

- Rede (*Network*)
  - É a camada da interligação entre "padrões de rede" diferentes;
  - Controle de operação e contabilização de recursos;
  - Delgada nas redes locais.



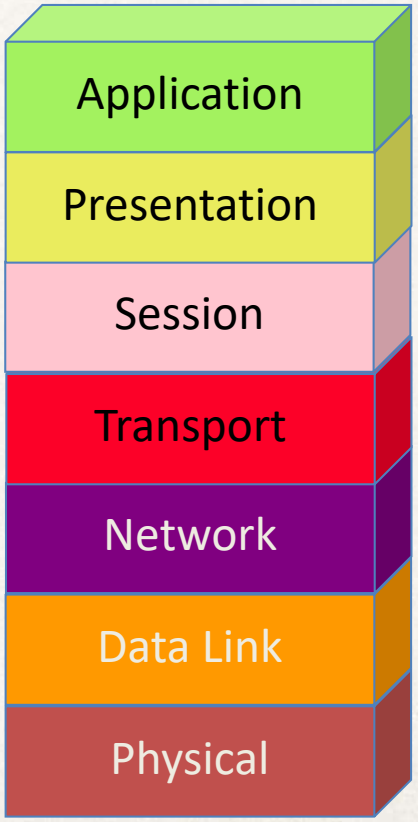
O modelo OSI - Funções Camadas

- Transporte
  - Primeira camada fim a fim !
  - Estabelece qualidade de serviço (QoS);
  - Estabelecimento conexões & multiplexação.



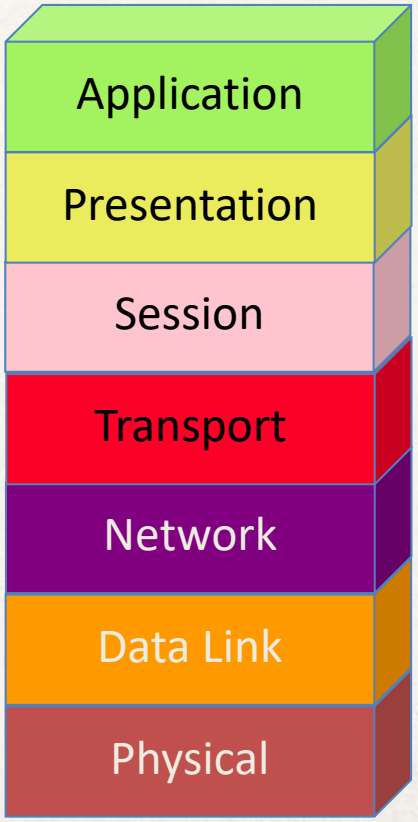
O modelo OSI - Funções Camadas

- Sessão (*Session*)
  - Determina pontos de checagem intermediária;
  - Controle de fluxo;
  - Sincronização.



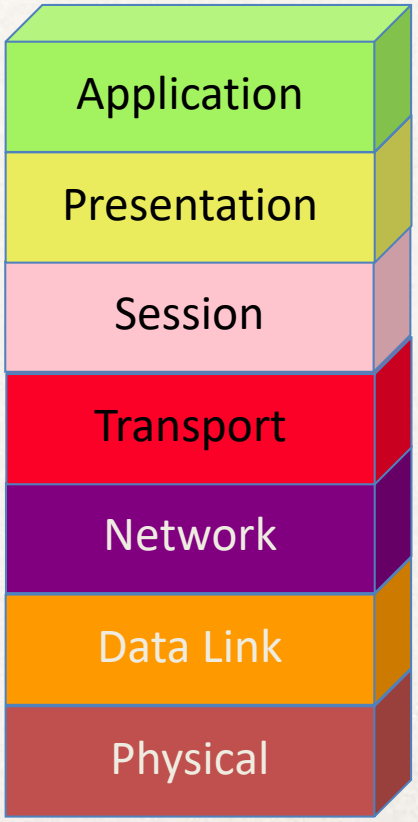
O modelo OSI - Funções Camadas

- Apresentação (*Presentation*)
  - Não está relacionada à comunicação em si;
  - Sintaxe e semântica;
  - Criptografia, compactação;
  - Estruturas de dados.

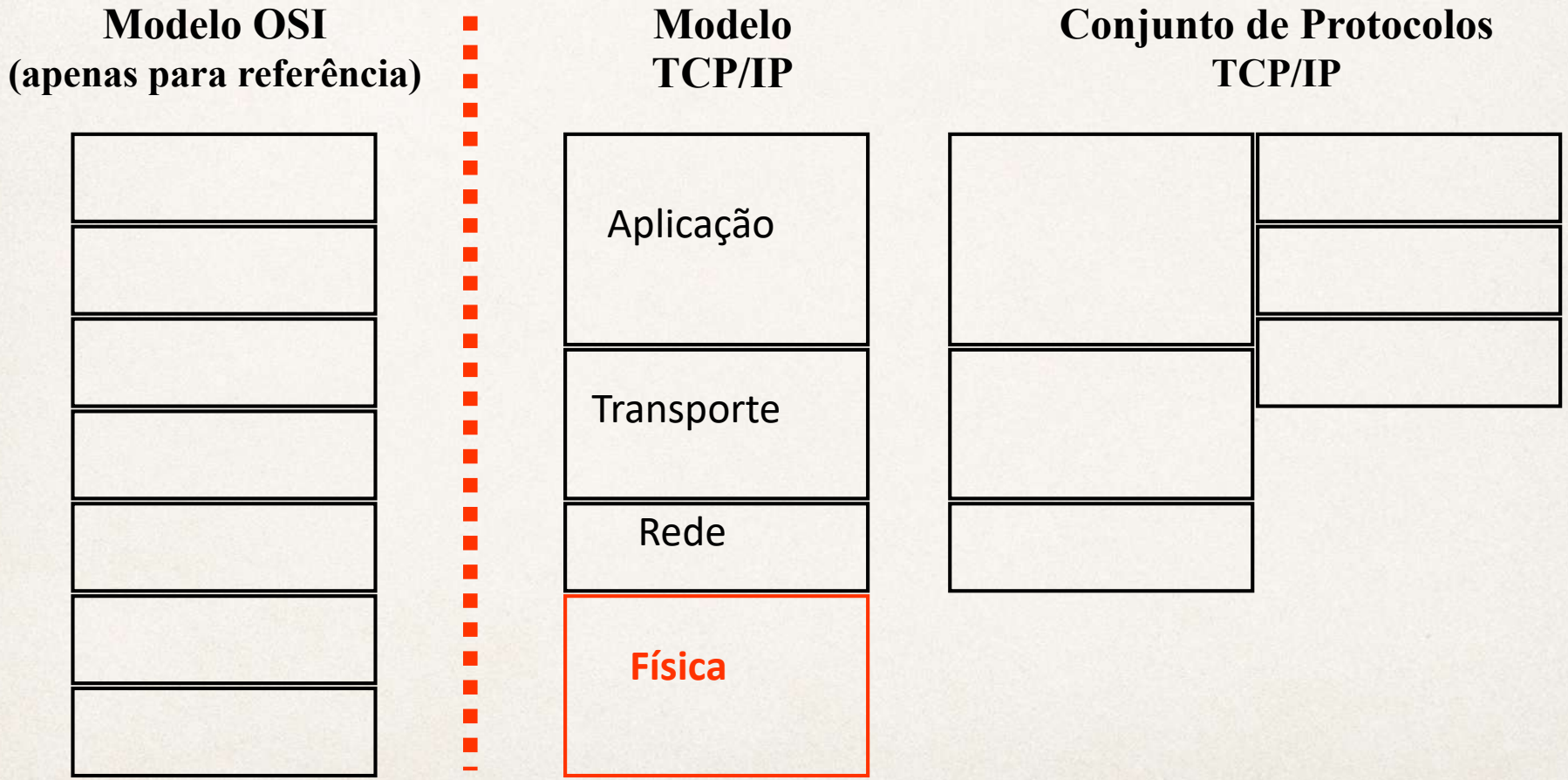


O modelo OSI - Funções Camadas

- Aplicação (*Aplication*)
  - Aplicações associadas à comunicação de dados :
    - Telnet
    - Serviços de Diretório
    - Correio eletrônico
  - Serviços de Sistemas Operacionais de Rede
    - Serviços de Arquivo & FTP
    - WEB Server, WEB cache etc

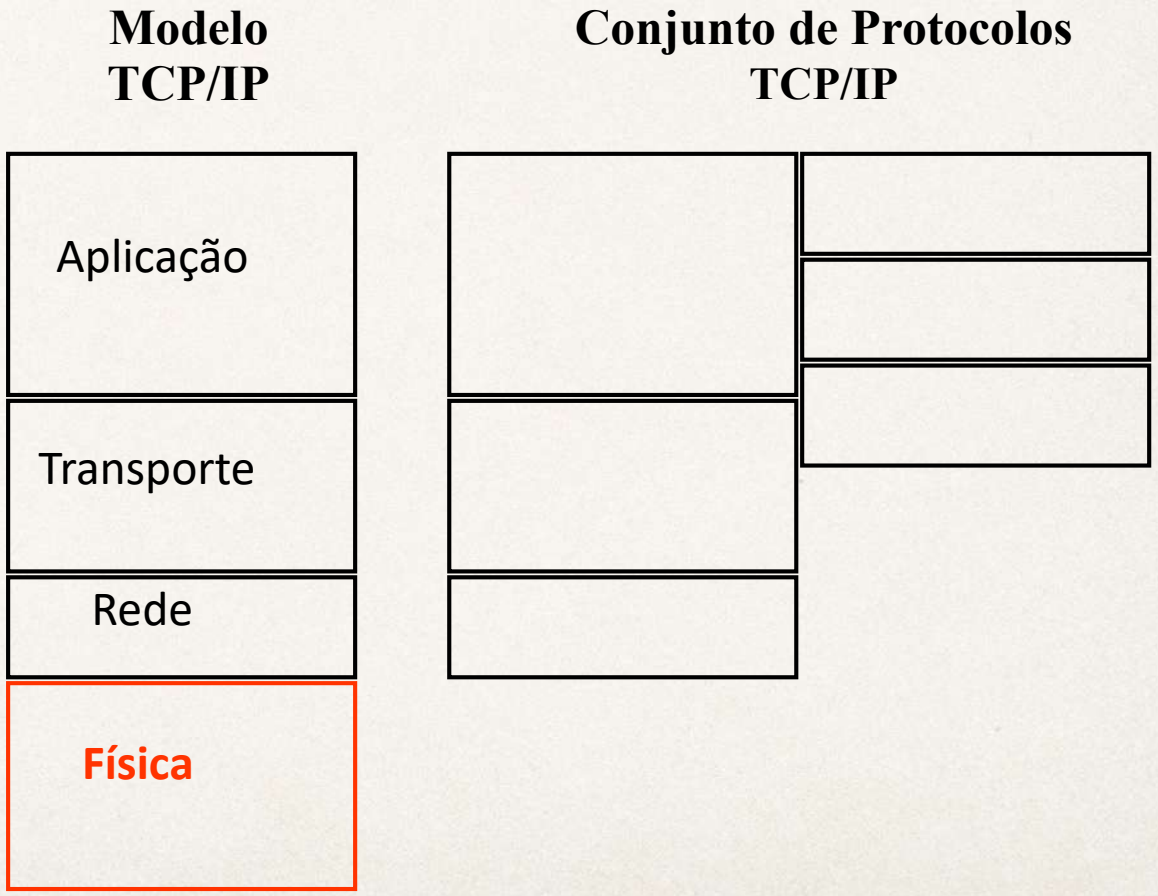


Comparando os modelo OSI e TCP/IP



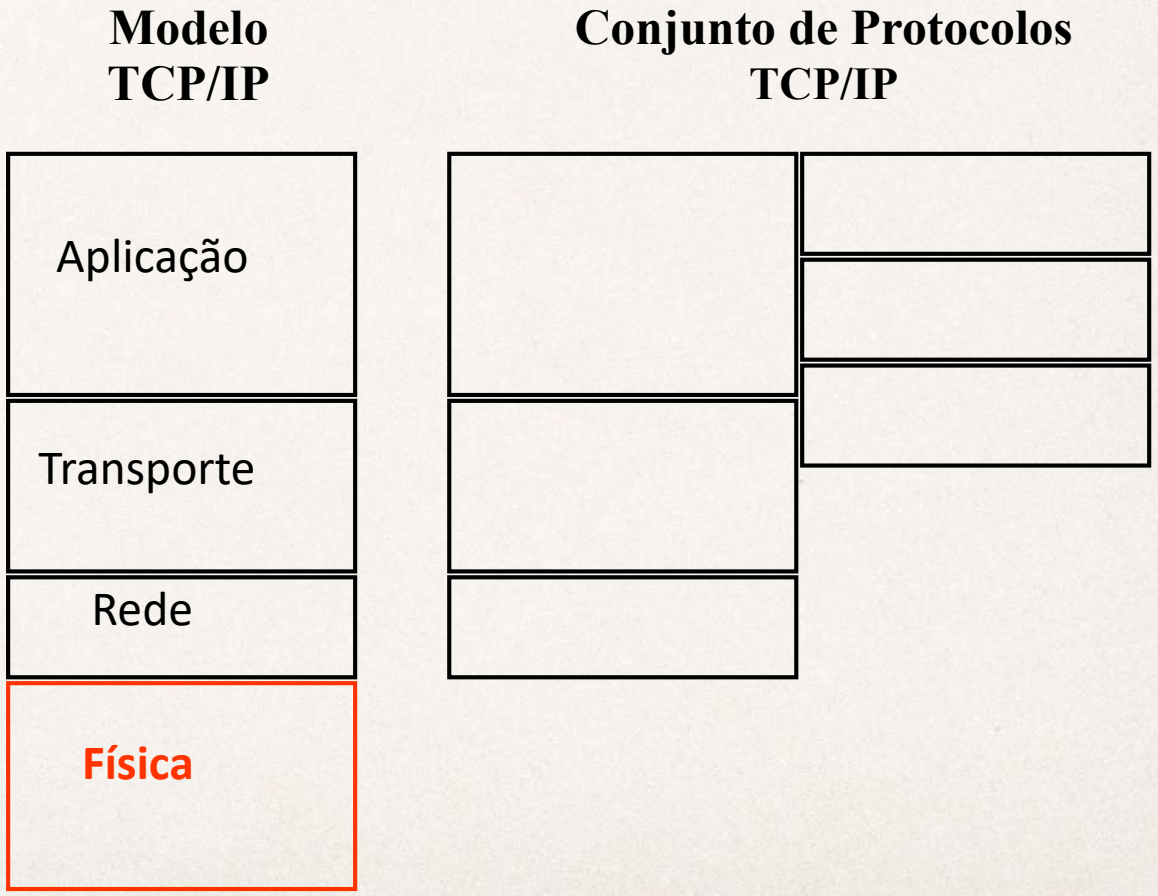
Comparando com o modelo OSI

- Independência das camadas inferiores
  - Implementação em LAN e WANs é mais simples;
  - Até mesmo padrões de rede local que não são 100% compatíveis com o modelo OSI podem implementar o protocolo (ATM, por exemplo).



Comparando com o modelo OSI

- Número de camadas
  - As camadas de rede e de transporte são as únicas semelhantes ao modelo OSI.
  - A camada de Aplicação incorpora funções das 3 camadas superiores do modelo OSI.



Endereçamento: Máquinas e Processos

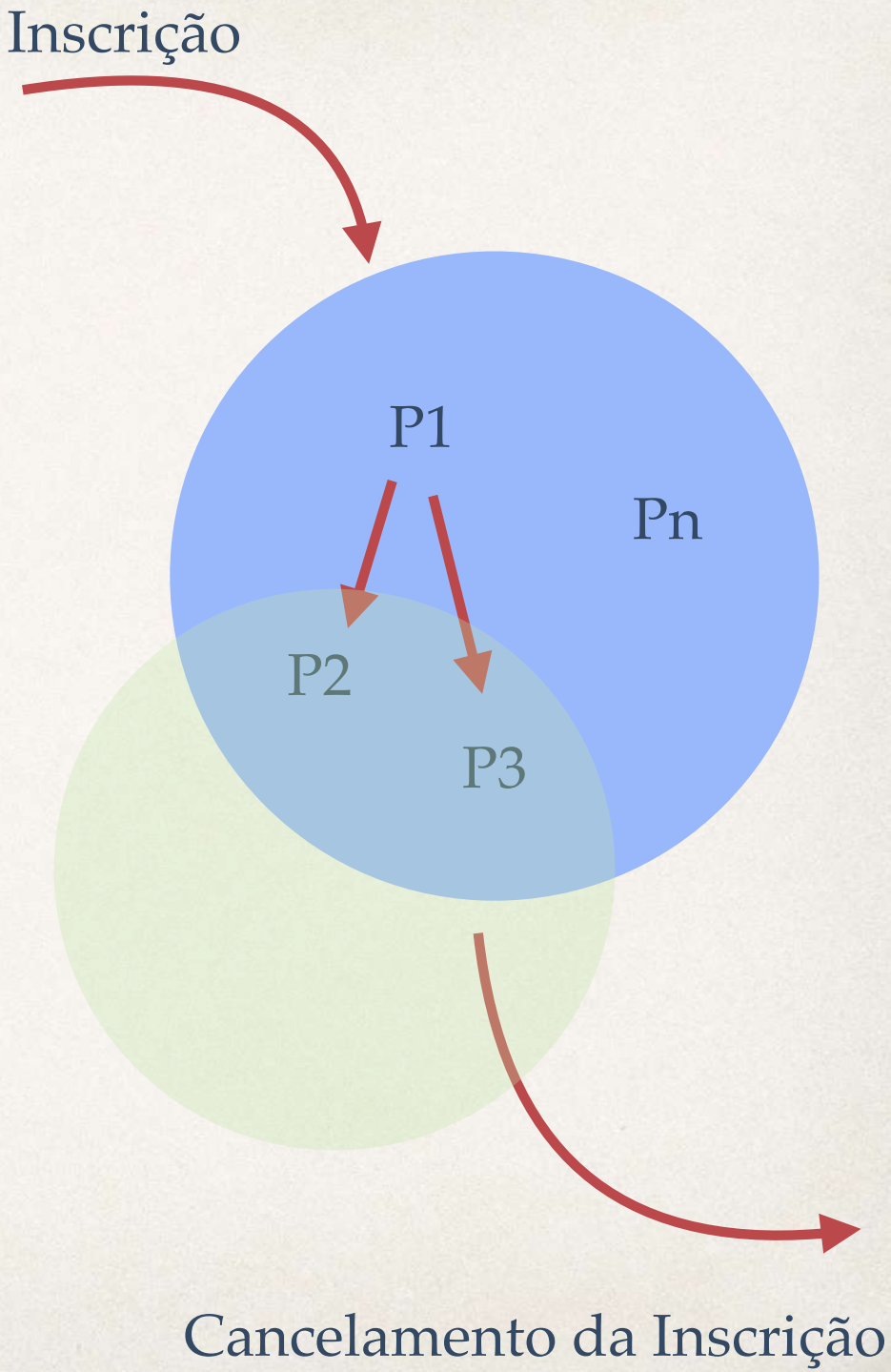
- Tipicamente, considerando o modelo de referência OSI, implementa-se endereçamentos diferenciados em 3 das camadas do modelo: Enlace, Rede e Transporte.
- Dois desses endereços identificam o dispositivo computacional; um deles identifica o processo/*thread* envolvido na comunicação.

Aula 03

V.1.11 - 2021.10

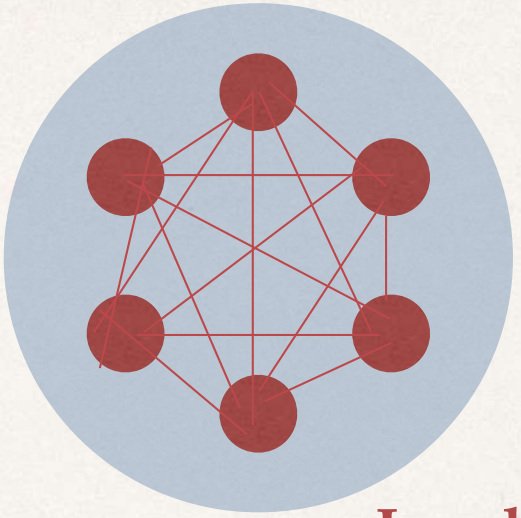
Conceito de Grupos

- Abertos e Fechados
  - Fechados aceitam mensagens unicast externas para membros do grupo.
- Unicast, Multicast e Broadcast
  - Fácil de implementar X Tráfego elevado;
  - Tráfego pequeno X Controle dos Grupos;
  - Tráfego pequeno X Validação do endereço de destino em todos os processos.
- Gerência por Servidor ou Distribuída

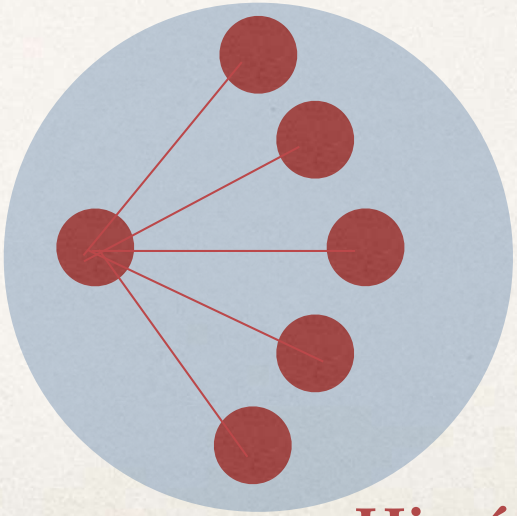


Conceito de Grupos

- Tomada de Decisões
  - Grupos Igualitários e Hierárquicos
- Igualitários
  - 50% dos votos + 1
- Hierárquicos
  - Processo coordenador



Igualitário



Hierárquico

Grupo Igualitário

- Estrutura simétrica, razoavelmente protegida contra falhas de qualquer processo, já que os demais podem continuar o trabalho;
- Pode sofrer de problemas de performance devido à tomada de decisões exclusivamente por votação.

Grupo Hierárquico

- Possíveis falhas no coordenador provoca parada momentânea, enquanto se elege um novo coordenador;
- Detecção de falhas do coordenador precisa ser implementada de forma eficaz;
- Rapidez decisória.

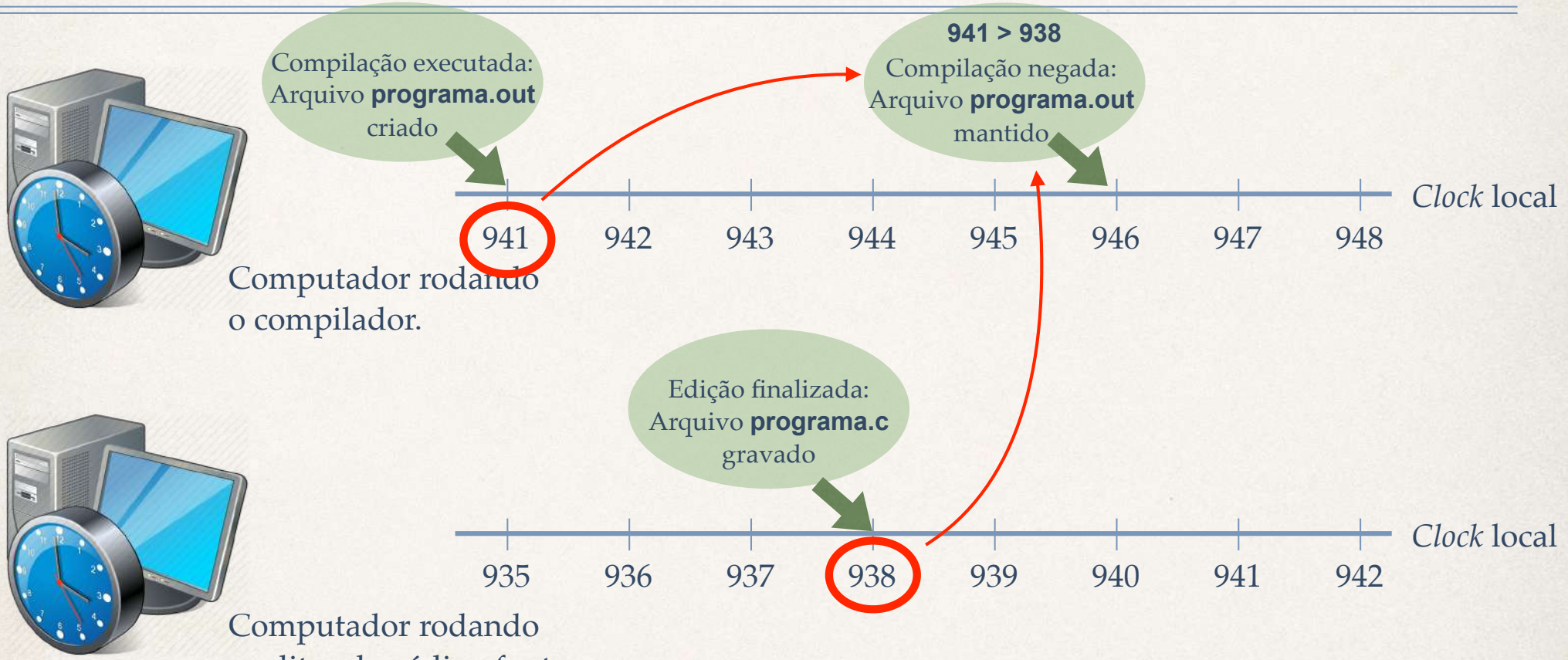
Atomicidade em Mensagens

- Ou a mensagem chega para todos (multicast / broadcast), ou não chega para nenhum
  - Requisito para sistemas distribuídos?
- Implementação?
  - Falhas são possíveis por diversos motivos;
  - Como garantir atomicidade? Ver ABCAST.

Sincronismo Sistemas Distribuídos

- O problema do tempo
  - Componentes precisam cooperar e trocar informações;
  - Exemplos:
    - Operações envolvendo múltiplos arquivos distribuídos;
    - Transações financeiras e de comércio eletrônico.
- É necessário implementar controles para garantir o sincronismo entre os componentes.

Sincronismo: Arquivos Distribuídos



V.1.11 - 2021.10

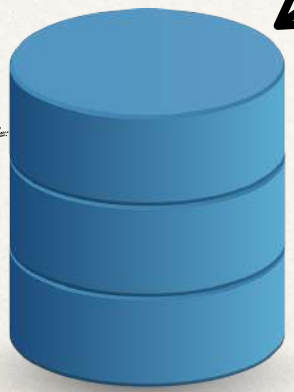
Sincronismo: Aplicações Financeiras

Quem paga os R$ 120,00?

Atualização 1:
Crédito de Juros de 1,2%

Atualização 2:
Crédito de R$ 10.000,00

Na ordem correta:		
Saldo		10.000,00+
Juros	120,00+	10.120,00+
Crédito	10.000,00+	20.120,00+
Saldo Final		**20.120,00+**



Bancos de Dados Replicados

Na ordem inversa:		
Saldo		10.000,00+
Crédito	10.000,00+	20.000,00+
Juros	240,00+	10.240,00+
Saldo Final		**20.240,00+**

Saldo Inicial:
R$ 10.000,00

V.1.11 - 2021.10

Sincronismo: O Tempo

- Em Sistemas Distribuídos, cada dispositivo possui seu próprio relógio.
- *Clock* Lógico X *Clock* Físico
  - *Clock* Lógico garante apenas o sincronismo;
  - *Clock* Físico relaciona processos com o horário.
- Clock Lógico
  - Se duas máquinas não interagem, não precisam estar sincronizadas;
  - Os processos em máquinas diferentes podem exigir uma determinada ordem de execução, e não um horário específico;
  - A sincronização, então, pode ser relativa.

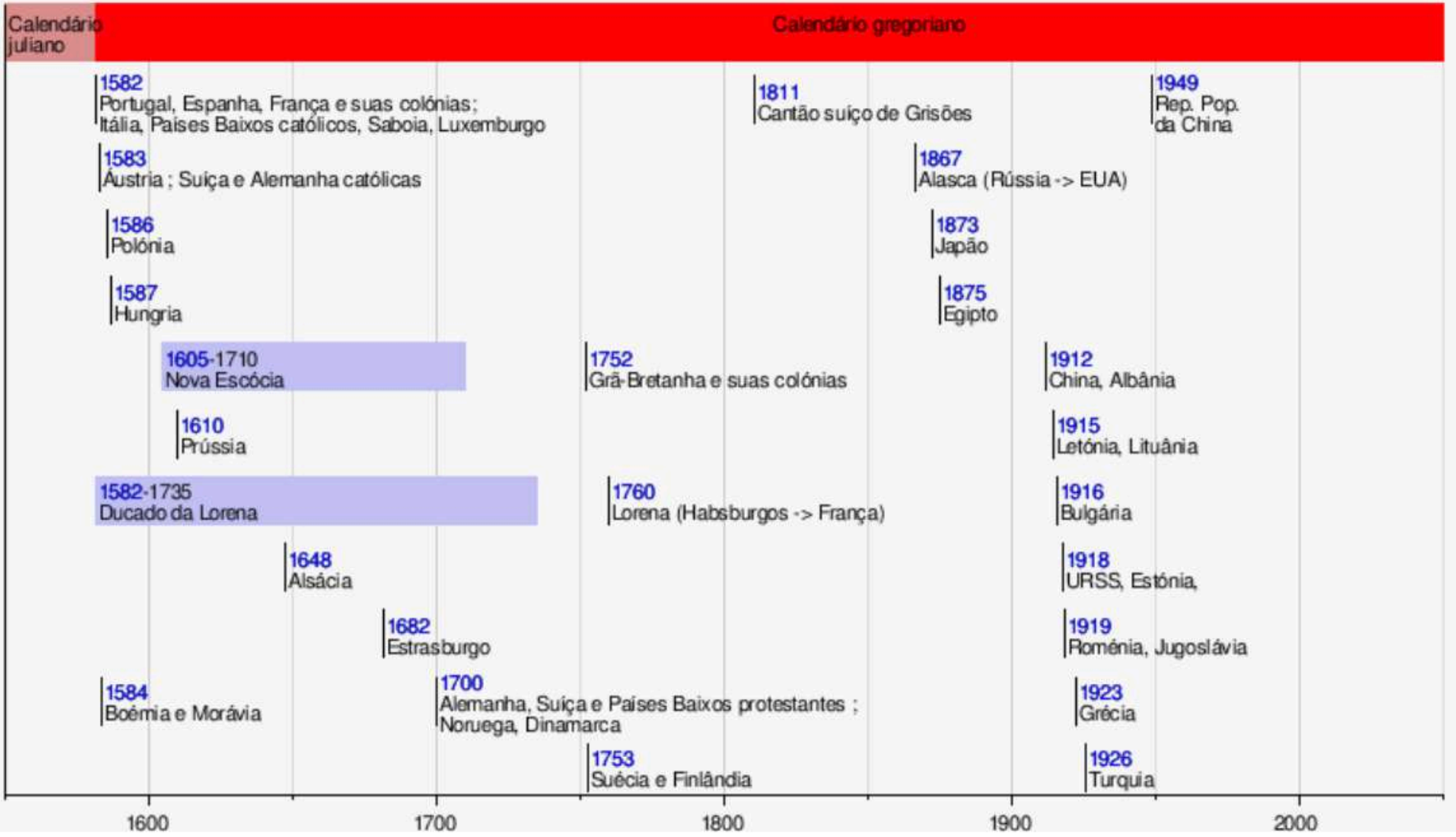
O Tempo e o *Clock* Físico

- O tempo é medido com base na astronomia;
- Um segundo = 1/86400 do dia solar
- 1 dia solar = 1 rotação da Terra
  - Curiosidade: o período não é constante, e tem ficado mais longo. A 300 milhões de anos atrás, um ano durava cerca de 400 dias. Redução da velocidade tem relação com o atrito do mar com a atmosfera.
- Em 1948, foi inventado o relógio atômico baseado em Césio 133
  - Muito mais preciso (1 segundo = 9.192.631.770 transições do Césio);
  - Igual ao segundo solar no ano da criação do relatório);
  - Hoje está 3ms adiantado em relação ao ano solar (anos ficando mais longos).
- TAI - *Temps Atomique International* (francês)
  - Média de todos os relógios atômicos (400, em cerca de 50 países), desde 01 de janeiro de 1958.
- UTC - Universal Time Coordinated
  - Baseado no TAI, mas mantém sincronismo com o tempo astronômico;
  - Para manter o sincronismo, introduz saltos de segundo(s) - *leap seconds*.

O tempo na História

- As unidades de tempo também foram afetadas por eventos históricos;
- Até 1582, utilizávamos o Calendário Juliano (criado pelo imperador romano Júlio César em 45 a.c.);
- Em 1582, o Papa Gregório XIII introduziu uma mudança através da bula papal *Inter gravíssimas*;
  - Os dias 5 a 14 de outubro de 1582 foram "pulados";
  - Anos seculares são bissextos apenas se múltiplos de 400.

Adoção do Calendário Gregoriano



Clocks Físicos

- Computadores não utilizam as unidades de tempo astronômicas;
- Relógios são tipicamente imprecisos
  - Computadores utilizam relógios baseados em cristais de *quartzo* (erro típico de 1 em 1 milhão). Isso gera diferenças entre diferentes *clocks*:
    - Entre 2 relógios em determinado momento: *skew;*
    - Entre os ritmos de 2 relógios: *drift;*
  - Valores típicos das imprecisões:
    - Cristal de Quartzo convencional: *drift* de 1 segundo em 11~12 dias (10^{-6});
    - Cristal de Quartzo de alta precisão: *drift* de 1 segundo em no mínimo 4 meses (10^{-7} ~ 10^{-8});
    - TAI: *drift* de 1 segundo a cada 10^{13} (maior que a duração do universo)
      - Os horários UTC são divulgados via rádio (terrestre e satélite), com erro entre 0,1 e 10ms
      - Tipicamente o TAI e UTC são utilizados por um computador "mestre", ou diretamente.

Implementação nos Computadores

- Timer
  - Registradores associados
    - Counter (Contador)
    - Holding Register (Armazenamento)
- Counter (Contador)
  - Decrementado por cada oscilação do cristal
  - Quando Counter = 0
    - Interrupção
    - Recarga do Counter pelo valor do Holding Register
  - Interrupções
    - Clock Tick
      - 60 vezes por segundo, por exemplo

Clock x Timer

- Hardware
  - Armazena Data e Hora padrão
  - Ajuste manual pelo usuário
    - Armazenado como deslocamento da data e hora padrão em número de *ticks*
  - Clock tick
    - Adiciona 1 ao tempo da memória
      - Atualiza relógio

Solução de Lamport

- Relação "happens-before"
  - $a \rightarrow b$ é lida "a acontece antes de b"
  - Todos os processos concordam que o evento a ocorreu antes de b
    - Se a e b são eventos num mesmo processo e "a acontece antes de b", então $a \rightarrow b$ é verdadeira
    - Se a representa um SEND de P1 e b um RECV de P2, sendo P_1 e P_2 processos distintos, então $a \rightarrow b$ é verdadeira.

Solução de Lamport

- Relação "happens-before"
  - Transitividade
    - $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$

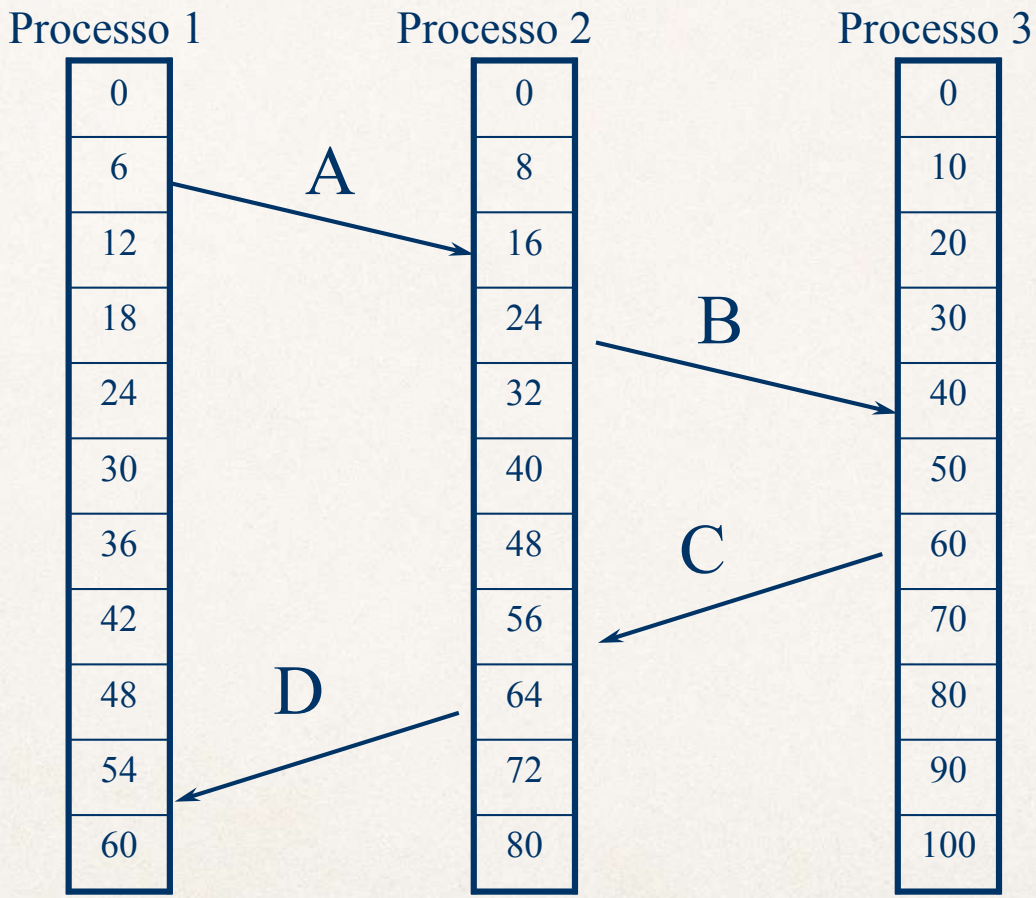
- Se os eventos x e y ocorrem em processos que não trocam mensagens (concorrentes)
  - É falso que:
    - $x \rightarrow y$ e $y \rightarrow x$

Solução de Lamport

- Se $a \rightarrow b$, então $C(a) \rightarrow C(b)$
  - *Clock de a é menor que o de b*
    - Clock como tempo

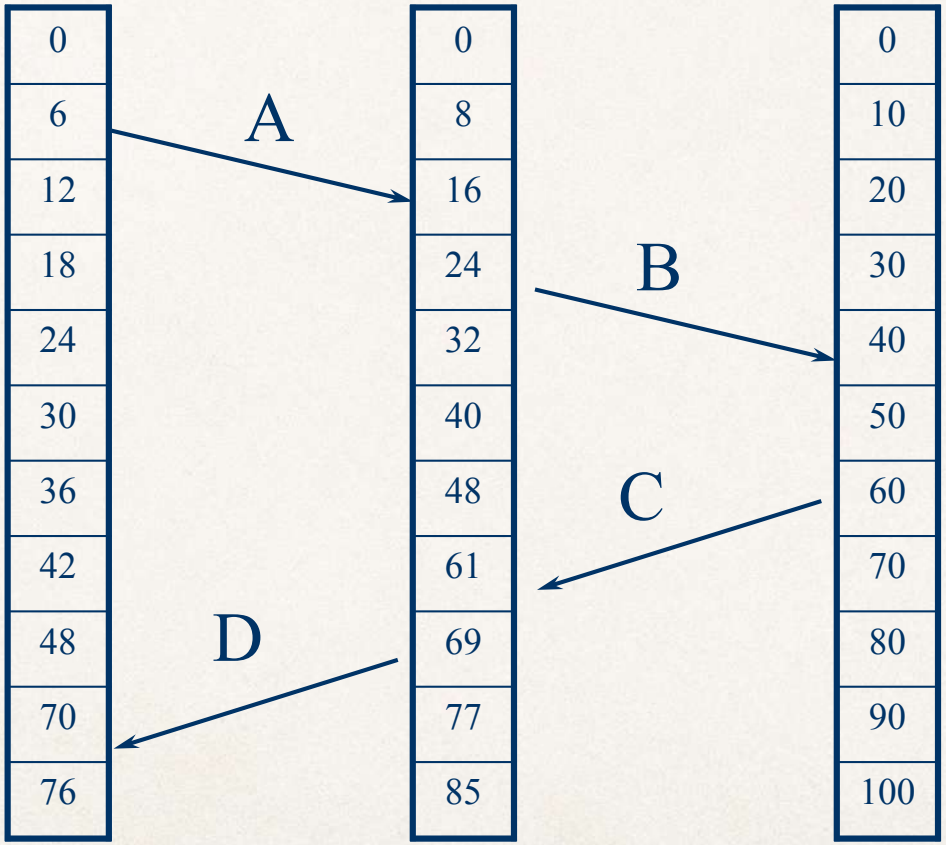
- Clock sempre é incrementado, nunca decrementado.

Solução de Lamport



Processos 1, 2 e 3 executam em máquinas distintas com diferenças de velocidade de Clock.

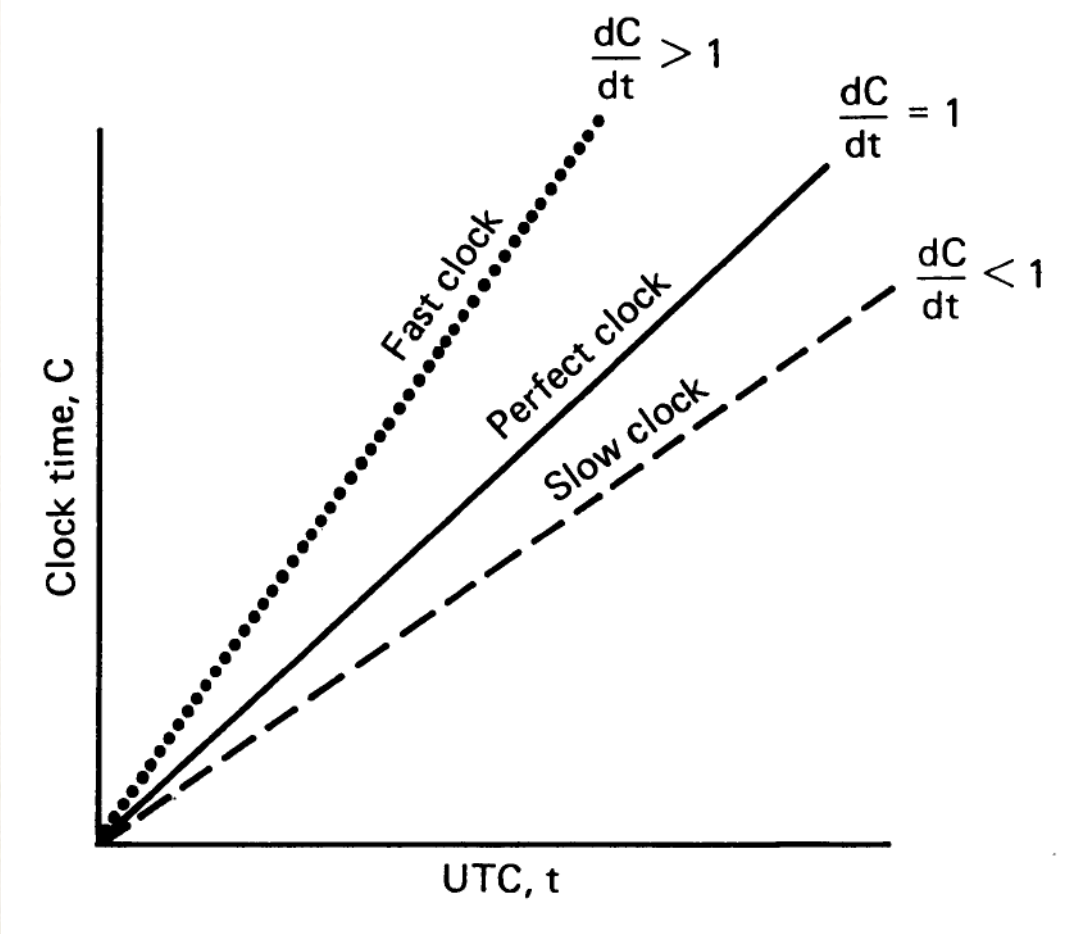
Solução de Lamport



Solução de Lamport

- Requisitos para Tempo Global
  - Entre dois eventos deve haver ao menos um Clock Tick.
    - Múltiplos eventos de SEND ou RECV sucessivos
      - Avançar o Clock
  - Dois eventos não devem ocorrer no mesmo tempo
    - ID do processo

Clock Físico



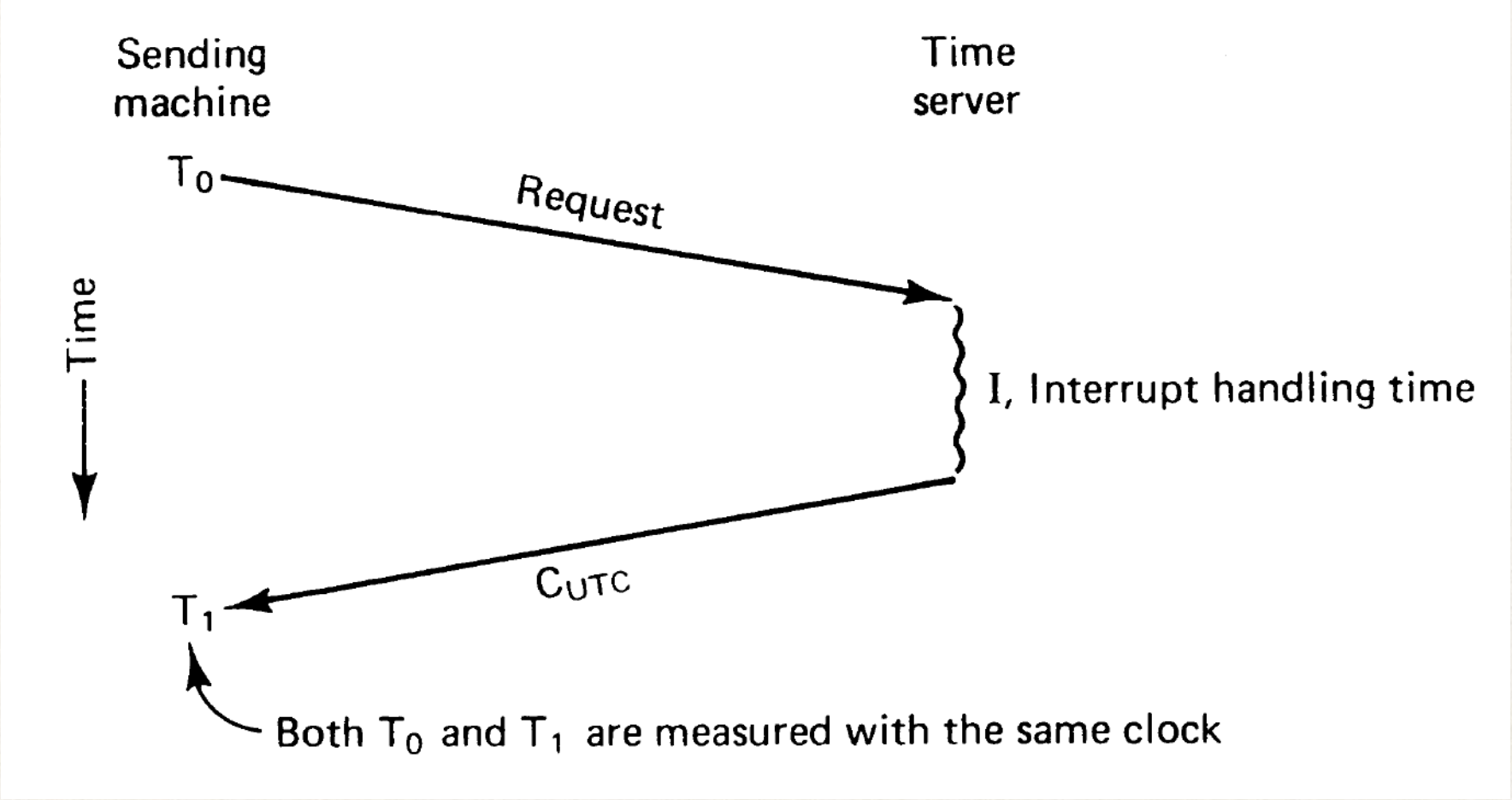
Algoritmo de Cristian

- Servidor de Tempo
  - Receptor WWV
  - Sincronização Externa (fonte externa)
- Clientes questionam a hora do servidor
  - Resposta imediata
- Problemas
  - Ajuste de Clock
    - Clock nunca deve ser atrasado
    - Overhead do Kernel do Servidor

Algoritmo de Cristian

- Relógio do computador *Ci* é sincronizado com uma fonte externa S:
  - $|S(t) - Ci(t)| < D$ para $i = 1, 2, \ldots N$ num intervalo I de tempo real
  - Os relógios *Ci* são exatos dentro do limite *D*

Algoritmo de Cristian



Aula 04

V.1.11 - 2021.10

Algoritmo de Cristian

- **Um servidor de tempo ST recebe informação de uma fonte UTC**
  - Processo *p* requisita tempo em *T0* e recebe *t* em *T1* de *ST*
  - *p* ajusta o seu relógio para $t + (T1 - T0)/2$
  - Precisão é $\pm ((T1 - T0)/2 - min)$:
  - O instante mais recente em que ST coloca *t* na mensagem *mreply* é *min* depois de *p* enviar *mrequest*
  - O tempo mais atrasado foi *min* antes de *mreply* ter chegado ao processo *p*
  - *O tempo segundo o relógio de ST quando mreply* chega a *p* está no intervalo $[t+min, t + (T1 - T0) - min]$

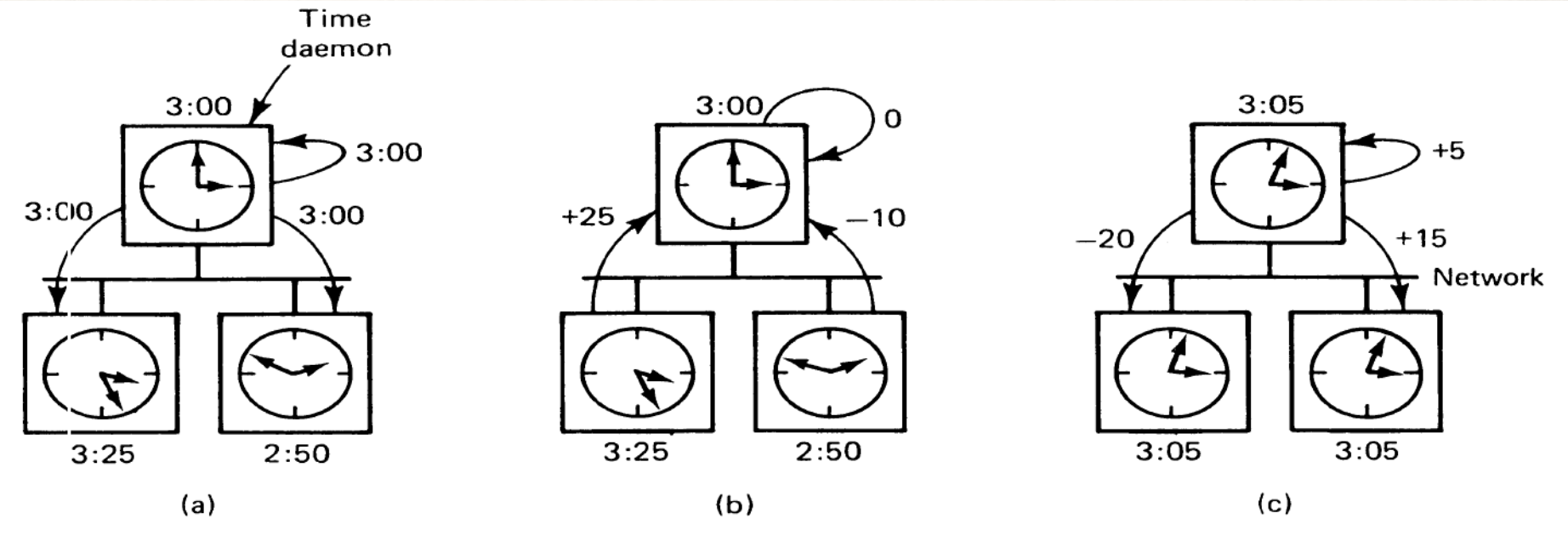
Algoritmo de Berkeley

- Servidor Ativo
  - Sincronização Interna (sincronização entre relógios)
    - Relógios podem não estar sincronizados externamente:
      - Pode existir ***drift*** coletivo
  - Questiona todas as máquinas pela hora local
  - Ajuste pela média
    - Avanço do relógio
    - Atraso via mudança de *ticks*

Algoritmo de Berkeley

- Os relógios de vários computadores sincronizam-se entre si:
  - $| Ci(t) - Cj(t) | < D$ para $i = 1, 2, \ldots N$ num intervalo I de tempo real
  - Os relógios *Ci e Cj estão sincronizados dentro do limite D*

Algoritmo de Berkeley



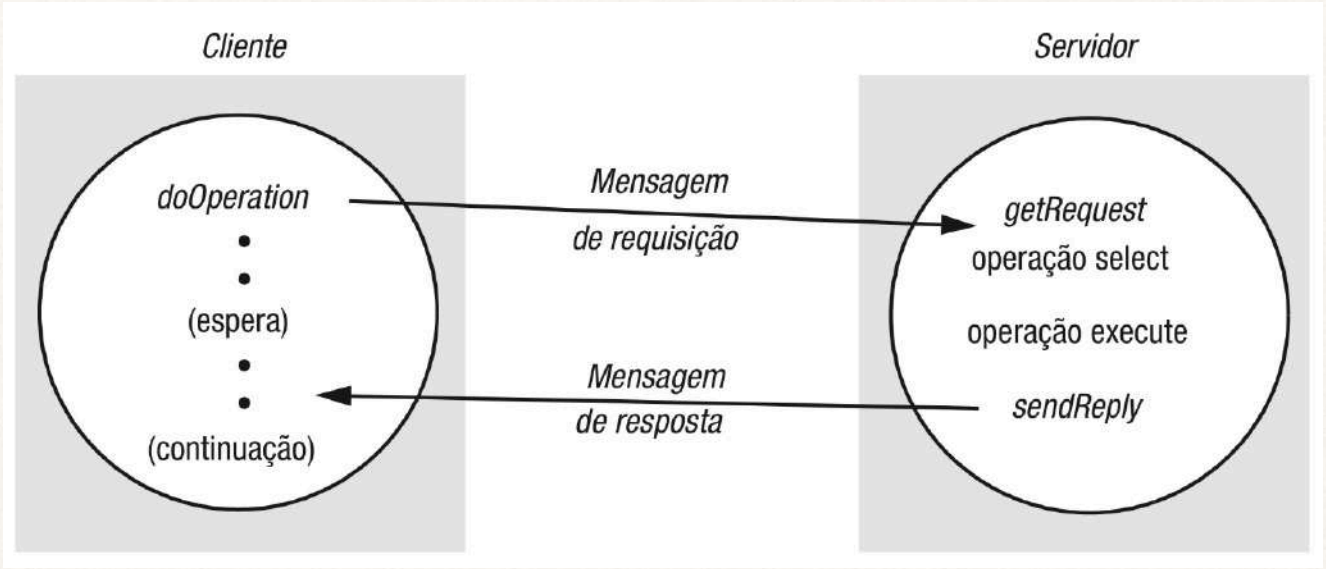
Média de Horário

- Solução descentralizada
  - Cada máquina transmite, via broadcast, sua hora corrente
  - Cada máquina coleta o envio de todas as outras.
    - Após receber todas as mensagens, computa nova hora.
    - Soluções:
      - Média.
      - Média com descarte de extremos.

Invocação Remota

- Protocolos Requisição-Resposta - tipicamente sobre UDP - TCP é muito pesado
  - Confirmações são redundantes;
  - No estabelecimento das conexões, troca-se mais um par de mensagens para substituir os mecanismos do TCP;
  - Controle de Fluxo tipicamente redundante (fluxos pequenos)
- RPC (*Remote Procedure Call*)
- RMI (*Remote Method Invocation*)

Primitivas de Requisição / Resposta



- *doOperation* invoca operações remotas, especificando o endereço do servidor remoto e a operação a ser invocada. Após execução, o processo é bloqueado até que o servidor execute a operação e transmita uma mensagem de resposta;
- *getRequest* é um processo servidor que aguarda requisições;
- *sendReply* envia a mensagem de resposta ao cliente.

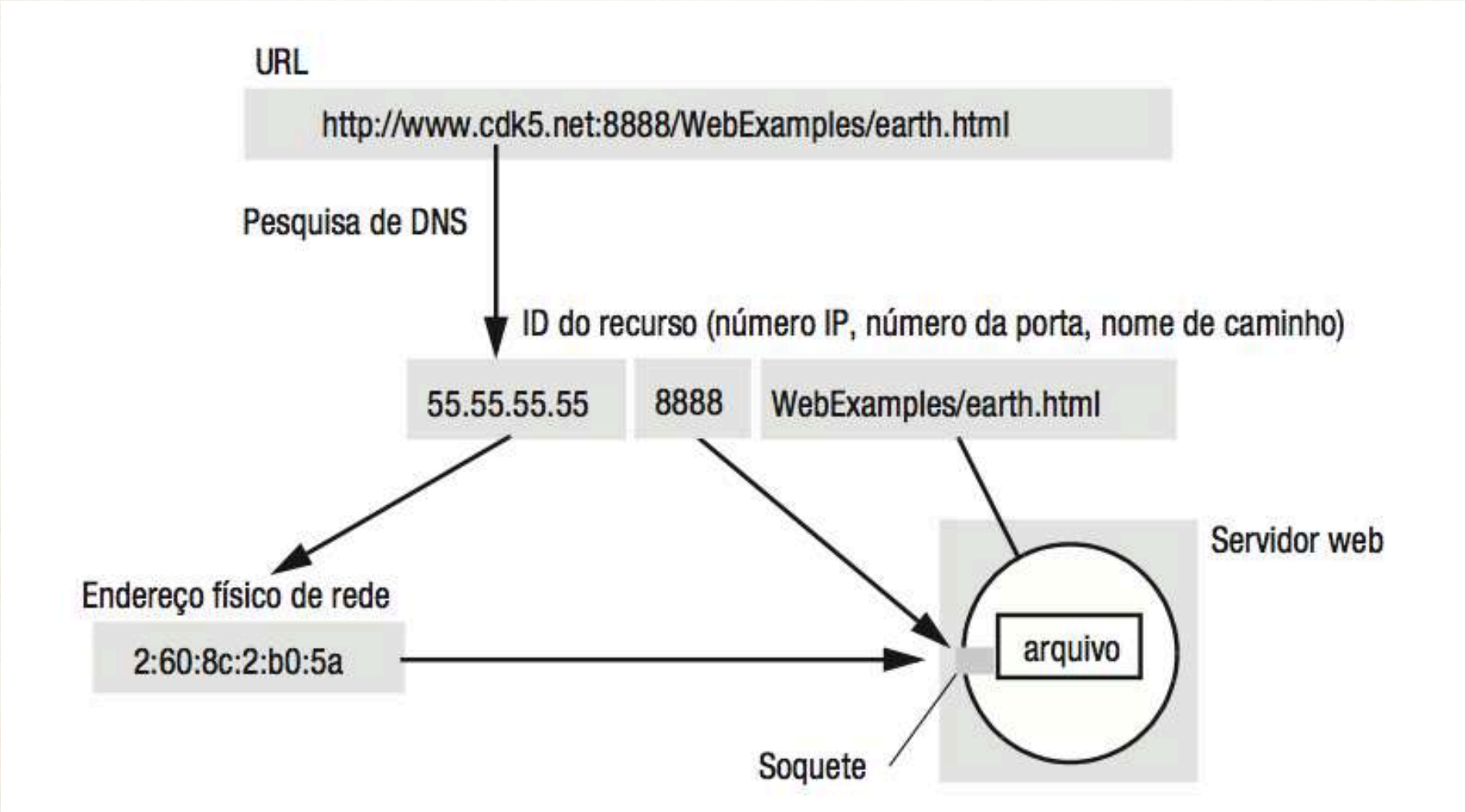
Primitivas de Requisição / Resposta

- Mensagens têm identificadores exclusivos:
  - *requestID* gerado por sequência numérica no processo remetente;
  - Identificador do processo remetente (porta + IP).
- Falhas de comunicação
  - Na transmissão, *time-out* X Repetição da mensagem & descarte;
  - Na recepção, o protocolo muda em função da operação ser ou não idempotente (histórico ou nova execução?)

Serviços de Nome

- Nomes "puros", endereços e outros atributos
- Resolução de Nomes: nome > atributos
- “Serviços de Nome (DNS)”: pesquisa hierárquica
- URI (Uniform Resource Identifiers) e URL (Uniform Resource Locators)

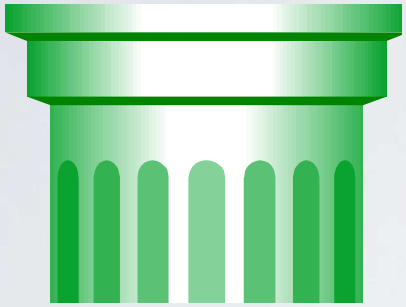
Composição do Nome de Domínio

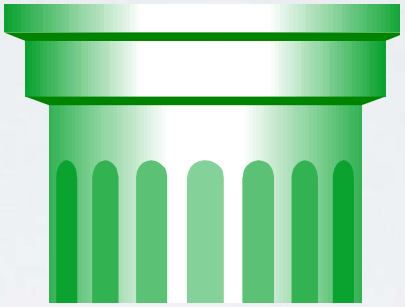


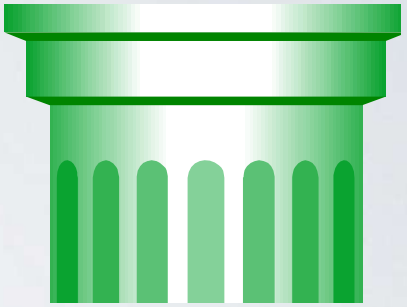
Nomes de Domínio

- Domain Name System
  - RFC 1034 e 1035 (1987)

SEGURANÇA - 3 PILARES







- Integridade
  - Exatidão
  - Completude
- Confidencialidade
  - Acesso apenas para pessoas autorizadas
- Disponibilidade
  - Os dados estarão lá quando forem necessários

OUTROS PARÂMETROS

- Autenticidade
  - Certeza de que um objeto (em análise) provém das fontes anunciadas e que não foi alvo de mutações ao longo de um processo;
- Não repúdio
  - Também chamada de irretratabilidade, define a propriedade pela qual o emissor não pode negar a autenticidade de uma mensagem.

PONTOS DE APLICAÇÃO

- Usuário
  - Confidencialidade?
- Tráfego
  - Integridade
  - Confidencialidade
- Serviço
  - Disponibilidade

SEGURANÇA EM CAMADAS
DEFENSE IN DEPTH

- Princípio dos Castelos Medievais: primeiro o descampado expõe o atacante saindo da floresta; depois, o fosso, a ponte levadiça, a muralha, e, mesmo dentro, vielas para dificultar o acesso à edificação principal. Por fim, uma torre onde se entrincheirar se tudo desse errado;
- Em TI:
    - *Firewall*;
    - IDS (*Intrusion Prevention System*);
    - *Gateway*;
    - Servidores;
    - Estações de Trabalho;
    - Treinamento.

TI: CONTROLES APLICÁVEIS

- *Firewall*
- Controle de Conteúdo
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)
- Antivírus
- Criptografia
- Autenticação (Forte?)

TI: CONTROLES APLICÁVEIS

- *Firewall*
  - Fica na fronteira entre duas redes;
  - Controla o tráfego entre elas;
  - Política de Lista "branca" e "negra";
  - Pode tratar de forma diferenciada uma rede específica (DMZ).
- Controle de Conteúdo
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)
- Antivírus

TI: CONTROLES APLICÁVEIS

- *Firewall*
- Controle de Conteúdo
  - Determina o acesso ou bloqueio da WEB em função do conteúdo;
  - Pode estar associado a horário, usuário, computador etc;
  - Conteúdos podem ser identificados de diversas formas:
    - Lista "branca" ou "negra" de URLs;
    - Seqüências de texto;
    - Categorias.
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)

TI: CONTROLES APLICÁVEIS

- *Firewall*
- Controle de Conteúdo
- Detecção de Intrusão (IDS)
  - **Ferramenta tipicamente passiva;**
  - **Monitora e analisa eventos, registrando LOG e emitindo alertas;**
  - **Normalmente fica distribuído na rede em diversos pontos.**
- Prevenção de Intrusão (IPS)
- Antivírus
- Criptografia

TI: CONTROLES APLICÁVEIS

- *Firewall*
- Controle de Conteúdo
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)
  - **Ferramenta ativa;**
  - **Detecta eventos e toma ações visando proteger a rede.**
- Antivírus
- Criptografia
- Autenticação (Forte?)

TI: CONTROLES APLICÁVEIS

- *Firewall*
- Controle de Conteúdo
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)
- Antivírus
  - Identifica padrões de códigos maliciosos, bloqueando seus efeitos;
  - Pode ser instalado em 3 pontos: estação, servidor e *gateway*.
- Criptografia
- Autenticação (Forte?)

TI: CONTROLES APLICÁVEIS

- *Firewall*
- Controle de Conteúdo
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)
- Antivírus
- Criptografia
  - **Ferramenta típica para controle de integridade e confidencialidade;**
  - **Usada como tecnologia padrão para VPNs.**
- Autenticação (Forte?)

TI: CONTROLES APLICÁVEIS

- Controle de Conteúdo
- Detecção de Intrusão (IDS)
- Prevenção de Intrusão (IPS)
- Antivírus
- Criptografia
- Autenticação (Forte?)
  - Garante a autenticidade;
  - É considerada "forte" quando envolve mais de um dos elementos:

    O QUE VOCÊ ... É? SABE? TEM?

CRIPTOGRAFIA

- Histórico
- Cifras de Substituição
- Cifras de Transposição
- Cifra de Chave Única (inquebrável)
- Princípios Fundamentais
- Criptografia Simétrica e Assimétrica
- Chave Pública e Privada
- Algoritmo RSA

CIFRAS DE SUBSTITUIÇÃO

- Cifra de César (pula 3 letras);
- Deslocamento Genérico
- Substituição Monoalfabética

CIFRAS DE TRANSPOSIÇÃO

- Escolher palavra sem letras repetidas como chave
- Distribuir letras do texto horizontalmente, desprezando os espaços em branco;
- Transmitir por colunas, seguindo a ordem alfabética das letras da chave
- No destino, seguir o processo inverso

CIFRA DE CHAVE ÚNICA

- Operação OU-EXCLUSIVO do conteúdo a ser criptografado com sequência pseudo-aleatória conhecida por ambos os lados;
- No destino, repetir o processo para decriptografar

SIMÉTRICA OU ASSIMÉTRICA

- Criptografia Simétrica
  - Emissor e Receptor compartilham a mesma chave, que precisa ser encaminhada em segredo;
- Criptografia Assimétrica
  - Chaves do emissor e receptor são diferentes, e não precisam ser trocadas.

CHAVE PÚBLICA E PRIVADA

- Bob e Alice: figuras típicas
- Para Bob enviar uma mensagem para Alice
  - Bob solicita a chave pública de Alice;
  - Alice fornece a chave pública;
  - Bob utiliza esta chave para criptografar o conteúdo;
  - Alice recebe a mensagem criptografada, e com sua chave privada, é capaz de decodificar a mensagem.

CRIPTOGRAFIA RSA

- Origem da sigla:
  - Rivest
  - Shamir
  - Adleman
- Criado em 1978, e até hoje não foi quebrada

CRIPTOGRAFIA RSA

- Dificuldade: fatorar o produto de dois números primos grandes;
- Passos:

1o.) Escolher 2 números primos grandes "p" e "q";

2o.) Calcular "n" e "z" da seguinte forma:

$n = p \cdot q$

$z = (p - 1) \cdot (q - 1)$

3o.) Escolha um número "d" tal que "d" e "z" sejam primos entre si;

4o.) Calcule "e", tal que:

$(e \cdot d)$ **mod**[*1] $z = 1$

*1 A operação "**mod**" retorna o resto da divisão entre seus operadores. Ex: 10 **mod** 4 = 2.

CRIPTOGRAFIA RSA

- De posse dos valores calculados, divida a mensagem em trechos "T" com valor binário inferior a "n". Para criptografar, transforme "T" em "C":

  $C = T^{\mathbf{e}} \bmod \mathbf{n}$, onde "e" e "n" formam a chave pública

- Para decriptografar, transforme "C" em "T":

  $T = C^{\mathbf{d}} \bmod \mathbf{n}$, onde "d" e "n" formam a chave privada

O PROBLEMA DA CONFIABILIDADE DAS CHAVES

- Um risco da criptografia por chave pública-privada é a possível interceptação da mensagem de Bob solicitando a chave pública de Alice;
- Um invasor poderia responder no lugar de Alice, receber a mensagem, e depois redirecioná-la para Alice;
- Para resolver, é importante associar a chave pública ao seu proprietário -> é aí que entram os certificados digitais
    - As entidades certificadoras (CA - Certification Authorities) associam o emissor à sua respectiva chave pública, e adicionam um conteúdo criptografado no certificado com a chave privada da CA, que pode ser checada pelo emissor antes do envio da mensagem.

INFRAESTRUTURA DE CHAVE PÚBLICA

- A infraestrutura de chave pública (PKI - *public-key infrastructure*) permite criar, gerenciar, distribuir, usar, armazenar e revogar certificados digitais;
- A PKI associa as chaves públicas com as identidades de usuário inequívocas através de uma autoridade certificadora (CA). A garantia de que esta associação está correta é dada pela Autoridade de Registro (RA), de forma que fica garantido o não-repúdio.

CONTEÚDO DE UM CERTIFICADO DIGITAL

- **Serial Number**: Identificação do certificado.
- **Subject**: Identificação da pessoa física ou jurídica.
- **Signature Algorithm**: Algoritmo usado para criação da assinatura.
- **Signature**: Assinatura do Emissor.
- **Issuer**: Órgão Emissor.
- **Valid-From**: Data de emissão.
- **Valid-To**: Data de Validade.
- **Key-Usage**: Objetivo da Chave (Ex. encriptação, assinatura etc).
- **Public Key**: Chave pública.

Trabalho Opcional

Soluções de mercado para segurança em Sistemas Distribuídos

- Análise de 2 soluções, uma de pequeno porte, e outra para empresas médias/grandes;
- Foco nos UTMs (Unified Threat Management)
- Fabricantes: SonicWall, CISCO, Juniper, Fortinet

Equipes de 2 a 4 alunos

- Escolha do fabricante e formação das equipes até a próxima aula

Aula 05

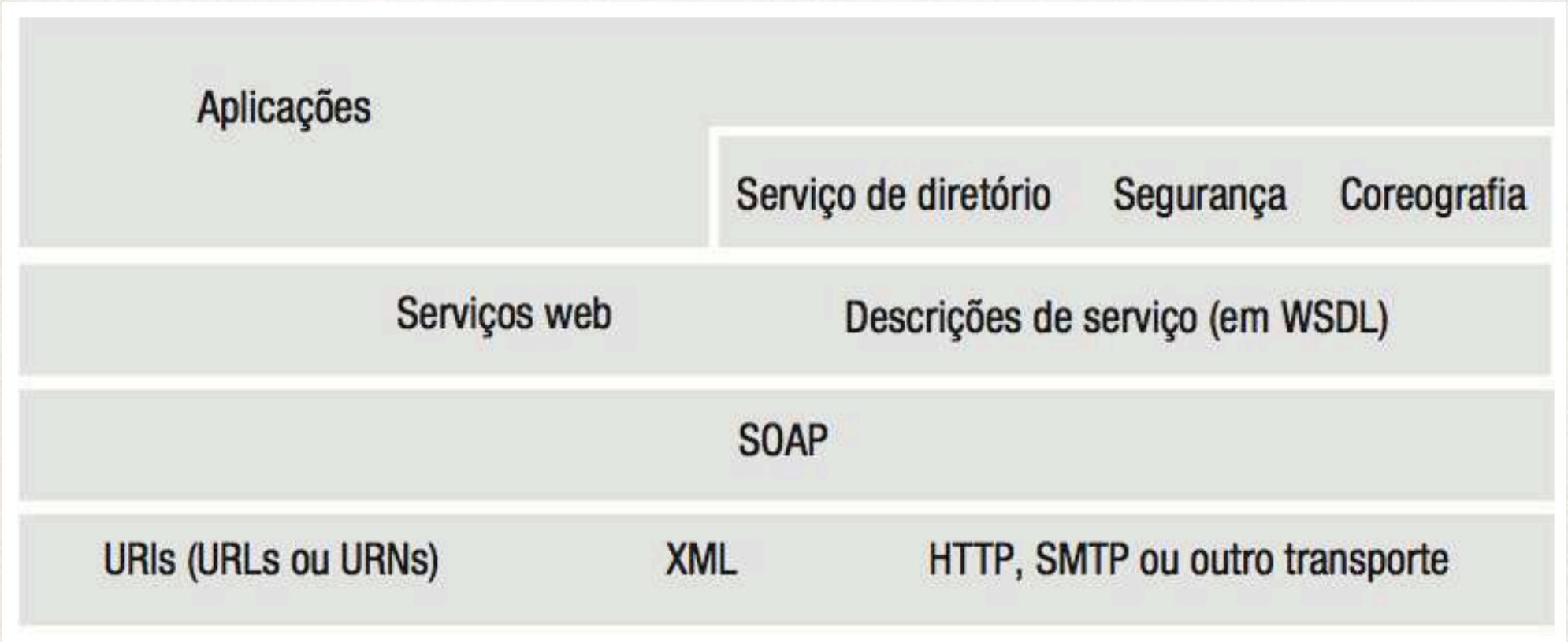
V.1.11 - 2021.10

Serviços Web

- Interface de serviço mais genérica do que simples navegadores
  - Tipicamente são ofertados por aplicações específicas, mesmo que às vezes essas possuam interfaces Web;
  - Utilizam requisições HTTP, com mensagens de requisição/resposta empacotadas em XML via SOAP;
  - Permitem B2B e *mashup* (*softwares* que misturam serviços já existentes com interfaces inovadoras);
  - Computação de grade (*grid*) e em nuvem.
- Serviço WEB ≠ Servidor WEB
- Retorno ao modelo cliente-servidor, com serviços especializados?

HTTP: *Hyper Text Transfer Protocol*
XML: *eXtensible Markup Language*
SOAP: *Simple Object Access Protocol*
B2B: *Business "To" Business*

Infraestrutura e Componentes dos Serviços Web



Fonte: Figura 9.1 de Sistemas Distribuídos - Conceitos e Projeto - 5 edição - George Couloris

Transações Distribuídas

V.1.11 - 2021.10

Uma transação …

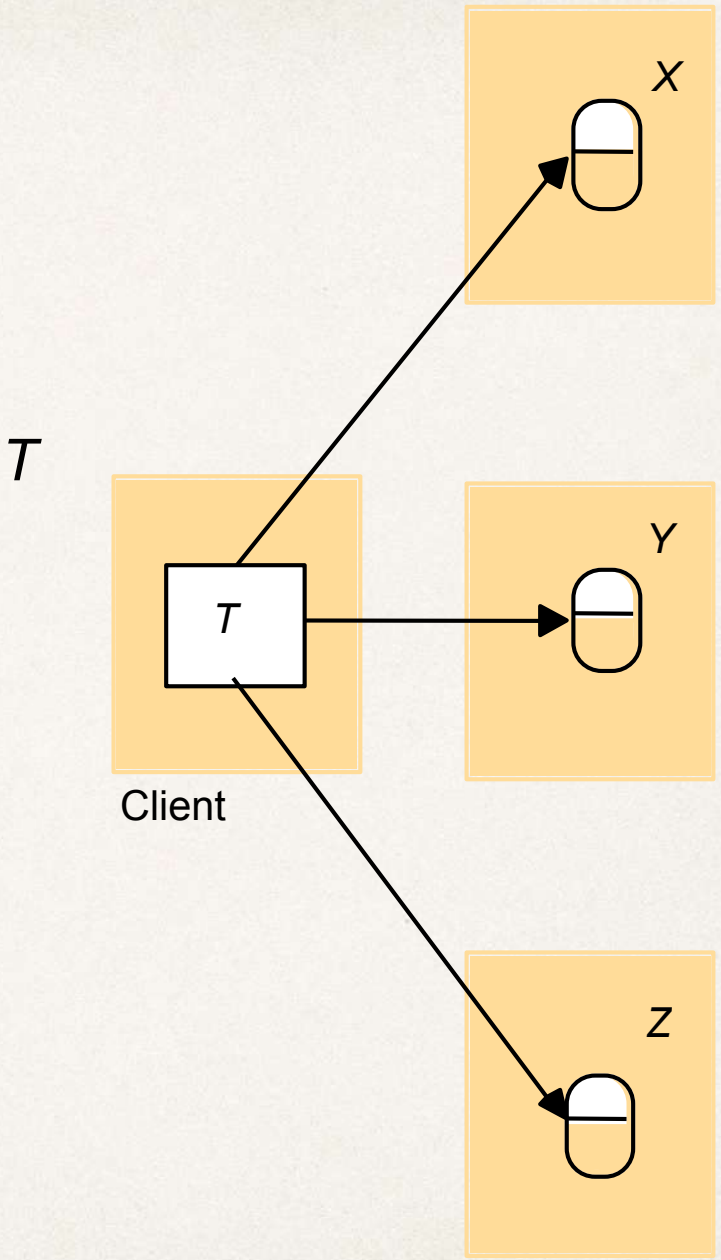
- Define uma sequência de operações que serão executadas pelos dispositivos componentes do SD;
- Deve garantir integridade e consistência dos dados;
- Opera de forma sequencial, mesmo que de forma concorrente com outros processos.

Uma transação em um SD …

- Determina operações em diferentes dispositivos;
- Determina que um dos dispositivos (servidor) assumirá o papel de coordenador;
  - Outros dispositivos são chamados de participantes;
  - O protocolo, através da comunicação entre os dispositivos, permitirá decidir se uma transação pode ser efetivada ou será cancelada;
- Pode ser plana ou aninhada.

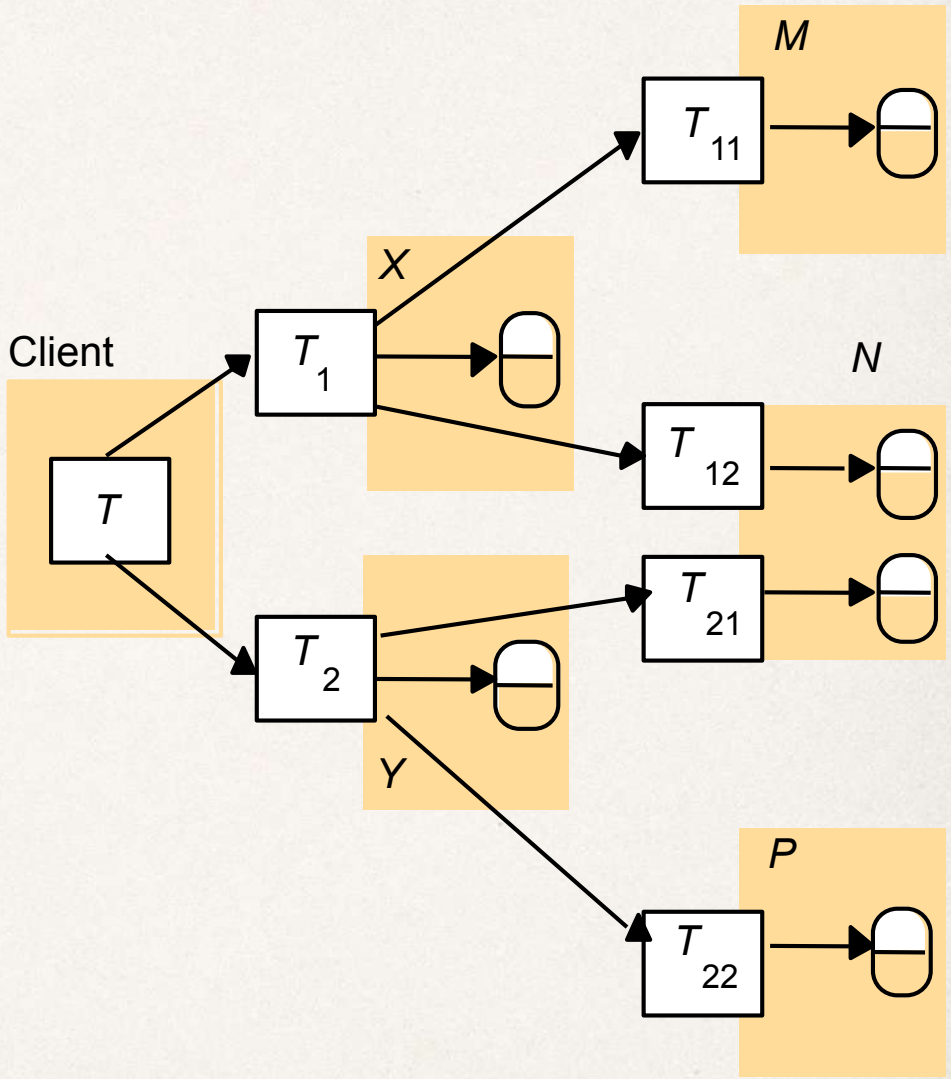
Transação Plana

- Todo o trabalho é feito no mesmo nível entre o início e o término de uma transação;
- A transação acessa os objetos dos servidores em sequência;
- Não é possível efetivar ou cancelar partes dela.



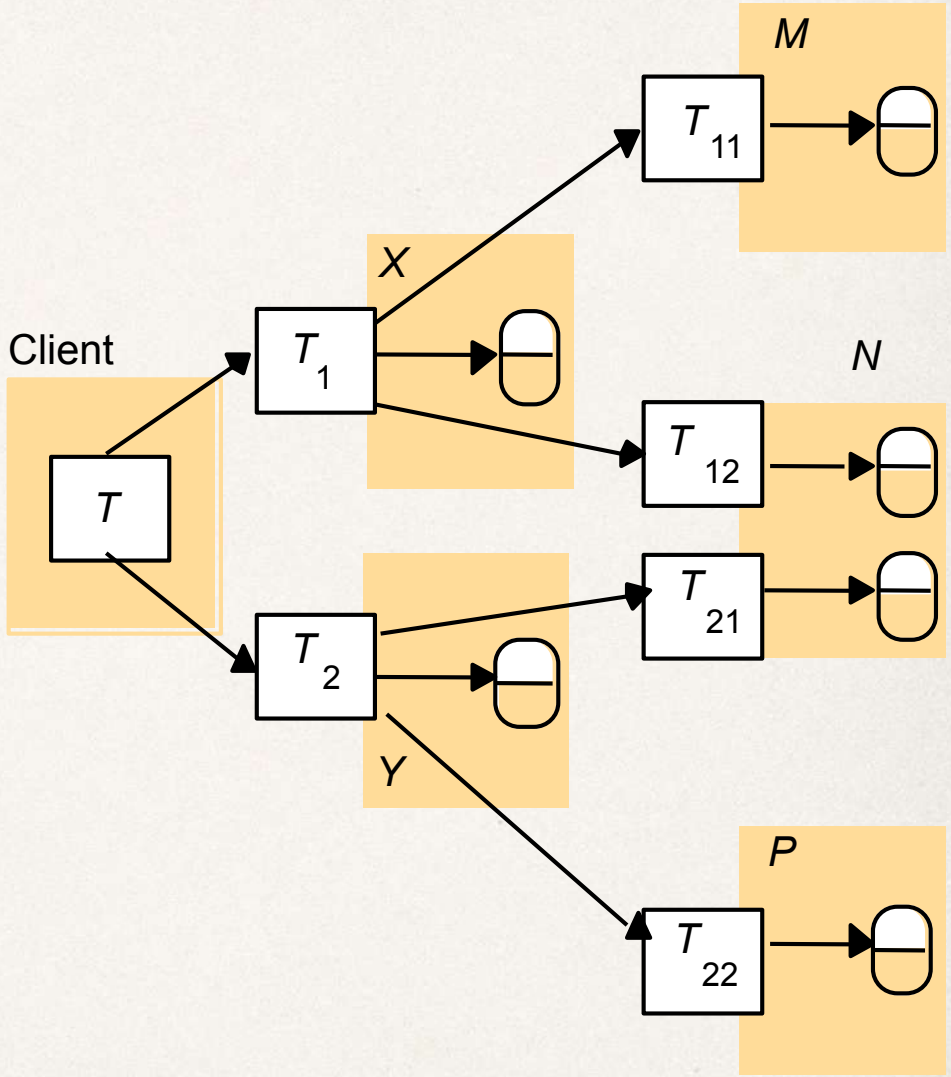
Transação Aninhada

- A transação de nível superior pode abrir subtransações e assim por diante;
- Subtransações no mesmo nível podem ser executadas de modo concorrente;
- Subtransações podem ser efetivadas ou canceladas independentemente
  - Nas transações planas, uma falha causaria o reinício da transação inteira.
- Uma transação só pode ser efetivada ou cancelada depois que suas descendentes tiverem sido concluídas (com sucesso ou não).



Transação Aninhada

- Quando uma transação ascendente é cancelada, todas as suas subtransações são canceladas mesmo que tenham sido efetivadas provisoriamente.
  - Por exemplo, se T2 é cancelada, então T21 e T211 também devem ser canceladas, mesmo que possam ter sido efetivadas provisoriamente.
- Quando uma subtransação é cancelada, a transação ascendente pode decidir se vai ser cancelada ou não.
  - No exemplo, T decide ser efetivada, embora T2 tenha sido cancelada.



Transações: Propriedades

- **A**tomicidade
- **C**onsistência
- **I**solamento
- **D**urabilidade

Atomicidade

- Toda a transação, mesmo que envolva diversos passos, ou subtransações múltiplas, deve ser vista como uma ação completa e única;
- Se alguma operação não for concluída com êxito, todos os resultados obtidos na execução, até o momento da falha, devem ser desprezados;
- Exemplos:
  - Saque de dinheiro no caixa eletrônico;
  - Compra de uma passagem aérea.

Consistência

- É importante garantir a consistência dos dados tanto após a confirmação, quanto se a transação for abortada;

- Preservar a consistência significa que, se uma transação iniciar em um estado do sistema consistente, quando ela terminar, por qualquer motivo, inclusive por falha, o estado continuará consistente.

Isolamento

- Cada transação deve ser executada sem interferência de outras transações;
- Resultados intermediários não devem ser visíveis por outras transações;
- Transações executadas de forma concorrente devem provocar resultados similares à execução sequencial das mesmas transações.

Durabilidade

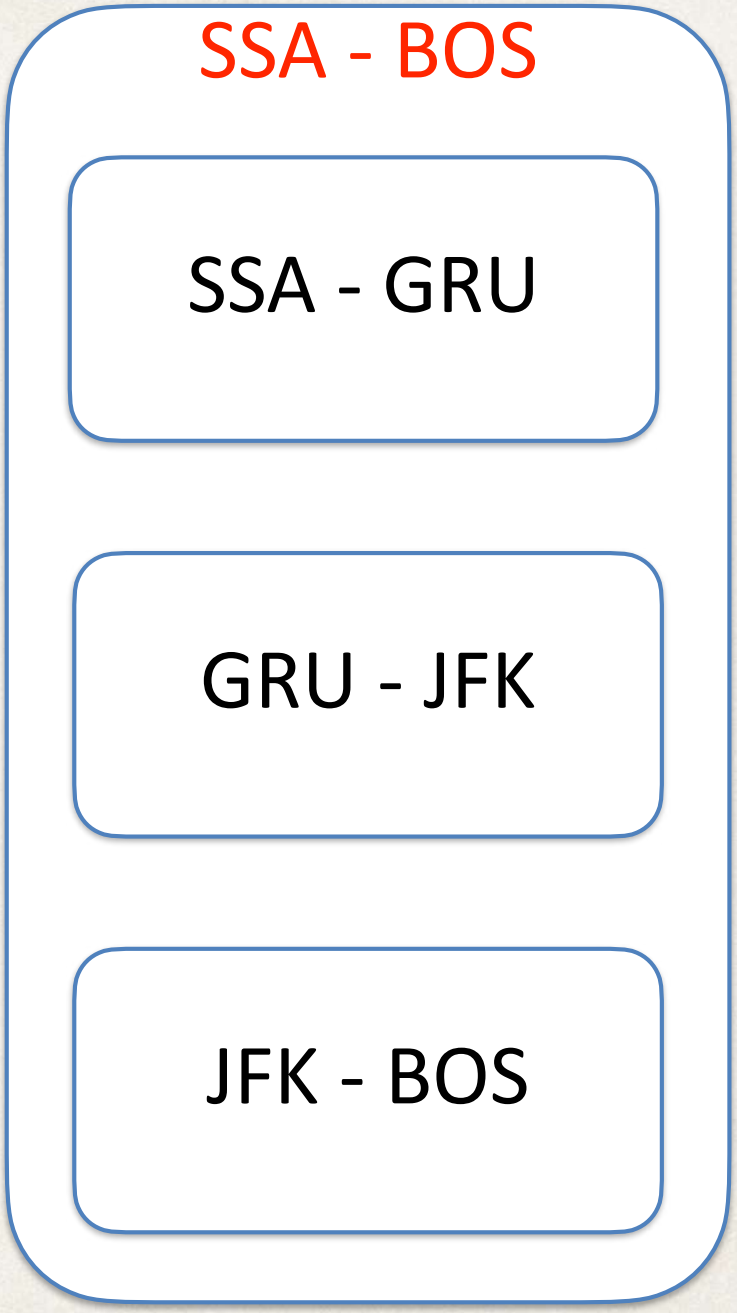
- O efeito de uma transação que termina com sucesso deve ser persistente, ou seja:
  - Todos os dados, mesmo em transações pendentes, devem ser armazenados em meio de armazenamento permanente, de modo que possa ser recuperado em caso de falha do dispositivo.

Programando Transações Atômicas

- *Begin_transaction*: marca o início de uma transação;
- *End_transaction*: marca o final de uma transação;
- *Abort_transaction*: termina a transação recuperando os valores anteriores ao *Begin_transaction*;
- *Read*: leitura realizada dentro de uma transação;
- *Write*: gravação realizada dentro de uma transação.

Programando Transações Atômicas

- Passagem aérea para um curso de verão em Harvard (passagem aérea SSA -> BOS)
  - Subtransações intermediárias;
  - Mesmo que confirmadas, precisam poder ser canceladas.
- A transação acessa os objetos dos servidores em sequência;
- Não é possível efetivar ou cancelar partes dela.



Programando Transações Atômicas

- Durabilidade;
- Arquivo com o "LOG" das operações até o *End_Transaction*
  - O arquivo de LOG também deve ser durável !
- Isolamento.

Programando Transações Atômicas

- Mantendo o LOG
  - *Begin_transaction*: grava o “início" no LOG;
  - *Write*: grava os dados atualizados no LOG;
  - *Abort_transaction*: grava o “abortado" no LOG;
  - *End_transaction (commit)*: grava “commit” no LOG, copia dados do LOG para os arquivos e grava “Completo" no LOG.
- Recuperação de Desastre -> leitura do LOG
  - Tem “início”? Verifica o resto;
  - Tem “Completo”? Não é necessário fazer nada;
  - Tem “abortado”? Desfazer alterações registradas no LOG;
  - Tem “commit”, mas não tem “Completo”? Copia dados do LOG para os arquivos e grava “Completo" no LOG.

Transações Atômicas Distribuídas

- Múltiplos dispositivos e localidades;
- Conceitos são similares, porém há diferenças no tratamento de falhas;
- Modos de Falha
  - Falha ou parada de um dispositivo ou localidade;
  - Falha de rede ou disco;
  - Falhas bizantinas (*).

Falhas Bizantinas

- Um grupo de generais, cada qual comandando uma parte do exército bizantino, precisa coordenar um ataque a um castelo;
- Cada general precisa decidir se ataca, ou não, o castelo;
- A falta de unanimidade pode provocar a morte de muitos soldados;
- Alguns generais votam seletivamente, a depender do grupo que o questiona;
- Separados fisicamente, existe também a questão dos mensageiros, que podem falhar na entrega das mensagens, ou alterar os votos;
- Pode-se definir um voto padrão para o caso da ausência de respostas ser a maioria (p.ex. Desistir do Ataque).

V.1.10 - 2020.11

Aula 06

V.1.11 - 2021.10

Protocolo de Efetivação em 2 Fases

Two-Phase Commit

- O objetivo é que todos os participantes de uma transação distribuída possam confirmar ou abortar a transação de forma unânime;
- É eleito um dispositivo coordenador (algoritmos de eleição);
- O dispositivo coordenador monta uma lista dos participantes. Por sua vez, cada dispositivo tem a identificação do coordenador;
- Ao abrir uma transação o coordenador associa à mesma um identificador da transação (TID);

Protocolo de Efetivação em 2 Fases

Two-Phase Commit

- Fase 1 - dispositivo coordenador:
  - Registra "*Prepare (TID)*" no LOG em meio durável;
  - Envia "*Vote_Request (TID)*" para todos os participantes;
  - Espera retorno.
- Fase 2 - dispositivo coordenador:
  - Se qualquer dispositivo responder "*Abort (TID)*", grava "*Abort (TID)*" no LOG durável, envia "*Global_Abort (TID)*" para todos os participantes, e aborta localmente a transação TID;
  - Se **todos os dispositivos** responderem"*Ready_to_commit (TID)*", grava "*Commit (TID)*" no LOG durável, envia "*Global_Commit (TID)*" para todos os participantes, e efetiva localmente a transação TID.

Protocolo de Efetivação em 2 Fases

Two-Phase Commit

- Fase 1 - dispositivo participante:
  - Recebe "*Vote_Request (TID)*" do coordenador;
  - Grava "*Abort (TID)*" ou "*Ready (TID)*" no LOG durável local;
  - Envia "*Abort (TID)*" ou "*Ready_to_commit (TID)*" para o coordenador;
  - Se decidiu abortar, aborta a transação localmente.
- Fase 2 - dispositivo participante:
  - Aguarda "*Global_Abort (TID)*" ou "*Global_Commit (TID)*" do coordenador;
  - Grava "*Abort (TID)*" ou "*Commit (TID)*" no LOG durável local;
  - Aborta ou efetiva a transação localmente a depender da mensagem.

Recuperação de Falhas p/ *2-Phase Commit*

- Falhas possíveis no Coordenador
  - Falha local no coordenador;
  - Estado de espera provocado pela ausência de resposta de pelo menos um dos participantes.
- Falhas possíveis no dispositivo participante
  - Falha local no dispositivo;
  - Estado de espera provocado pela ausência de resposta *"Global_Abort (TID)"* ou *"Global_Commit (TID)"* do coordenador.

Recuperação de Falhas p/ *2-Phase Commit*

- Diante da falta de resposta de um ou mais dispositivos ao "*Vote_Request (TID)*" do coordenador
  - Falha local do dispositivo;
  - Falha de rede.
- *Timeout* é considerado como equivalente à "*Abort (TID)*"
  - Grava "*Abort (TID)*" no LOG durável;
  - Envia "*Global_Abort (TID)*" para todos os participantes;
  - Aborta localmente a transação TID;

Recuperação de Falhas p/ *2-Phase Commit*

- Diante de falha local no dispositivo coordenador, inspecionar o LOG;
- Se encontrar "*Abort (TID)*" ou "*Commit (TID)*"
  - Reenviar mensagem correspondente;
  - Aborta ou efetiva a transação localmente.
- Se encontrar "*Prepare (TID)*", tomar uma das duas ações abaixo:
  - Reenvia "*Vote_Request (TID)*" para todos os participantes e espera retorno;
  - Aborta a transação unilateralmente, colocando "*Abort (TID)*" no LOG durável e envia "*Global_Abort (TID)*" para todos os participantes.
- Se não encontrar nada no LOG, aborta a transação conforme descrito acima.

Recuperação de Falhas p/ *2-Phase Commit*

- Diante da falta de resposta do coordenador após o dispositivo enviar "*Ready_to_commit (TID)*":
  - Envia mensagem ao coordenador, perguntando como proceder;
  - Se não obtiver resposta, envia mensagem para outros dispositivos participantes, verificando se eles sabem qual é a resposta;
  - Se qualquer outro dispositivo participante estiver em "*Abort (TID)*" ou "*Commit (TID)*", tomar a mesma ação localmente;
  - Se não obtiver resposta, aguardar dispositivo coordenador ser reiniciado, ou seja, o dispositivo é bloqueado no modo "*Ready*".

Recuperação de Falhas p/ *2-Phase Commit*

- Diante de falha local no dispositivo participante, inspecionar o LOG;
- Se encontrar "*Commit (TID)*"
  - Repetir a transação;
- Se encontrar "*Abort (TID)*"
  - Abortar a transação;
- Se não encontrar nenhum registro relacionado a TID
  - "*local_abort (TID)*"
- Se encontrar "*Ready (TID)*"
  - Seguir procedimento visto no slide anterior.

Sistema de Arquivos Distribuídos

Um sistema de arquivos distribuídos permite aos programas armazenarem e acessarem arquivos remotos exatamente como se fossem locais, possibilitando que os usuários acessem arquivos a partir de qualquer dispositivo computacional em uma rede [COULOURIS]

V.1.11 - 2021.10

Dados + “Atributos” (metadados)

Tamanho do Arquivo
Horário de Criação
Horário de Acesso (Leitura)
Horário de Modificação (Escrita)
Horário de Alteração de Atributo
Contagem de Referência
Proprietário
Tipo de Arquivo
Lista de Controle de Acesso

Sistemas de Arquivos Distribuídos

- Devem permitir o armazenamento e acesso a arquivos remotos exatamente como se fossem locais;
- Devem permitir que vários processos compartilhem dados por longos períodos, de modo seguro e confiável;
- O desempenho e segurança no acesso aos arquivos armazenados em um dispositivo remoto devem ser compatíveis aqueles observados no acesso aos arquivos armazenados em discos locais.

Tipos de SA Distribuídos

Statefull

Mantém informações de estado, como o nome do arquivo, o ponteiro para o arquivo, e a posição atual;

Reduzem o tráfego e o tamanho das mensagens.

Estateles

Não mantém informações, o cliente deve encaminhar todas elas a cada acesso;

Simplificam a implementação da Tolerância a Falhas.

Requisitos de um SA distribuído

Transparência

Acesso: clientes não devem precisar saber da distribuição dos arquivos;

Localização: espaço de nome de arquivos uniforme, já que a transferência de arquivos não pode impactar caminho -> Sistema de Nome robusto.

Mobilidade: não deve ser necessária alteração dos processos, nem de tabelas. no lado do cliente;

Desempenho: a performance de acesso deve ser satisfatória, mesmo em situações de carga;

Escala: ampliações ou reduções não devem trazer complexidade ou alterações operacionais;

Requisitos de um SA distribuído

Atual. Conc.

A **Atualização Concorrente de Arquivos** garante que:

A alteração de um arquivo por um cliente não deve afetar a operação de outros clientes;

Isso deve ocorrer mesmo que os clientes estejam manipulando o mesmo arquivo;

Exclusão Mútua.

Requisitos de um SA distribuído

Replicação

Um arquivo pode ser representado por várias cópias de seu conteúdo em diferentes locais;

Servidores podem compartilhar a carga de entrega de arquivos;

Isso garante melhoria no suporte à tolerância a falhas.

Requisitos de um SA distribuído

Heterogen.

A **Heterogeneidade** exige que as interfaces de acesso aos serviços para clientes e servidores:

- Suportem diversos dispositivos computacionais;
- Suportem diversos Sistemas Operacionais.

Requisitos de um SA distribuído

Tol.a Falhas

A **Tolerância a Falhas** prevê que os serviços continuem operando mesmo diante de alguns tipos de falha;

- As falhas podem ocorrer tanto nos clientes, como nos processos servidores;
- Em caso de falha no servidor, o cliente não pode ficar permanentemente bloqueado;
- Garantir idempotência, com cache e identificação das mensagens.

Requisitos de um SA distribuído

Só deve haver uma versão disponível de cada arquivo no SA Distribuído:

- Cliente A lê o arquivo X em um momento
- Cliente B lê o mesmo arquivo X em outro momento ou lugar

} Mesmo resultado

Cópias locais no cliente devem ter consistências confirmadas (*timestamps* ?)

Requisitos de um SA distribuído

Segurança

Mecanismos de Controle de Acesso:

- Listas de Controle de Acesso;
- Máscara de Direitos Herdados;
- Serviços de Diretório.

Aplicação dos pilares básicos da segurança:

- Integridade, Confidencialidade e Disponibilidade.

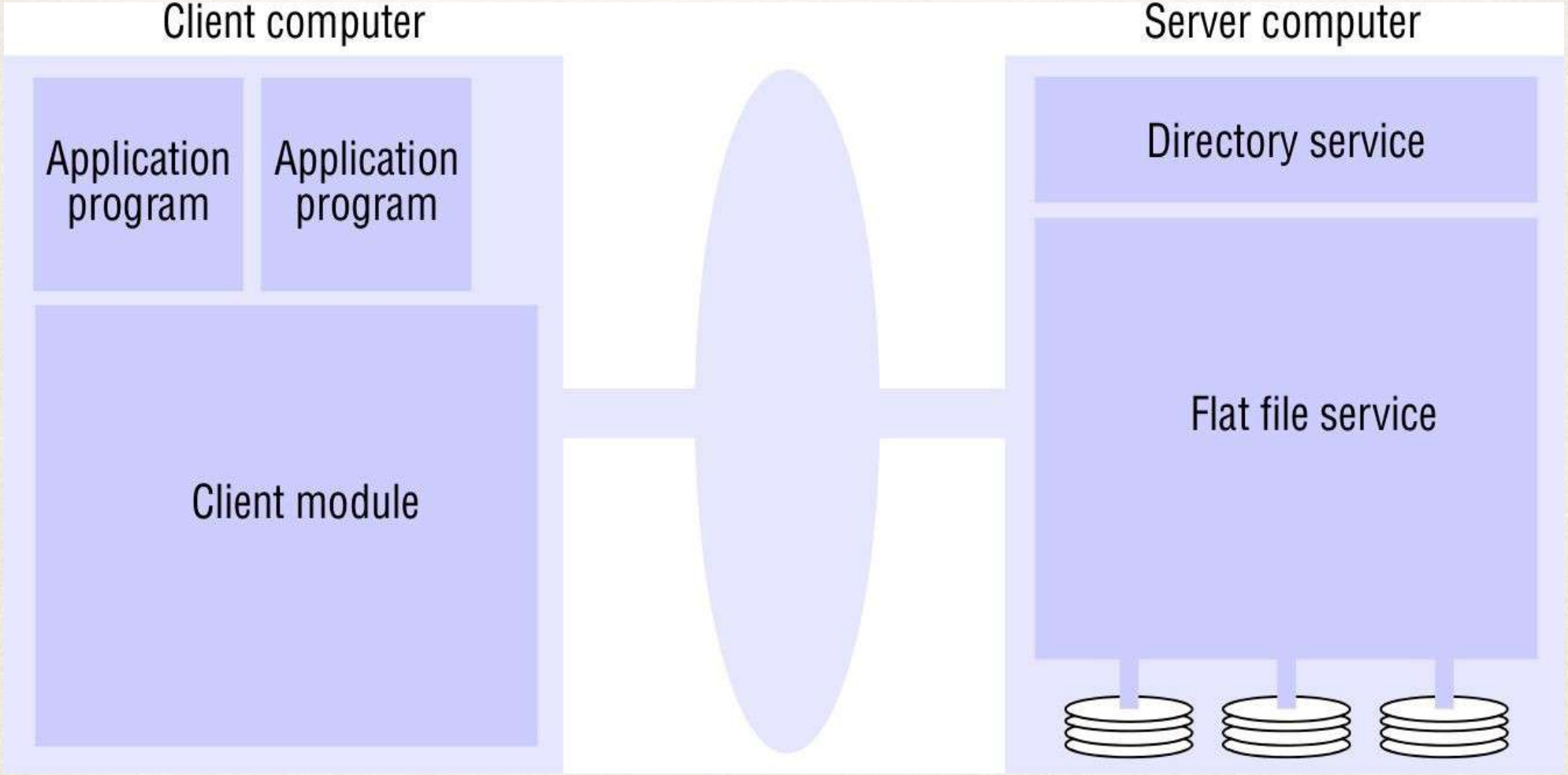
Requisitos de um SA distribuído

Um Sistema de Arquivos Distribuído deve oferecer similaridade a Sistemas de Arquivo convencionais em:

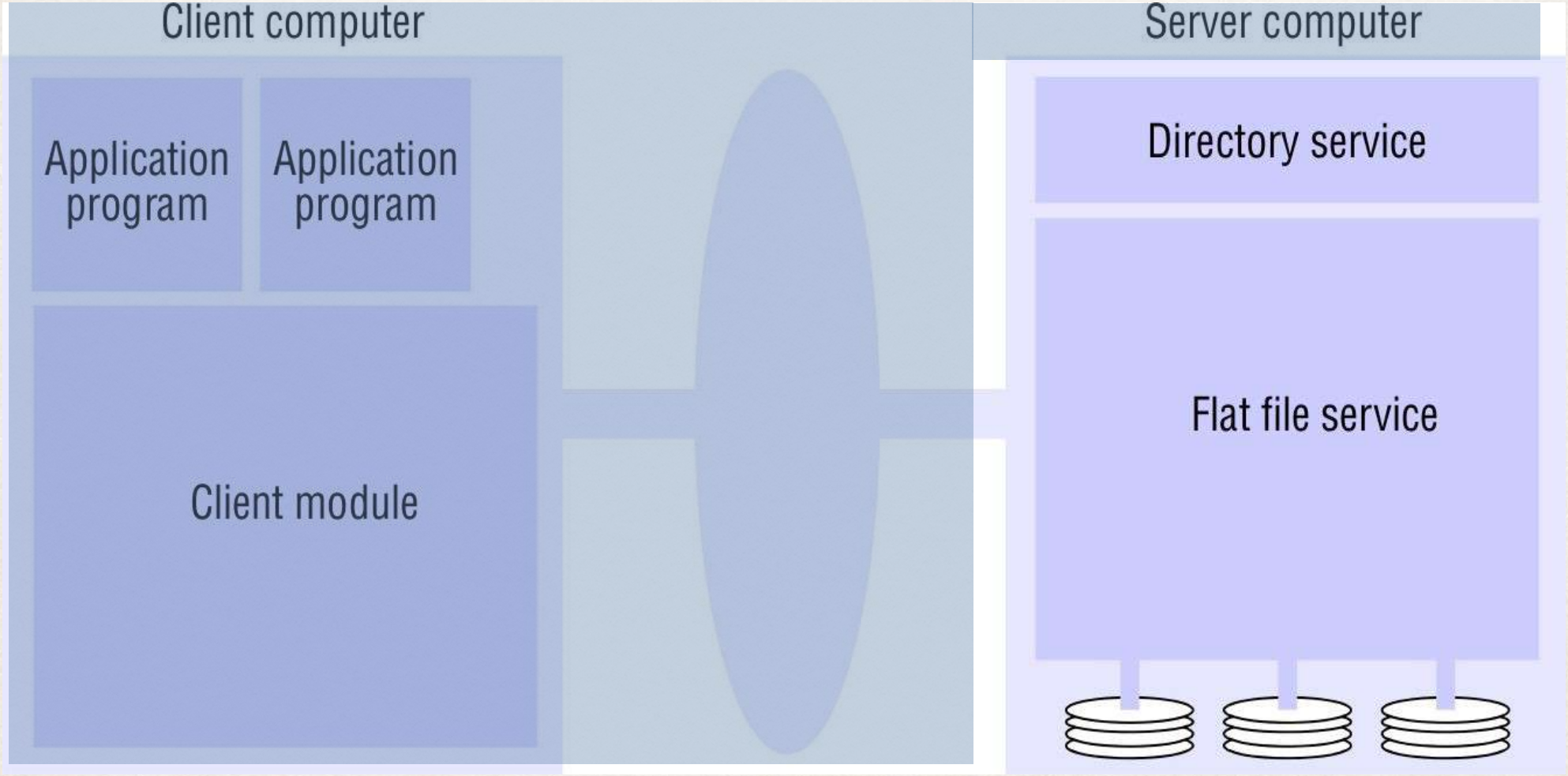
- Recursos
- Performance
- Simplicidade

Eficiência

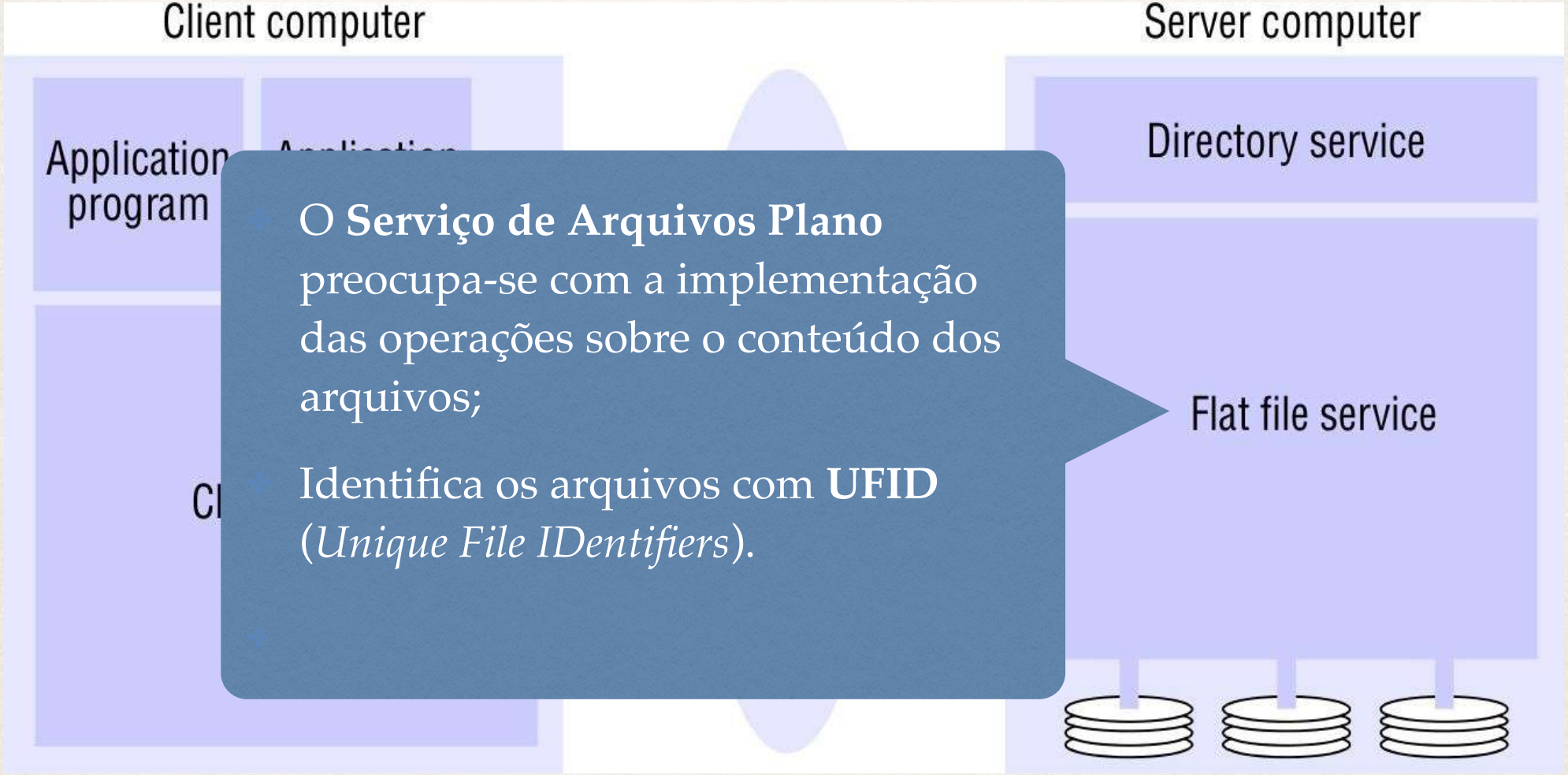
Arquitetura de um SAD



Arquitetura de um SAD

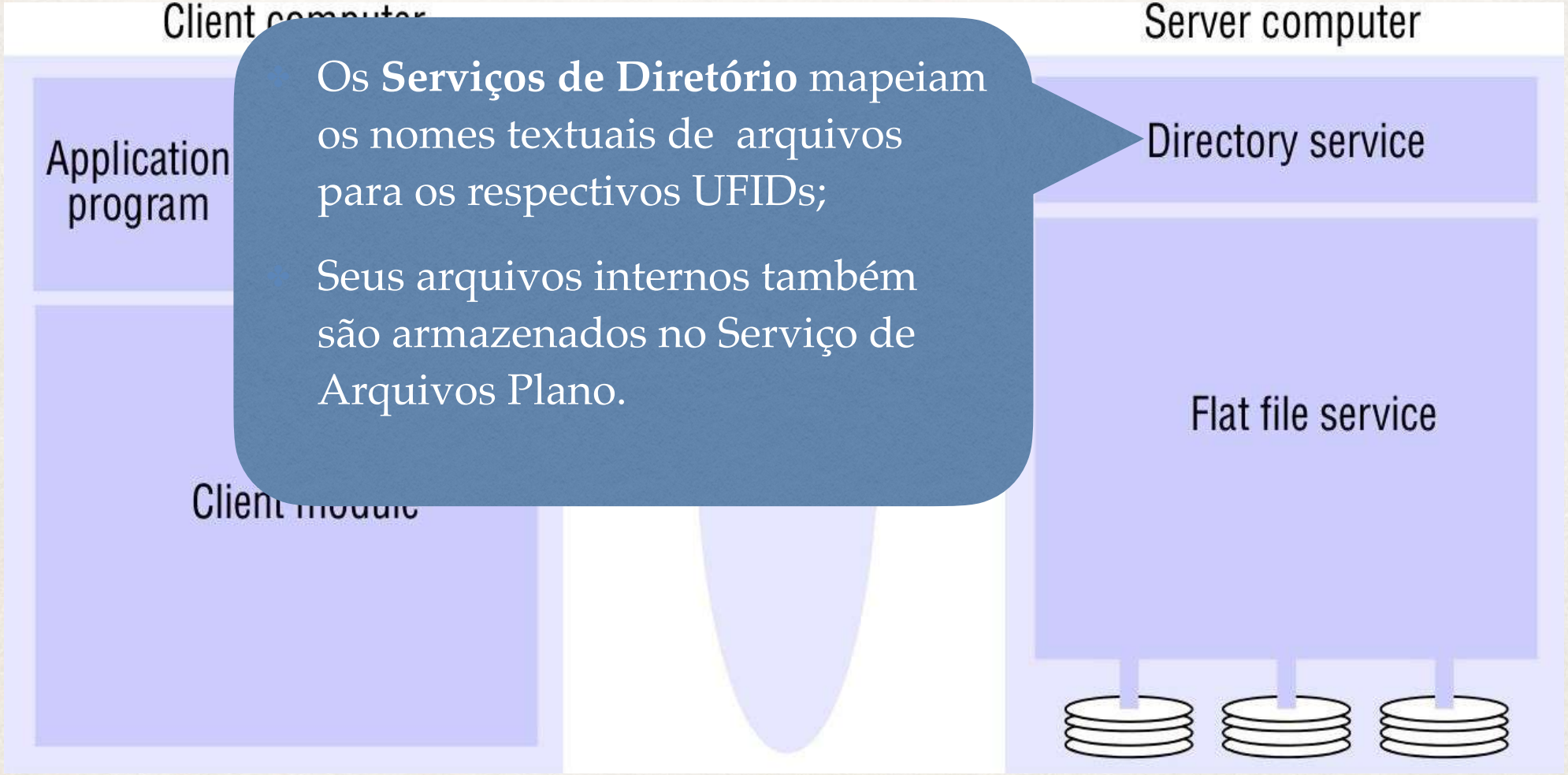


Arquitetura de um SAD



- O **Serviço de Arquivos Plano** preocupa-se com a implementação das operações sobre o conteúdo dos arquivos;
- Identifica os arquivos com **UFID** (*Unique File IDentifiers*).

Arquitetura de um SAD



- Os **Serviços de Diretório** mapeiam os nomes textuais de arquivos para os respectivos UFIDs;
- Seus arquivos internos também são armazenados no Serviço de Arquivos Plano.

Um exemplo de interface para o Serviço de Arquivos Plano

Read(FileId, i, n) → *Data* — gera *BadPosition*	Se *1* ≤ *i* ≤ *Length(File)*: lê uma sequência de até *n* elementos de um arquivo, começando no elemento *i*, e a retorna em *Data*. Gera uma exceção se o valor *i* é inválido.
Write(FileId, i, Data) — gera *BadPosition*	Se *1* ≤ *i* ≤ *Length(File)+1*: grava uma sequência de *Data* em um arquivo, começando no elemento *i*, ampliando o arquivo, se necessário. Gera uma exceção se o valor *i* é inválido.
create( ) → *FileId*	Cria um novo arquivo de tamanho zero e gera um UFID para ele.
Delete(FileId)	Remove o arquivo.
GetAttributes(FileId) → *Attr*	Retorna os atributos do arquivo.
SetAttributes(FileId, Attr)	Configura os atributos do arquivo (somente os atributos que não estão sombreados na Figura 12.3).

Figura 12.6 Operações do serviço de arquivos plano.

SAD - Desafios de Projeto

Uso eficiente do cache no cliente, garantindo desempenho igual ou superior que o dos sistemas de arquivo locais;

Manutenção da consistência entre várias cópias de arquivos de clientes colocados em cache;

Recuperação após uma falha do cliente, ou pior ainda, do servidor;

Desempenho nas operações, com escalabilidade.

Soluções para Arquivos em Rede

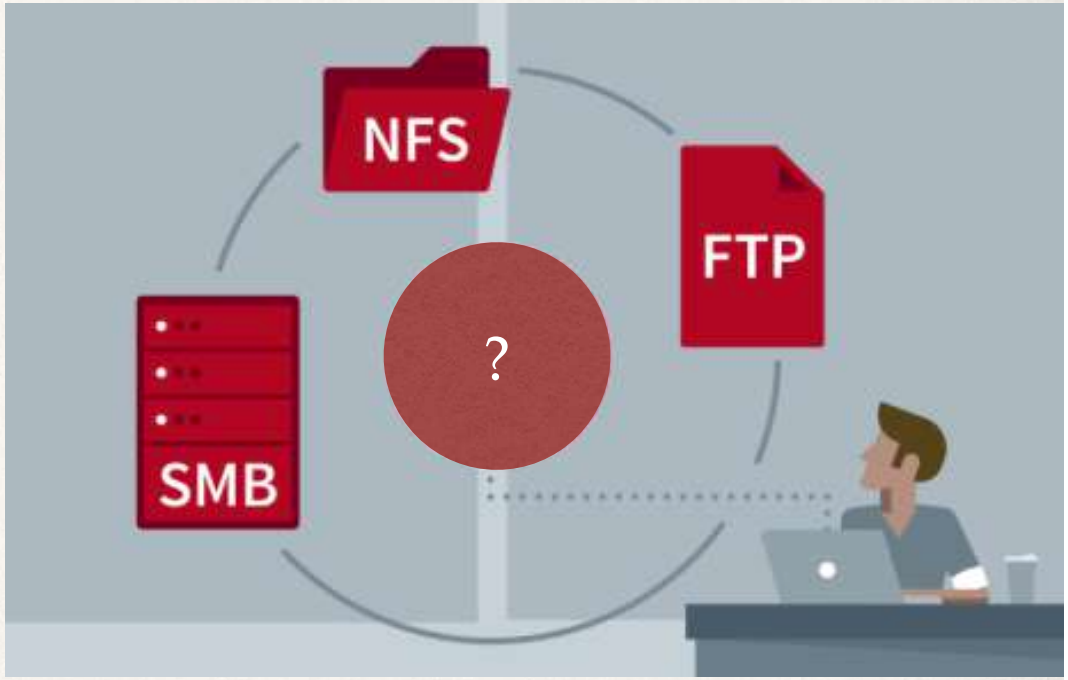
Arquivos são tipicamente ofertados por servidores em rede

- Recursos e investimentos concentrados;
- Segurança (principalmente quanto à disponibilidade) é mais fácil de implementar.

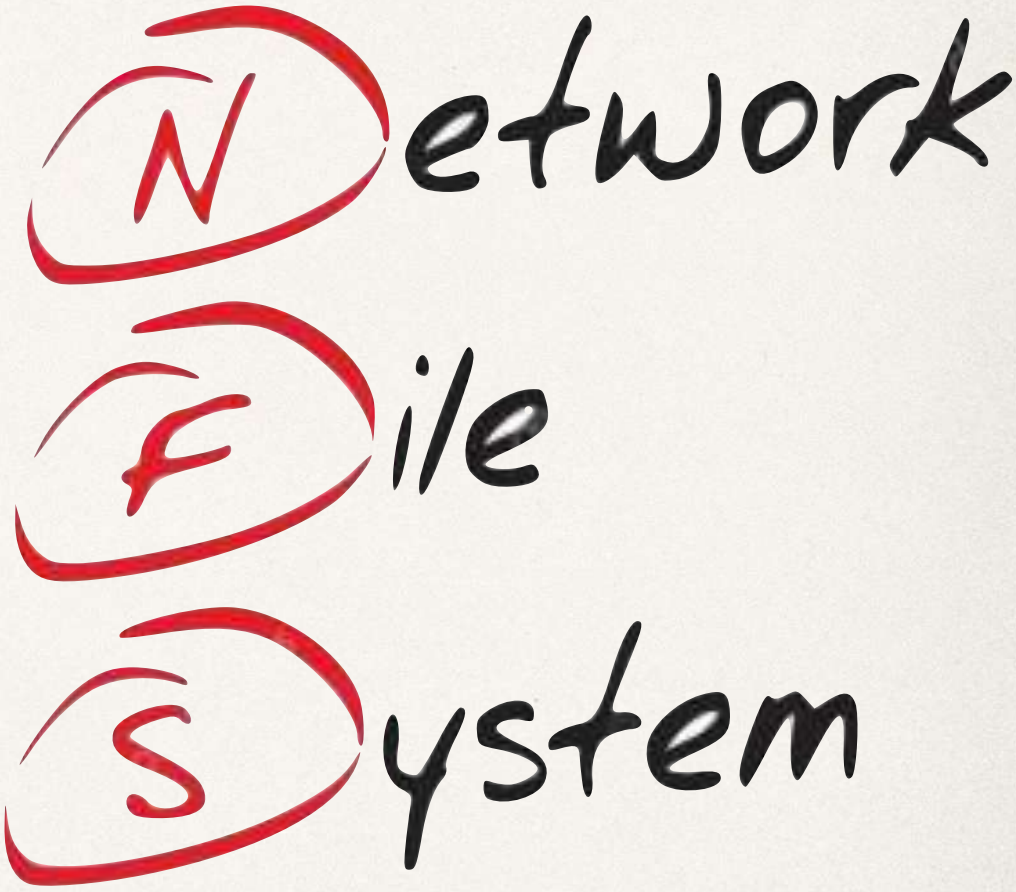
Soluções de recuperação/leitura utilizam HTTP ou FTP;

Soluções mais simples de compartilhamento muitas vezes adotam o SMB;

O NFS é a solução tipicamente preferida em Sistemas Distribuídos.



O NFS



Desenvolvido pela Sun Microsystems em 1984;

Primeiro Sistema de Arquivos desenvolvido como produto;

Permite que um usuário, em seu dispositivo cliente, acesse arquivos remotos como se fossem locais;

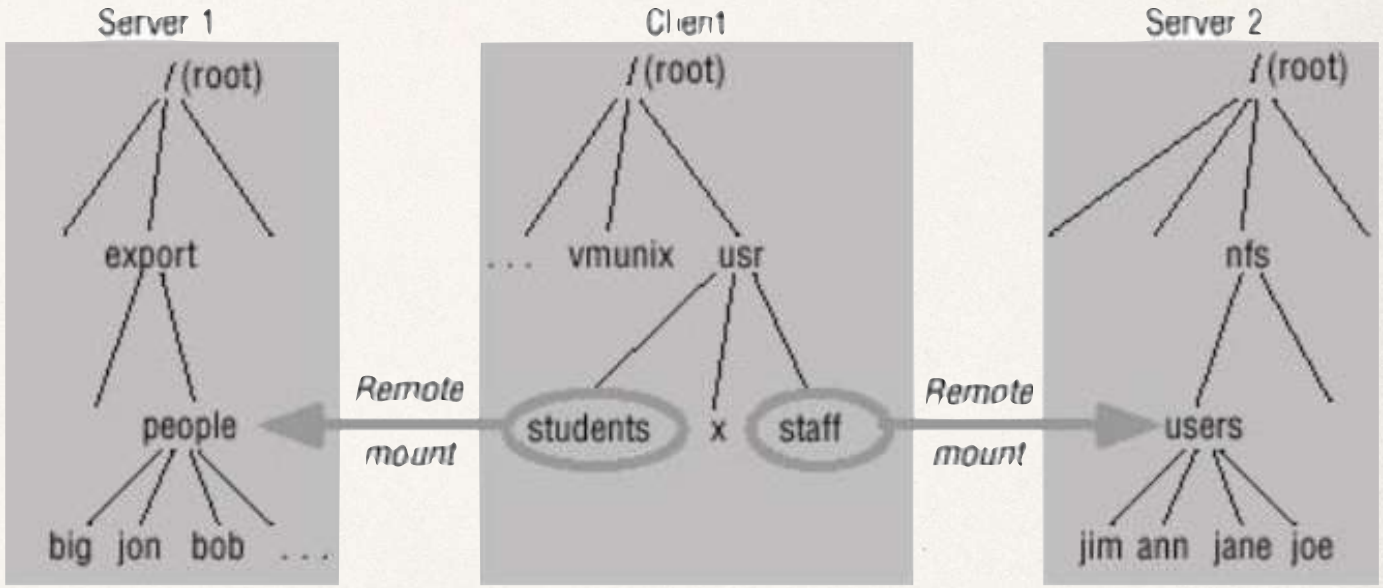
Criado para o SO UNIX, porém atualmente é independente do SO

- Existem implementações para os principais SOs de mercado;
- Independência alcançada através da utilização de RPCs (*Remote Procedure Calls*).

NFS - Operação e Protocolo

Utiliza o UDP/IP para transporte (1987);

Clientes e servidores estão na mesma rede local, tipicamente;



Cada servidor exporta um ou mais de seus diretórios, que por sua vez são “montados" nos sistemas de arquivo dos clientes, na posição desejada. Ex:

$mount server:/dir /local/dir –t nfs

NFS - Performance

Cache no Servidor

Arquivos, diretórios e atributos lidos são retidos no cache;

Se uma página é alterada, ela só será escrita em disco caso do buffer seja requisitado por outra página (*delay-write*);

Páginas alteradas são, no entanto, enviadas ao disco a cada 30s.

Cache no Cliente

Reduz o número de pedidos ao servidor (*read-ahead*);

Diferentes versões podem existir em diferentes clientes;

Validação por *timestamp*;

Clientes mantém *timestamp* da última modificação do arquivo no servidor.

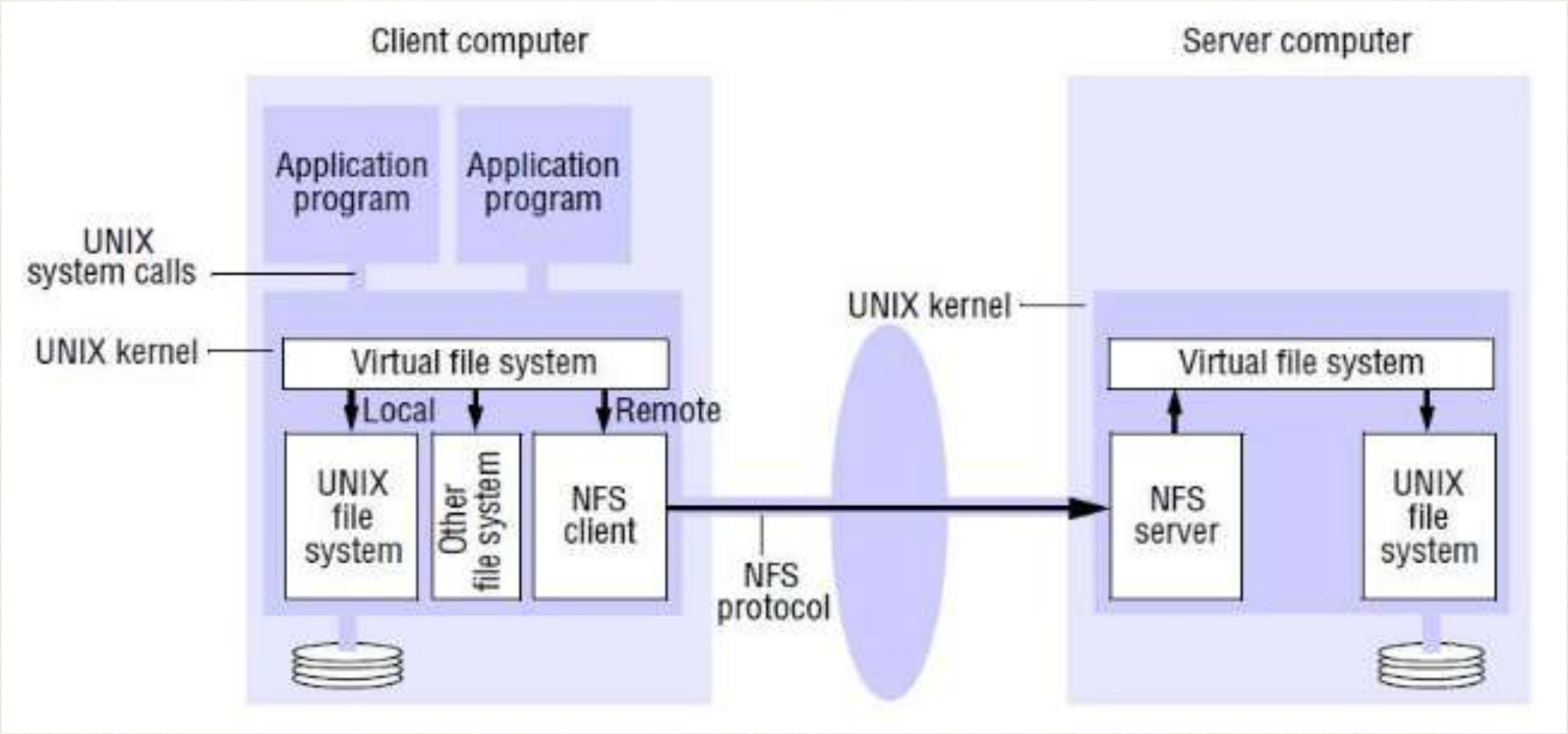
Sistema de Arquivos Virtual (VFS)

O VFS é o mecanismo pelo qual o NFS provê transparência de acesso

As operações de arquivo para sistemas locais ou remotos não têm distinção;

A distinção está entre os identificadores de arquivos utilizados pelo NFS e os locais.

NFS - Arquitetura



Outras opções: por exemplo, o SMB

Server Message Block

Desenvolvido na IBM nos anos 80;

Adotado pela Microsoft nos anos 90;

> Para garantir padronização pela IETF, foi criada uma versão chamada de CIFS (*Common Internet File System*), recebendo o nome final SMB/CIFS;

Foi implementado no Unix através de engenharia reversa por Andrew Tridgell, permitindo o uso de servidores Unix para estações Windows. Usando as 3 letras, adotou uma palavra com as 3 letras (SAMBA) para identificar o projeto de software livre compatível com o SMB.

> Utiliza o protocolo NBT para emular redes NetBIOS sobre TCP/IP. O NBTS acabou sendo chamado de WINS - Windows Internet Name Server, cruzando endereços IP com nomes NetBIOS.

É tipicamente utilizado para acesso a NAS (*Network Attached Storage*);

Tipicamente tem performance inferior ao NFS, embora possa oferecer recursos mais avançados de segurança.

Serviços de Nome

Nomes, Identificadores e Endereços

Permitem compartilhar recursos, identificar entidades de maneira inequívoca e fazer referência a localizações.

Nomeação

Simples ou Estruturada

Resolução de Nomes

Associar à entidade ao qual se refere

Nomes - definições

Entidade: máquinas, impressoras, processos, usuários, páginas Web, janelas gráficas, volumes de discos etc.

- As entidades são acessadas através de um ponto de acesso, ou simplesmente, endereço; Ex: servidor e endereço IP
- Um endereço pode ser utilizado para nomear ou identificar uma entidade;
- Endereços podem mudar.

Desafio é nomear de forma independente da localização física

- Humanos utilizam strings; máquinas usam números e *hashs*;
- Identificadores referenciam apenas UMA entidade, e vice-versa; não podem ser reutilizados;
- Nomes amigáveis são normalmente hierárquicos (*path* e domínios).

Nomes - resolução

Implementa-se uma vinculação de um nome a um endereço

- Tipicamente é uma tabela de pares (nome, endereço);
- Apresenta problemas de performance para tabelas centralizadas em redes de grande porte.

Implementação em um SD costuma ser distribuída, visando eficiência e escalabilidade;

Classes Típicas:

- Nomes Simples;
- Nomes estruturados.

Nomes Simples

Não contém identificação do ponto de acesso. Estes podem ser localizados por:

- *Broadcast*. Ex: ARP
  - Ineficaz para grandes redes;
  - Tempestades de *broadcast*;
  - Ocupação de receptores não destinatários.
- *Multicast*
  - Limita dispositivos "endereçáveis";
- Ponteiros Repassadores
  - Cadeia de ponteiros não é tolerante a falhas;

Nomes Estruturados

Nomes são organizados em Espaços de Nome

- Grafo Dirigido, com 2 tipos de nó
  - Nó-folha: representa entidade nomeada, sem saídas;
  - Nó de diretório: refere-se a outros nós, pode ter várias saídas;
  - Nós de diretório armazenam tabela onde cada ramo de saída é representado por um par <rótulo do ramo, identificador do nó>.
- Tipicamente existe um nó raiz;
- Nomes de caminho podem ser absolutos (saindo do nó raiz), ou relativos (relativos a um nó específico);

Trabalho: Análise Comparativa

Estudar e apresentar principais características e diferenças

- Sistema de Arquivos SMB, NFS e NTFS
- Comparar atributos, ACLs, estrutura e organização
- Vantagens e desvantagens
- Participação de Mercado

Replicação

Replicação - motivadores

Desempenho

- Escalabilidade numérica (mais de um servidor);
- Escalabilidade geográfica (se existe um servidor mais próximo, isso simplifica comunicação)

Disponibilidade

Tolerância a Falhas

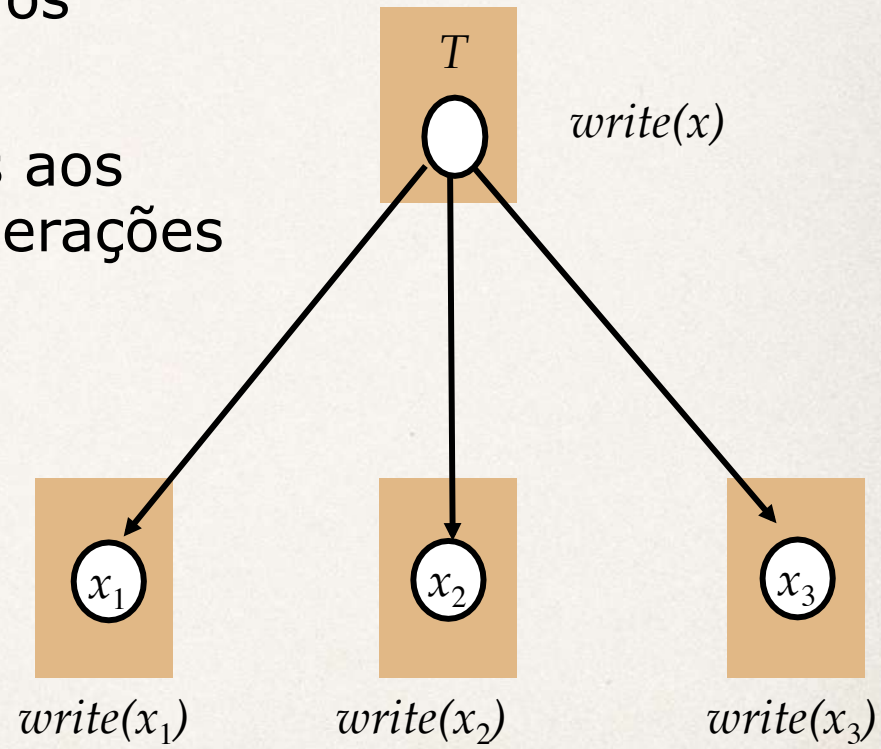
Transparência

Podem existir diversas cópias físicas de um único objeto lógico, que é o único que os usuários vêem;

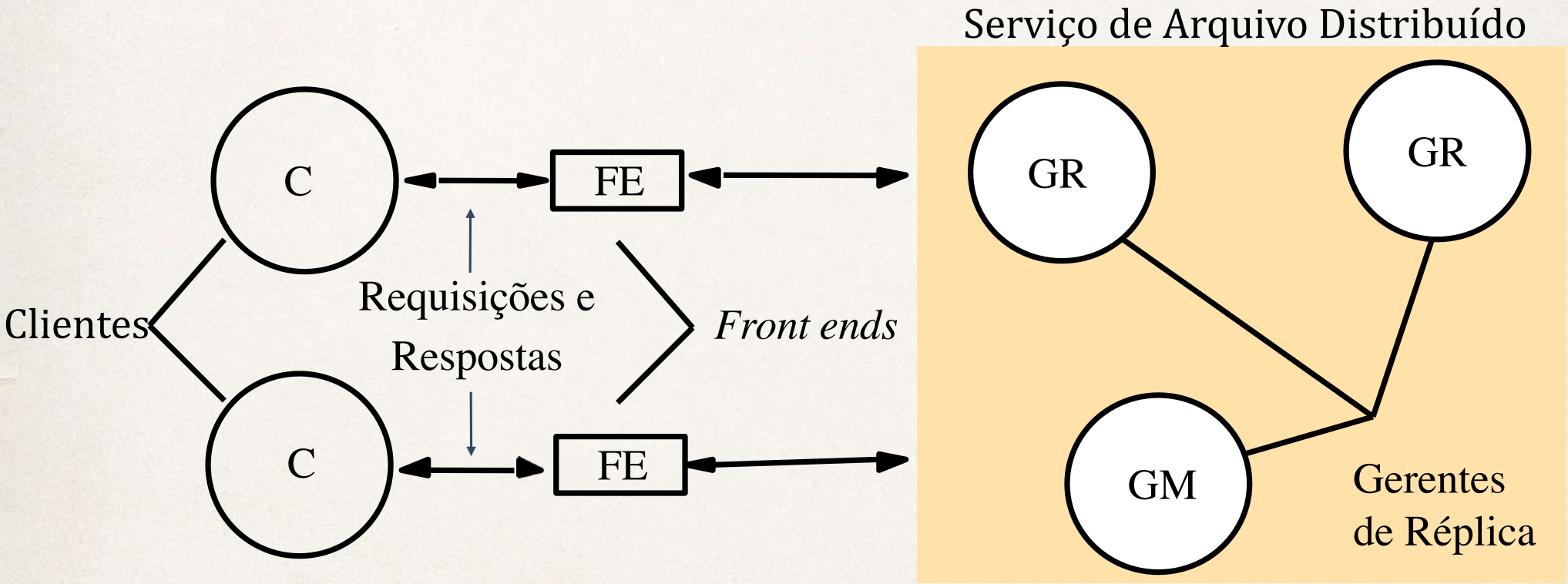
As operações são sempre direcionadas aos objetos lógicos, e "traduzidas" para operações nos objetos físicos pelo SAD;

Até mesmo a desconexão temporária não deve provocar problemas, já que as cópias físicas podem ser sincronizadas após o reestabelecimento da conexão

- O desafio da operação desconectada.



Arquitetura de Replicação



Coerência & Consistência

Coerência: a ordem das operações em um item de dados deve ser obedecida.

Ex: a leitura de x deve retornar o valor da gravação mais recente de x.

Consistência: a ordem deve ser preservada para todos os dados;

- Exige coerência nos dados individuais;
- Estende o conceito para operações envolvendo outros itens de dados.

Métodos de Ordenação das escritas

Na “ordenação forte”

- Todas as escritas devem ser obedecer à ordem de solicitação;
- Pode-se garantir as consistências estrita, sequencial e causal;

Na “ordenação fraca”

- As escritas são ordenadas por grupo, e não individualmente;
- Pode-se garantir as consistências fraca, saída (*release*) e saída (*entry*);

Consistência eventual, ou orientada ao cliente

O desafio é definir a propagação de atualizações e réplicas

Consistência na ordenação "Forte"

Estrita	Sequencial (Lamport)	Causal
Qualquer leitura de X retorna o valor mais recente da escrita de X.	Resultado das operações sempre corresponde à execução sequencial das mesmas.	A sequência é a mesma para todos os processos em escritas potencialmente causais.
- Tempo absoluto global; - Todas as operações são ordenadas; - É irrealizável em um SD.	- Podem existir 2 ou mais ordem corretas; - Todos vêem a mesma ordem; - Não há correlação obrigatória com o tempo. -	- Escritas concorrentes podem ser vistas em ordem diferente por diferentes processos.
	P1: W(x)a P2: W(x)b P3: R(x)b R(x)a P4: R(x)b R(x)a	P1: W(x)a W(x)c P2: R(x)a W(x)b P3: R(x)a R(x)c R(x)b P4: R(x)a R(x)b R(x)c

Consistência na ordenação "Fraca"

Fraca	Entrada	Saída
Os dados compartilhados só estão consistentes após sincronização.	Dados são atualizados antes da entrada na zona crítica.	Dados são atualizados antes na saída da zona crítica.
- Consistência por grupo de operações; - Variáveis de sincronização; - Definição de "zona crítica".	- Baseia-se na operação ACQUIRE(S); - *Lock*?	- Basea-se na operação RELEASE(S); - *Unlock*?
- Operações de sincronismo são sequencialmente consistentes; - Sincronismo apenas após todas as operações de escrita; - Leitura ou escrita apenas após a sincronização.		

Consistência eventual

Alguns SDs requerem escritas muito raramente

- A velocidade da propagação das escritas pode não ser crítica;
- Inconsistências temporárias podem ser toleráveis;
- Sem alterações frequentes, a convergência pode demorar.

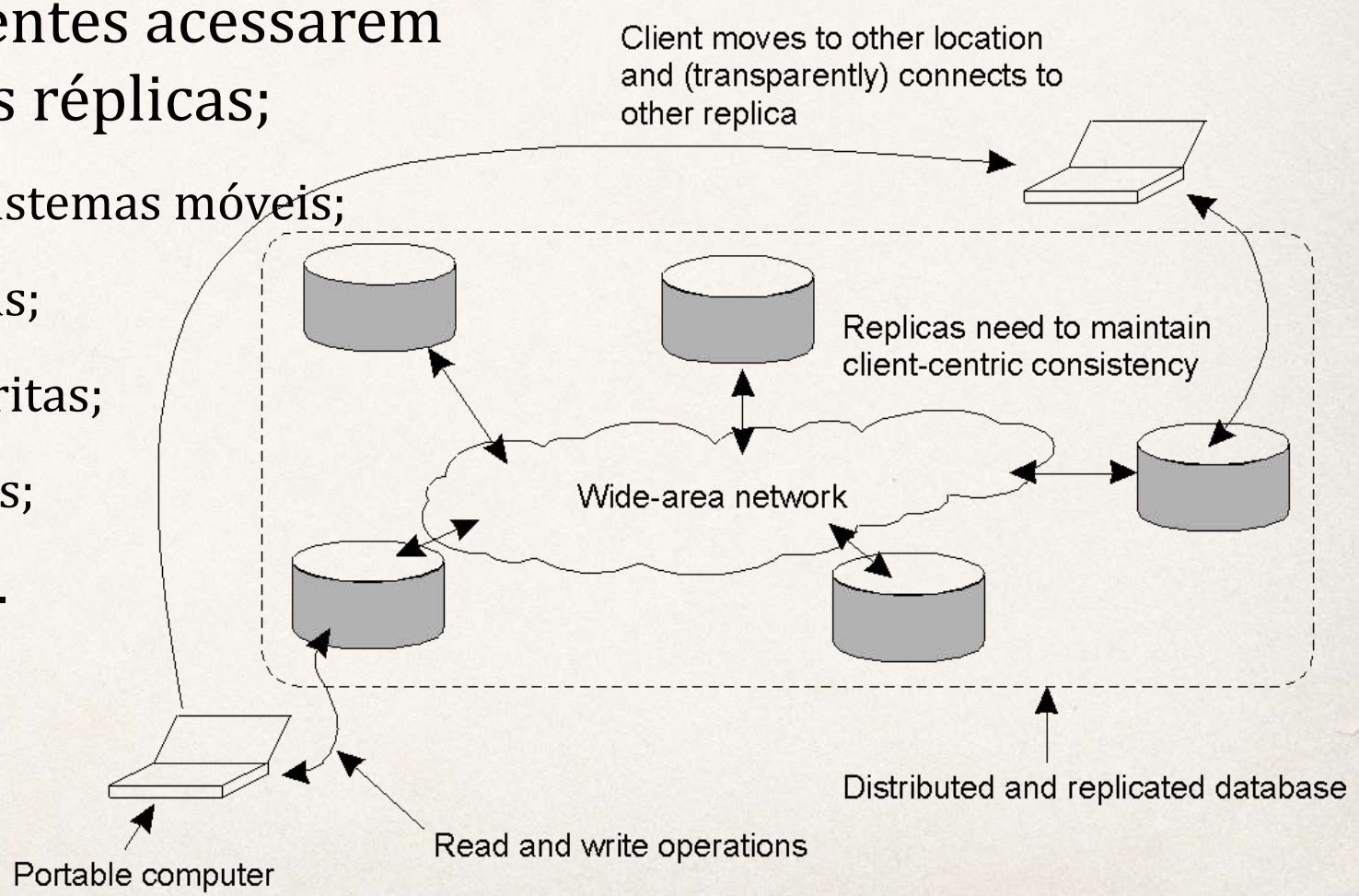
Consistência “barata”, ou eventual

- Ex: Cache de navegadores, DNS.

Consistência Eventual - problemas

Funciona se os clientes acessarem sempre as mesmas réplicas;

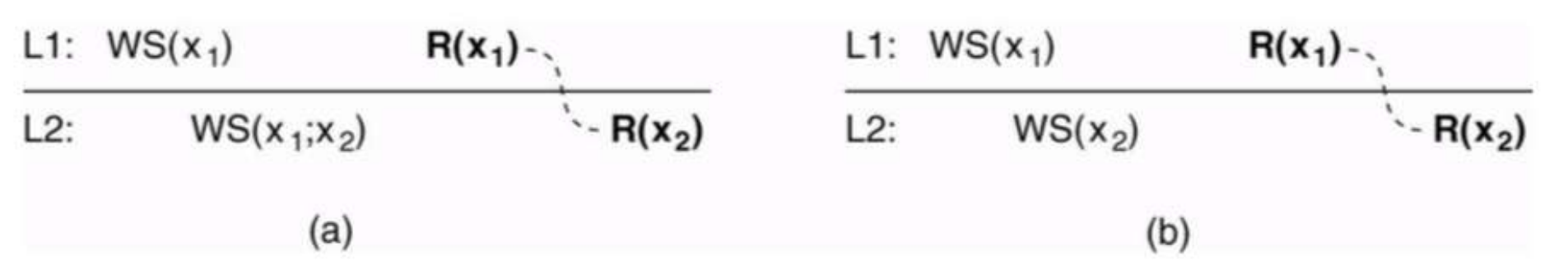
- Problemático para sistemas móveis;
- Leituras Monotônicas;
- Leitura das suas escritas;
- Escritas Monotônicas;
- Escritas após leitura.



Leitura monotônica

Se um cliente ler o valor de X no tempo T, leituras posteriores de X retornarão sempre o mesmo valor X, ou mais recente, independente da réplica;

Na figura abaixo, (a) é consistente, e (b) não é.



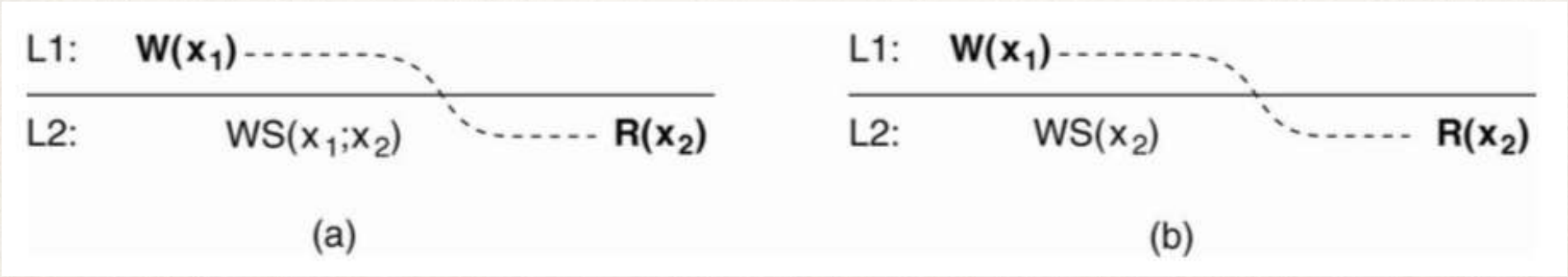
Aula 10

V.1.11 - 2021.10

Leitura das suas escritas

Se um cliente gravar o valor de X, todas as leituras posteriores retornarão X, independente da réplica;

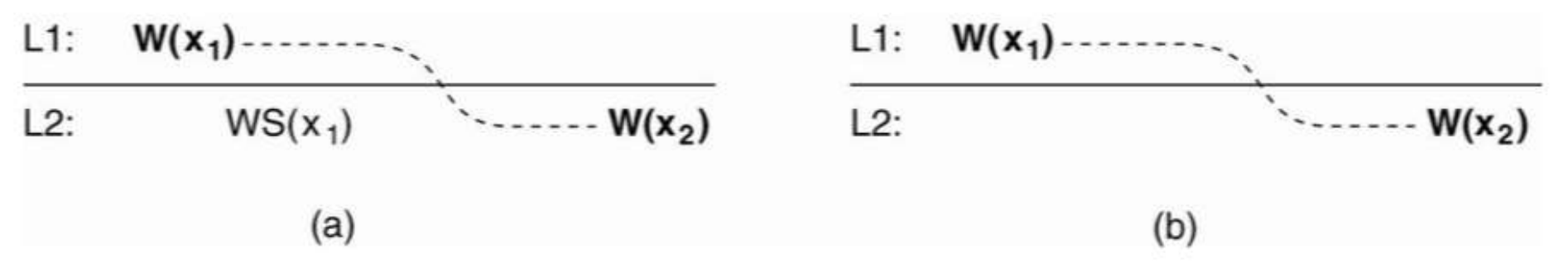
Na figura abaixo, (a) é consistente, e (b) não é.



Escrita monotônica

Se um cliente escrever o valor X, a operação será sempre completada antes de quaisquer novas escritas de X pelo mesmo cliente, independente da réplica;

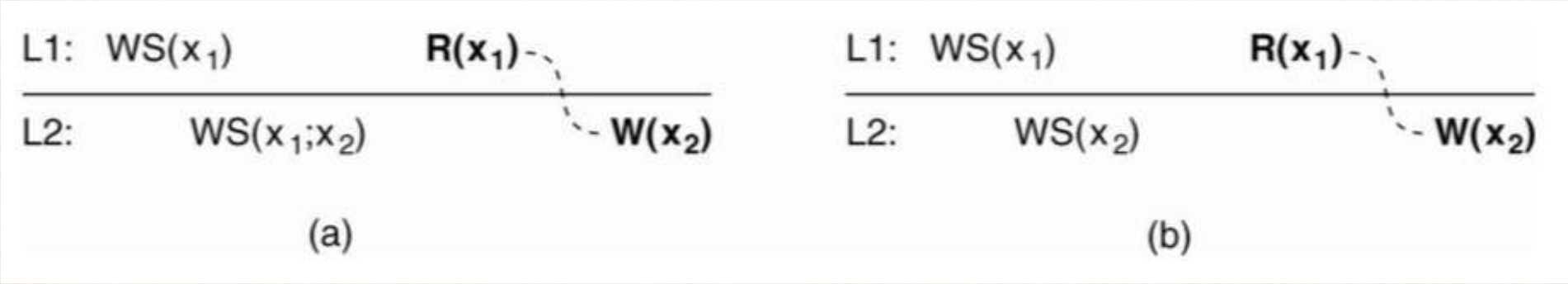
Na figura abaixo, (a) é consistente, e (b) não é.



Escrita após leituras

Se um cliente gravar o valor de X após ter lido o mesmo valor, a escrita ocorrerá sempre sobre a cópia lida, ou outra que seja mais recente, independente da réplica;

Na figura abaixo, (a) é consistente, e (b) não é.



Qual o método de Consistência?

Redundância

- Todas as cópias fortemente consistentes;
- Todas as cópias íntegras;

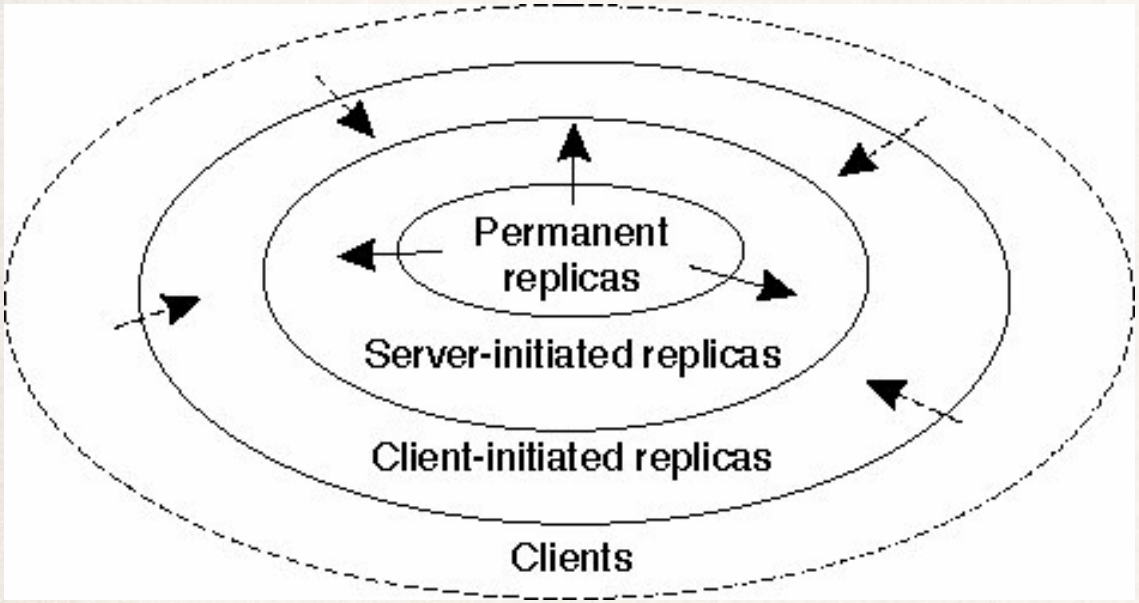
Desempenho

- Consistência exige mais processamento e comunicação;
- Perda de desempenho tem que ser considerada;

Escalabilidade

- O modelo precisa ser escalável para ambientes maiores;
- Evitar centralização e reduzir requisitos de comunicação.

Alternativas de Replicação



Permanentes

Espelhamento de réplicas específicas;

Iniciadas pelo servidor

O servidor decide (modo *push*);

Distribuição de carga;

Colocar réplicas próximas dos clientes;

Iniciadas pelos clientes

Os clientes decidem (modo *Pull*).

Propagação de Atualizações

O que encaminhar?

Notificações
Pode informar, por exemplo, a invalidade de uma operação

Dados atualizados
Uma opção seria encaminhar apenas os LOGs, a depender do ambiente

Operações realizadas
Chamada de “replicação ativa” - as réplicas precisam executar as operações novamente

Quem é responsável?

Servidor (push)

Cliente (pull)

Protocolos “epidêmicos"

A propagação acontece em eventos de consistência específicos.

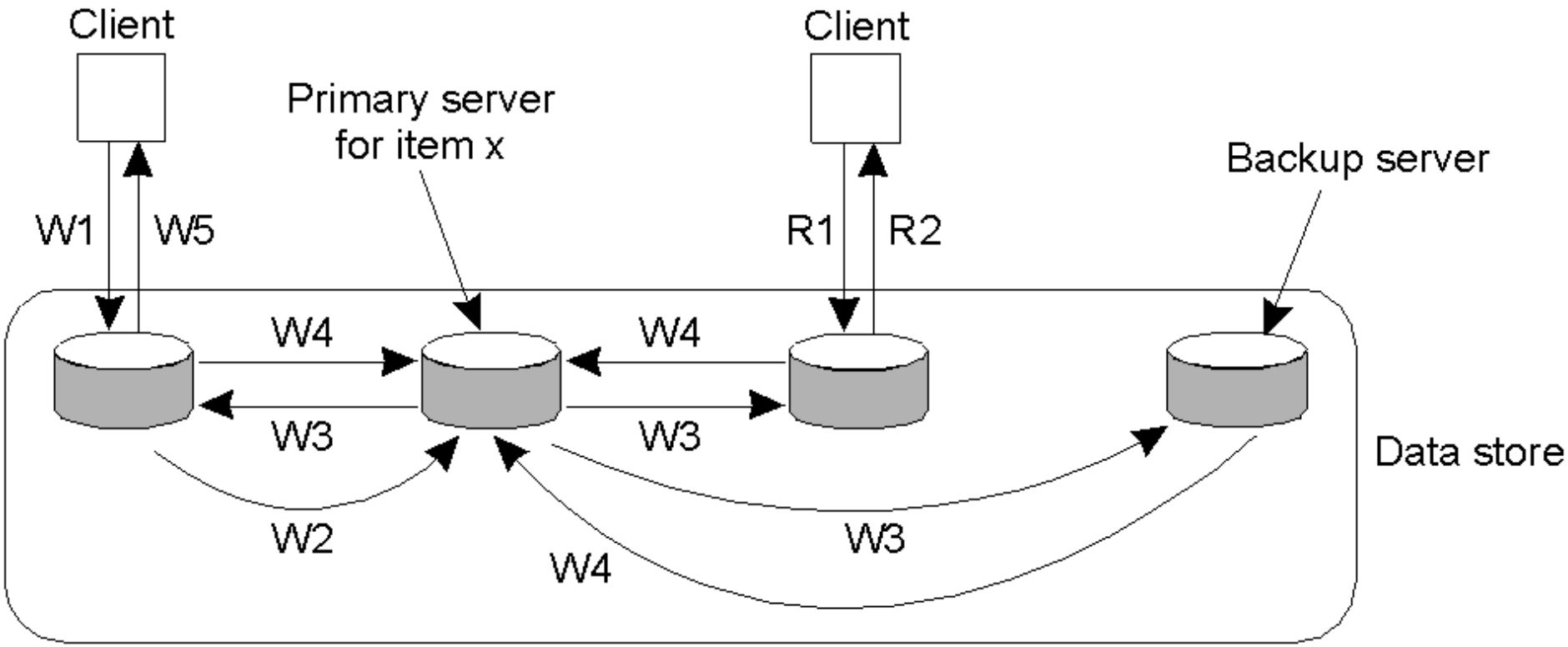
Protocolos *Push* X *Push*

Item	**Modo *Push***	**Modo *Pull***
Estado do servidor	Lista das réplicas e *caches* de clientes	Não é propagado
Mensagens enviadas	Atualiza	Envia e atualiza
Tempo de Resposta (cliente)	Imediato (ou dentro do tempo de atualização)	No tempo de atualização

Protocolos de Replicação

- Protocolos Primários
  - Escrita remota
  - Escrita Local
- Protocolos de escrita replicada
  - Replicação ativa
  - Protocolos baseados no “*quorum*"
- *Read-one-Write-All* (ROWA)

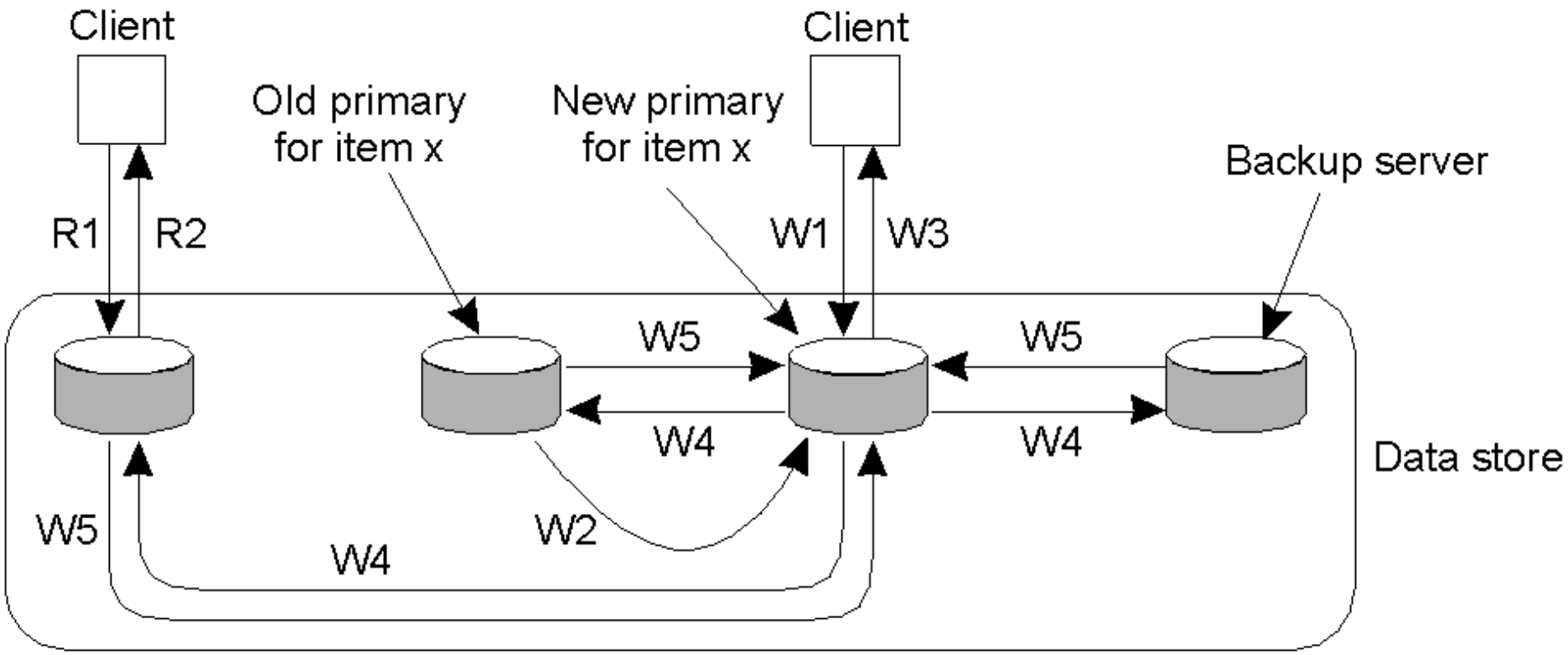
Protocolo *Primary Copy Remote-Write*



W1. Requisição de Escrita
W2. Encaminha requisição ao repositório primário
W3. Solicita atualização às replicas (back-up)
W4. Informa Atualização
W5. Informa completude da escrita

R1. Requisição de Leitura
R2. Resposta à solicitação de Leitura

Protocolo *Primary Copy Local-Write*



W1. Requisição de Escrita
W2. Move o item para o novo repositório local
W3. Informa completude da escrita
W4. Solicita atualização das réplicas (back-up)
W5. Informa completude da atualização

R1. Requisição de Leitura
R2. Resposta à solicitação de Leitura

Computação Móvel e Ubíqua

V.1.11 - 2021.10

Móvel e Ubíqua?

Móvel

- Dispositivos Portáteis (e portanto, móveis);
- Características específicas: menores, maior autonomia de bateria, menor capacidade de processamento;
- Transmissão sem fio (energia $\propto$ distância^2);

Ubíqua (ou pervasiva)

- Integração com o mundo contemporâneo;
- Está disponível em todos os locais.

Volatividade

Característica padrão dos sistemas móveis ubíquos

- Envolve dispositivos, usuários e componentes de SW;
- Envolve inclusive a bateria dos dispositivos móveis !

SW associa-se e interage quando ocorre:

- Mudança, falha, ou “aparecimento”;
- Necessidade de percepção e reconhecimento de contexto;
  - Noite/Dia; Dentro do Carro; Tocar/Vibrar no cinema?
- Segurança e Privacidade;
- Adaptação à falta de recursos computacionais e de E/S;
- Sensores e atuadores.

Associação

Dispositivos podem ser "*wearable*"

Ex: crachás inteligentes;

Neste caso, a associação precisa ser automática.

Tipos de Associação, e a questão da segurança

Previamente configurada	Espontânea
Orientada a serviços: *cliente e servidor de e-mail*	Orientada a usuários humanos: *navegador web e servidores web* Orientada a dados: *aplicativos de compartilhamento de arquivo P2P* Fisicamente orientada: *sistemas móveis e ubíquos*

Associação

Associações Físicas

- Por interação humana (Ex. Informa o quarto do Hotel);
- Por percepção usando sensores (Ex. QR Code);
- Associação direta (Ex. Conexão de rede, BT);
- Percepção por proximidade (Ex. NFC);
- Correlação Temporal/Física (Ex. Botões de configuração rápida).

Interoperabilidade

- Associar é diferente de interoperar (ambiente não controlado);
- Interfaces heterogêneas, porém adaptadas?
- Sintaxe idêntica (Ex. pipe Unix, HTTP);

Serviços de Descoberta

Baseados em dispositivos

Dado um dispositivo, verifica-se os serviços disponíveis;

Baseados em Serviços

Não interessa o dispositivo;

Método mais indicado.

Arrendamento

Frequência alta: problema com bateria;

Frequência baixa: problema com a volatilidade.

Problemas de Segurança

Hardware

Roubo e falsificação é mais simples em dispositivos móveis;

Capacidade computacional

Criptografia exige recursos;

Consumo de energia

Ataque de "privação de sono" contra modo "sleep";