

Arquiteturas tolerantes a falhas

micro sistemas COTS

Taisy Silva Weber
UFRGS

Arquiteturas tolerantes a falhas

✓ arquitetura: nível eficaz para suportar TF

✓ componentes

✓ conexões

processadores,
memórias,
controladores,
interfaces

barramentos ou
linhas de
comunicação

Tolerância a falhas de um sistema pode ser implementada ou por hardware, ou por software, ou ambas.

Hardware: mais eficiente.
Software: mais flexível.

Arquiteturas tolerantes a falhas

Existiram máquinas com o nome de **tolerantes a falhas**. Atualmente seriam chamadas de disponibilidade contínua ou alta disponibilidade.

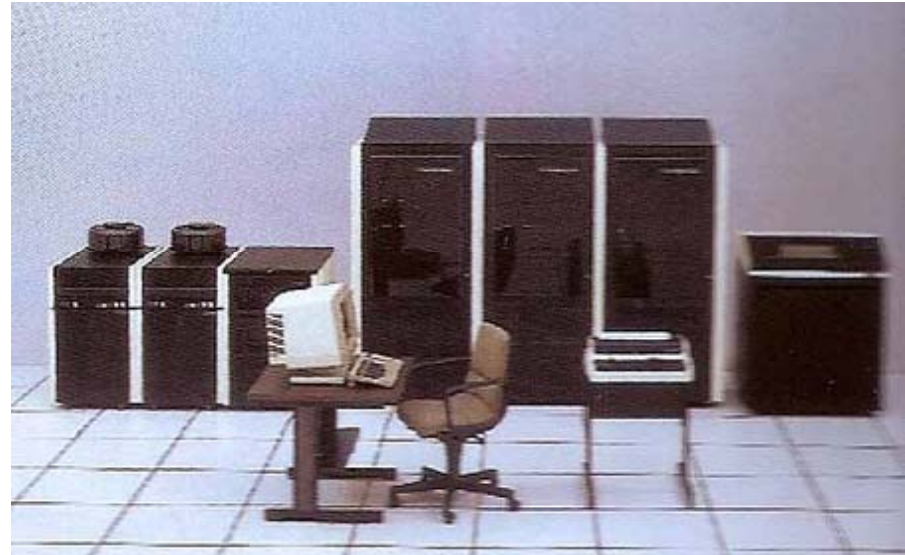


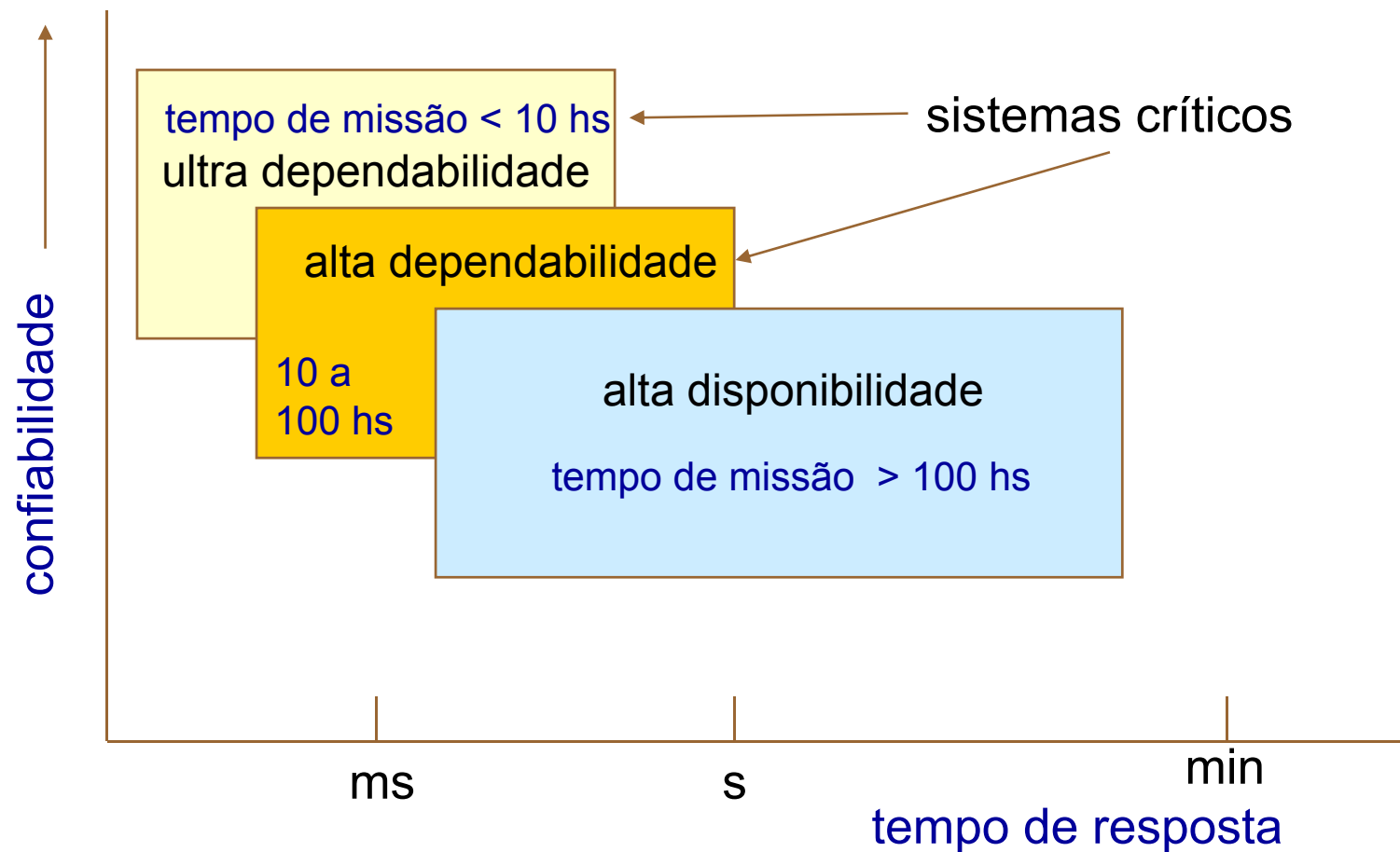
foto:

http://www.siliconvalleyfamilytree.org/home/tandem_computers

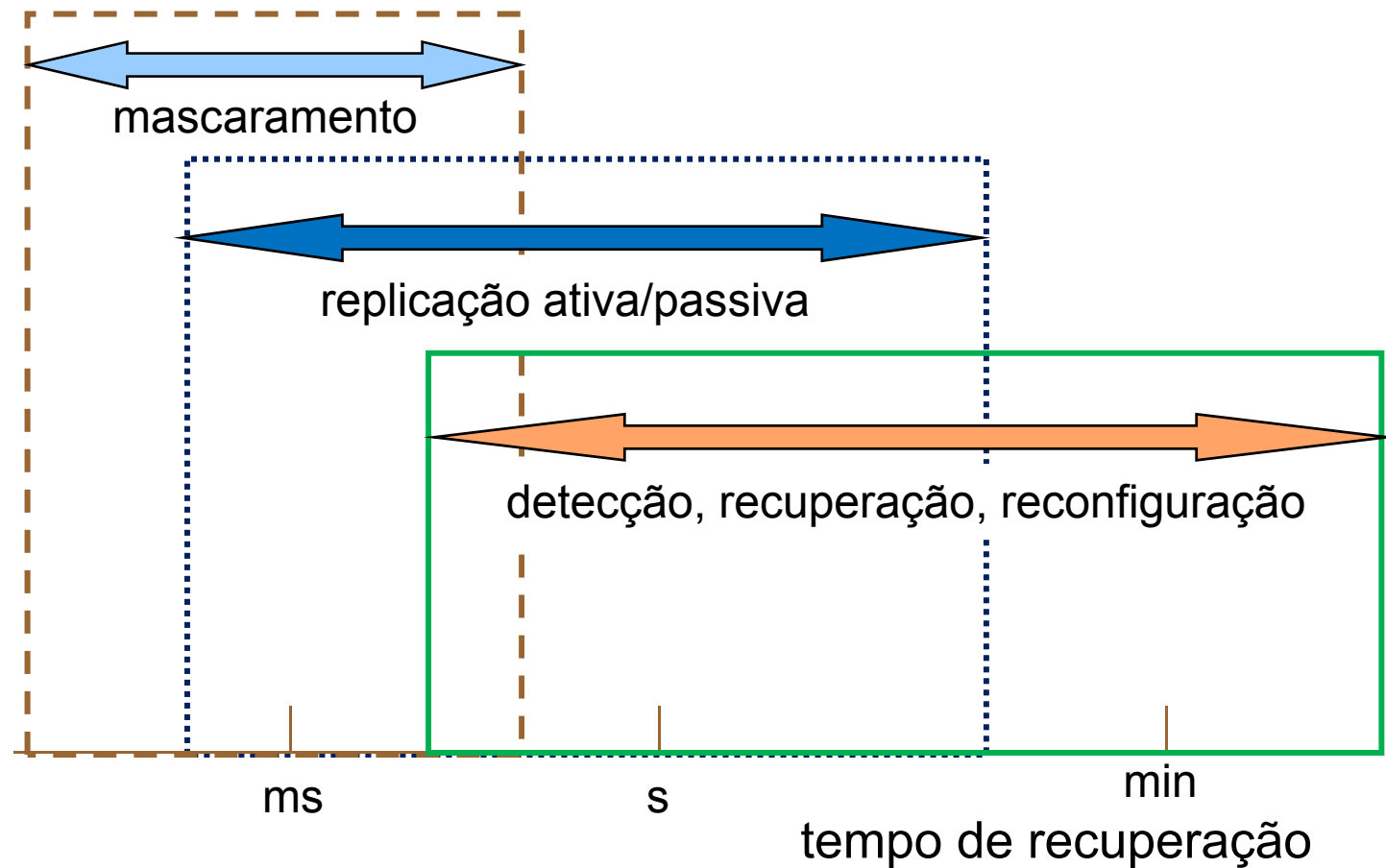
TF (aqui) é usada no sentido de arquiteturas que empregam qualquer das técnicas de tolerância a falhas.

Domínio de aplicação

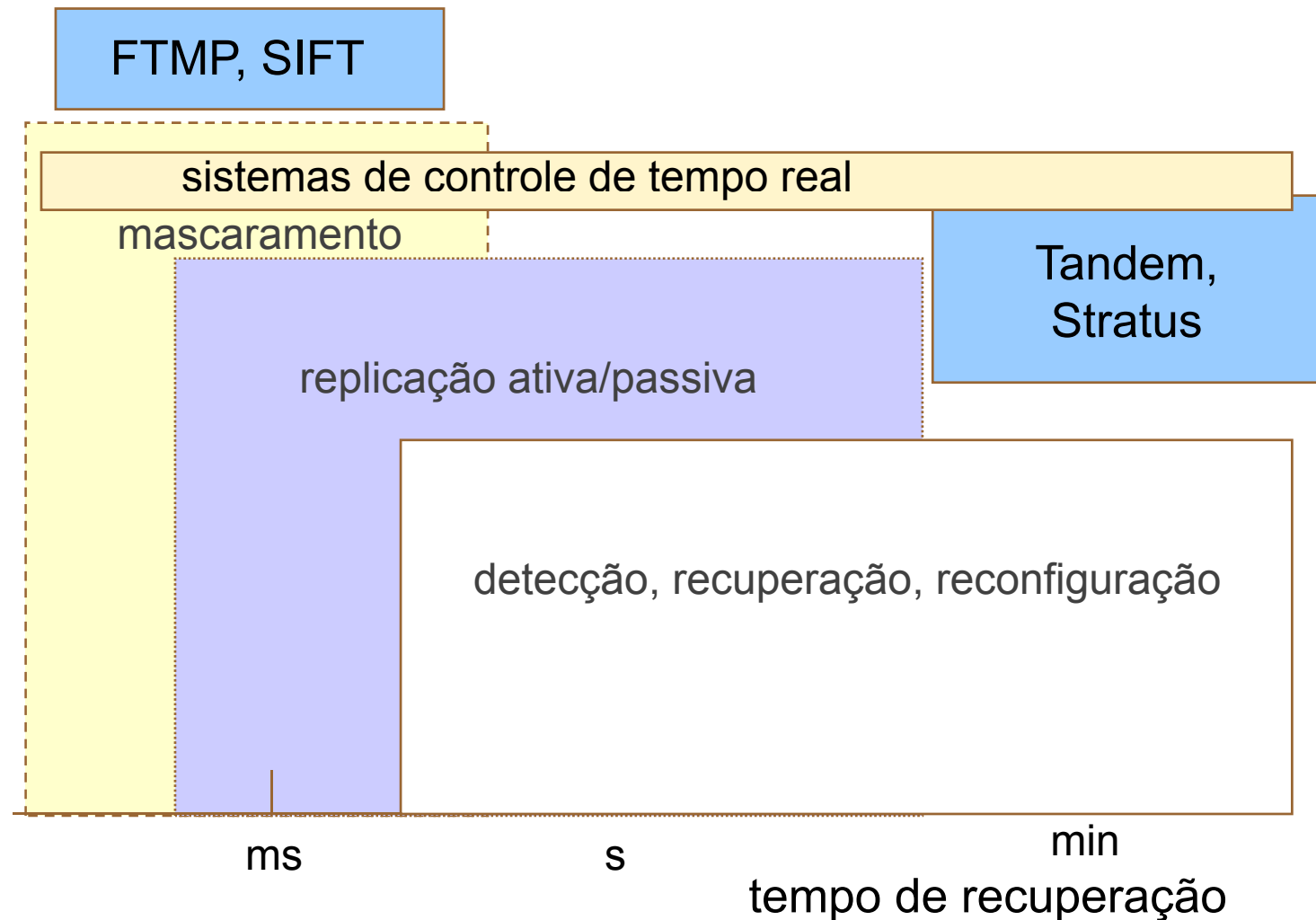
SURI, N.; WALTER, C.J.; HUGUE, M.M. **Advances in ultra-dependable distributed systems**. IEEE Computer Society Press. Los Alamitos. 1995.



Domínio de técnicas



Exemplos de sistemas

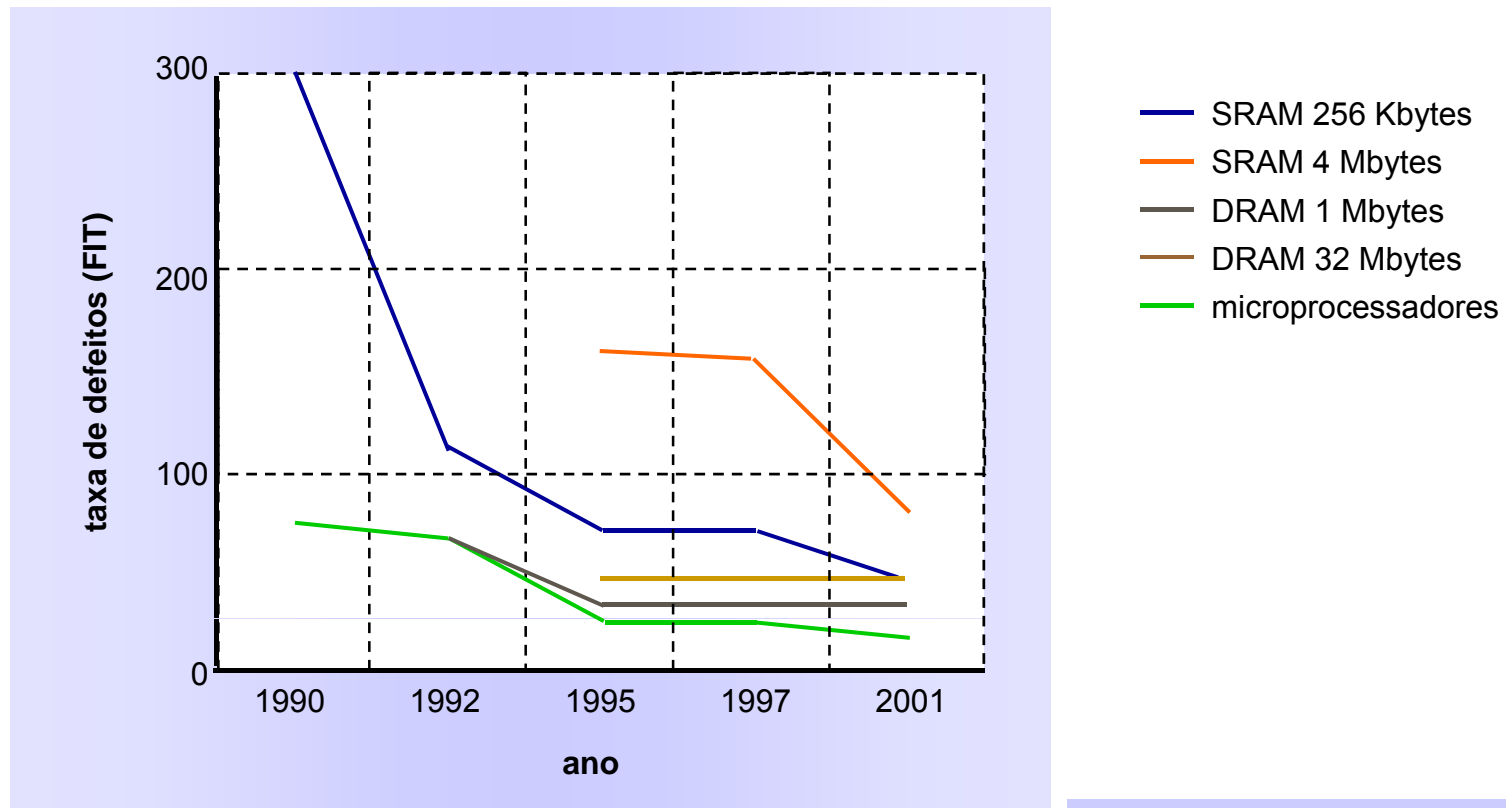


- ✓ componentes comerciais
 - ✓ produzidos em larga escala
 - ✓ baixo custo
 - ✓ facilidade de obtenção
 - ✓ baixa **taxa de defeitos** (para falhas permanentes)
 - ✓ FIT baixo e com tendência a diminuir
 - ✓ (FIT = failures per 10^9 hours)
 - ✓ muito suscetíveis a falhas transientes



desafio - construir sistemas tolerantes a falhas com componentes não confiáveis

Falhas permanentes em COTS



taxa de defeitos (em FIT) devido a falhas permanentes em dispositivos CMOS (COTS) na década de 90 até 2001

FIT = failures per 10^9 hours

Cristian Constantinescu, TRENDS AND CHALLENGES IN VLSI CIRCUIT RELIABILITY. IEEE Micro, 2003

Fontes de defeitos em COTS

✓ mas

- ✓ falhas permanentes não são a principal fonte de preocupação

FIT = failures per 10^9 hours
FIT é medida de taxa de defeito
(devido a falhas permanentes)

✓ problema maior: falhas transientes

- ✓ alta susceptibilidade a interferências ambientais

redução no tamanho dos componentes e no nível de potência aumenta a susceptibilidade a interferências por radiação e outros fatores que causam falhas transientes

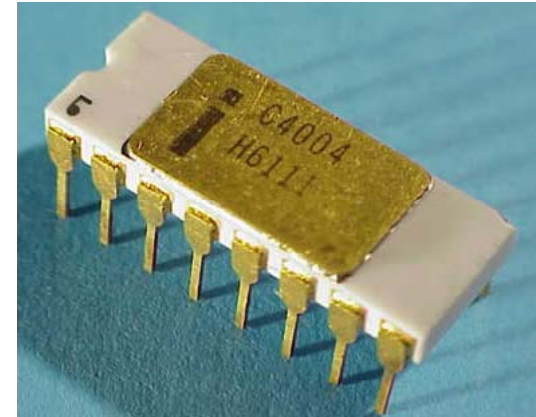
Estudo google sobre memória

- ✓ 2 anos
 - ✓ servidores com memória protegida por ECC
 - ✓ ECC comum (SECCDED) e Chipkill
 - ✓ chipkill de 4 a 10 vezes mais eficiente que SECCDED
 - ✓ erros corrigidos
 - ✓ mais do que 8% dos chips e 1/3 das máquinas por ano
 - ✓ FIT: 25000 a 70000 por Mbit
 - ✓ erros não corrigidos
 - ✓ 0,22% dos chips e 1,3% das máquinas
 - ✓ fortes evidências que erros *hard* são mais comuns que *soft*

Google Inc.: Schroeder, B.; Pinheiro, E.; and Weber, W. “**DRAM Errors in the Wild: A Large-Scale Field Study**.” SIGMETRICS/Performance '09, Seattle, WA, June 15-19, 2009.

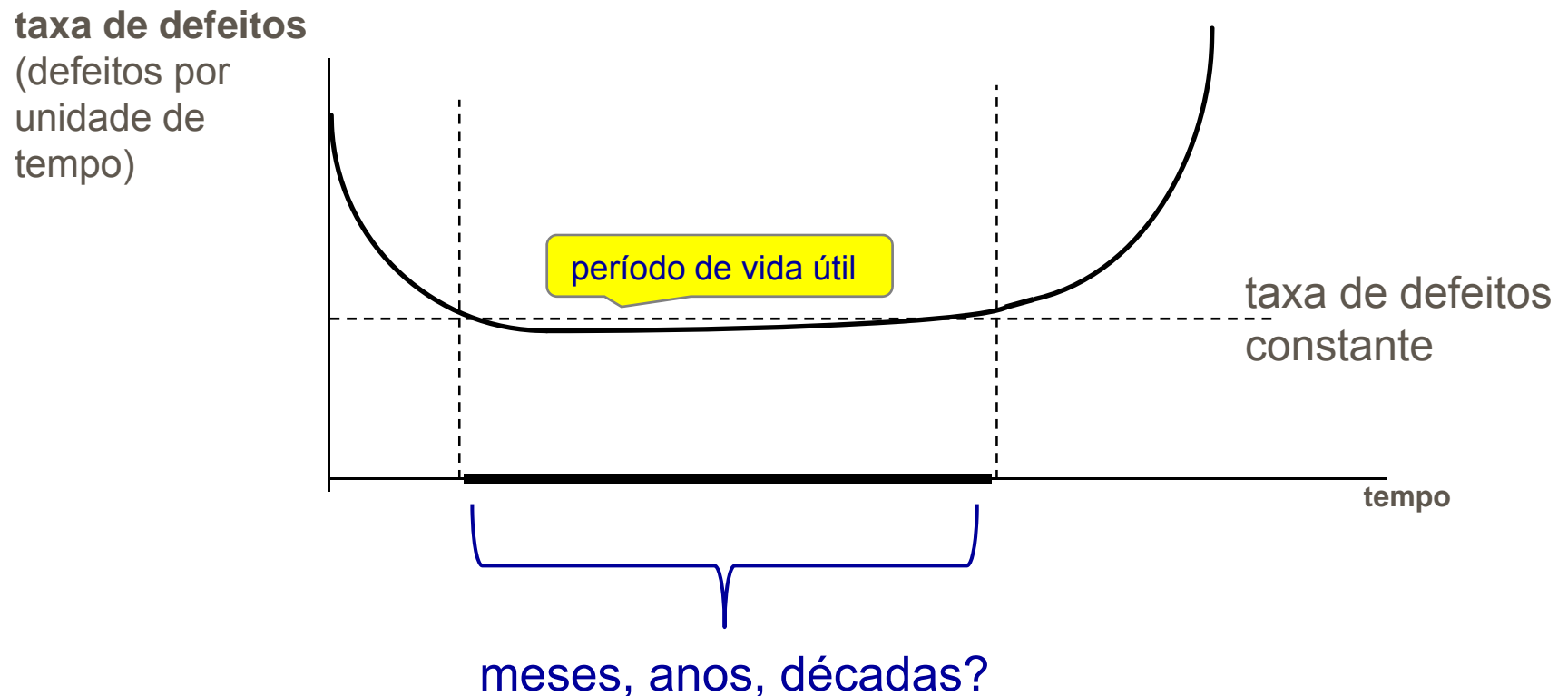
Desgaste em COTS

- ✓ incerteza sobre desgaste
 - ✓ medida de desgaste
 - ✓ importante para sistemas fechados em aplicações de vida longa
- ✓ desgaste pode começar após alguns anos
 - ✓ ainda sem medidas de envelhecimento
- ✓ taxa de defeitos pode aumentar no tempo
 - ✓ FIT baixo e constante pode ser uma medida com validade por curto espaço de tempo



Incerteza quanto a desgaste

qual o tempo de vida útil para o qual vale o FIT baixo e constante?



TF em microprocessadores comerciais

- ✓ solução externa:
 - ✓ duplicação ou replicação de chips
 - ✓ hardware adicional (votadores e comparadores)
- ✓ solução interna ao chip:
 - ✓ TF suprido pelo próprio microprocessador

basta diminuir o FIT dos microprocessadores e demais chips para garantir um sistema tolerante a falhas?

TF em COTS: exemplos

✓ Intel

- ✓ desde o 486 na família x86
- ✓ 432 na década de 70

432 era excelente conceitualmente, mas foi um fiasco comercial (alguns afirmam que jamais foi produzido)

sem o emprego de TF, os chips atuais não funcionariam, o MTTF seria muito pequeno

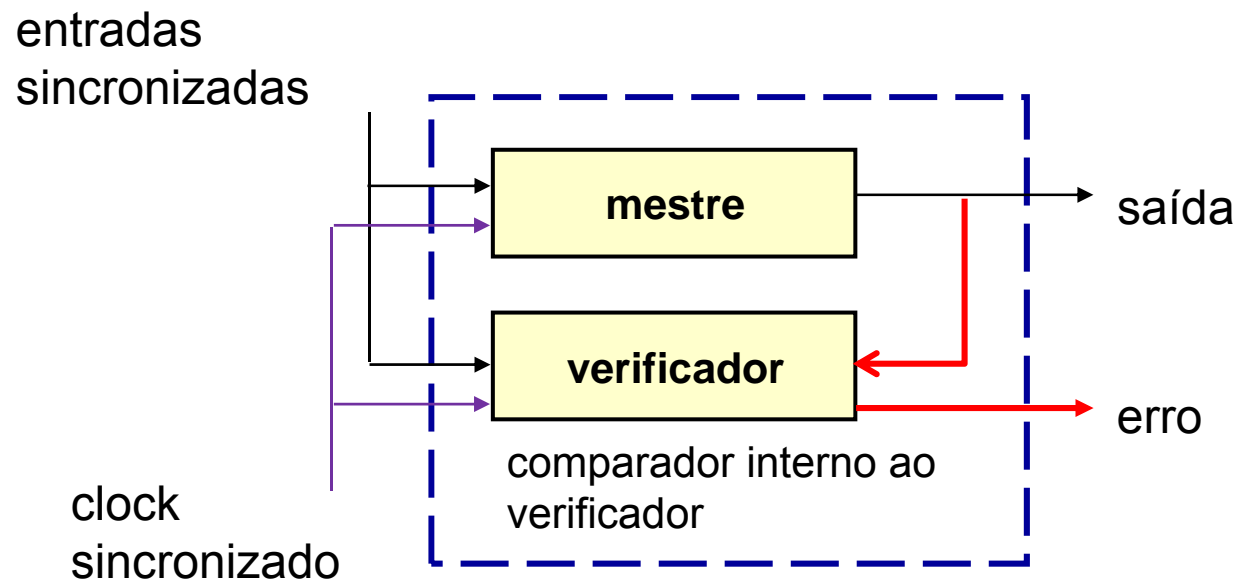
- ✓ microprocessadores Intel
 - ✓ 100 FITs (FIT = failures per 10^9 hours)
 - ✓ MTTF potencial aprox. 1100 anos
- ✓ grande MTTF não indica ausência de problemas
 - ✓ suscetibilidade a falhas transientes
 - ✓ incerteza sobre desgaste
 - ✓ numerosas falhas de projeto ([errata](#))
 - ✓ Intel P6 (início de 1999): 45 a 101 falhas de projeto
 - ✓ novas erratas: taxa de uma por mês

comportamento sob falhas transientes e erratas indicam a necessidade de tolerância a falhas externa ao chip (Avizienis)

Micros Intel - FRC

FRC - Functional Redundancy Checking

mestre e verificador devem estar sincronizados clock-by-clock (lockstep)



Pentiums

paridade: dados, caches, TLB e
memória de microcódigo
verificação de exceções: MCA
mestre / verificador **FRC**

ECC para dados

2 bits de paridade para linhas de
endereço associado a técnicas de retry
(re-tentativa)

paridade para sinais de controle

detecção de erros por paridade para
caches e barramentos

barramento de dados e memória com
ECC

Pentium



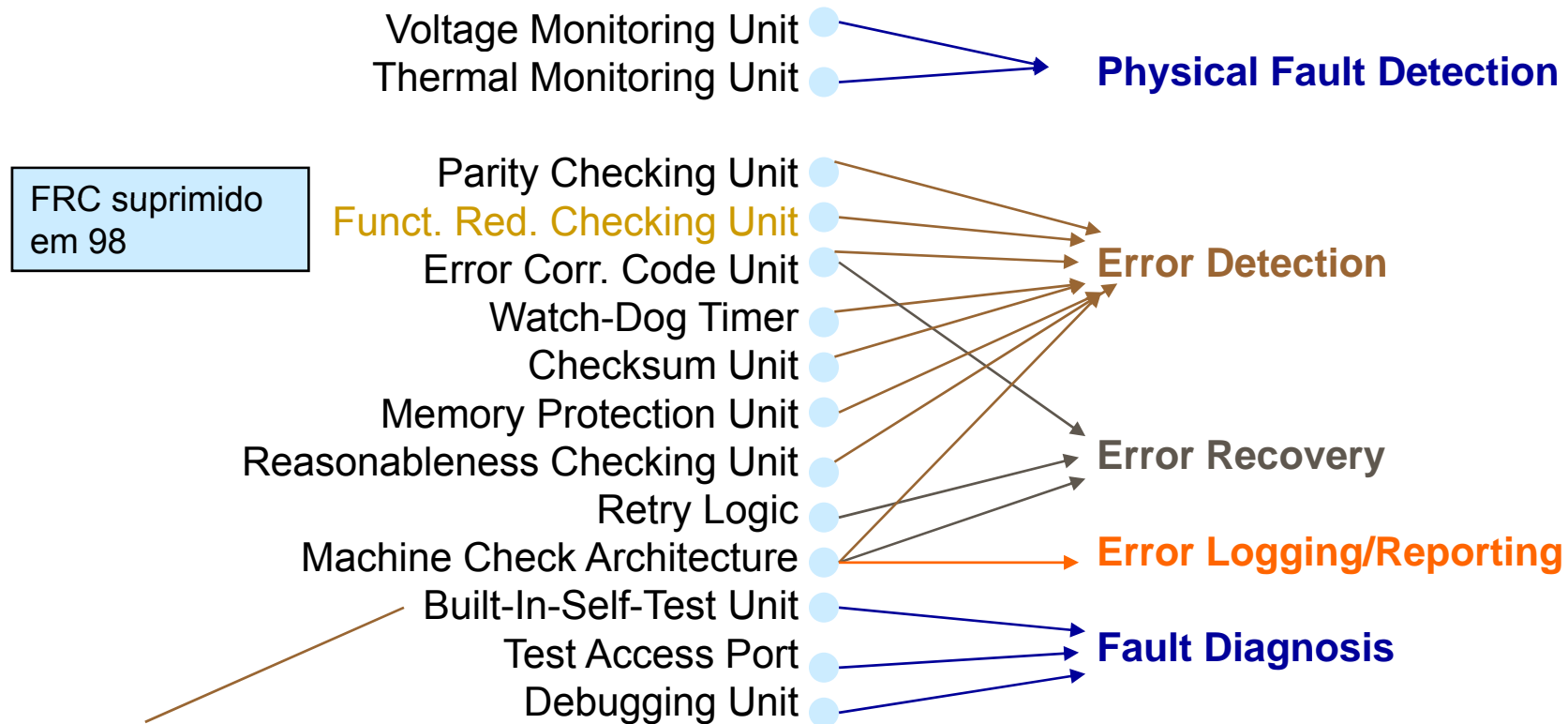
Pentium
Pro



Intel P6

AVIZIENIS, A. Fault Tolerance Infrastructure for
Dependable Computing with High-Performance COTS
Components. DSN, IEEE 2000

Pentium II



para teste dos elementos de memória interna (microcódigo, caches, TLB)



Itanium 2

- ✓ recuperação de erros de barramento de dados
- ✓ cache ECC (já existente no P6)
- ✓ correção de erro simples de memória
- ✓ re-tentativa na detecção de erro duplo de memória
- ✓ suporte a memória espelhada (spare)
- ✓ verificação de erros soft (transientes) na lógica interna: bit de paridade
- ✓ suporte a *lockstep*
- ✓ contenção de dados corrompidos

memória com defeito é substituída por memória estepe



marca a porção de memória com dados corrompidos e limita o uso dos dados a apenas um programa;
elimina os dados quando o programa termina ou sobrescreve a porção

Reliability, Availability, and Serviceability for the Always-on Enterprise. White paper, Intel, 2005

- ✓ machine check architecture
 - ✓ registradores dedicados para log de erros
 - ✓ facilita diagnóstico
 - ✓ capacidade de manipulação de erros

P6

MCA opcional - pode ser desligado por software

- ✓ Advanced MCA
 - ✓ segue padrões
 - ✓ padrões facilitam a interface com o SO e firmware
 - ✓ permite ao SO e ao firmware recuperarem erros complexos
 - ✓ pode *resetar* o sistema automaticamente em resposta a erros fatais

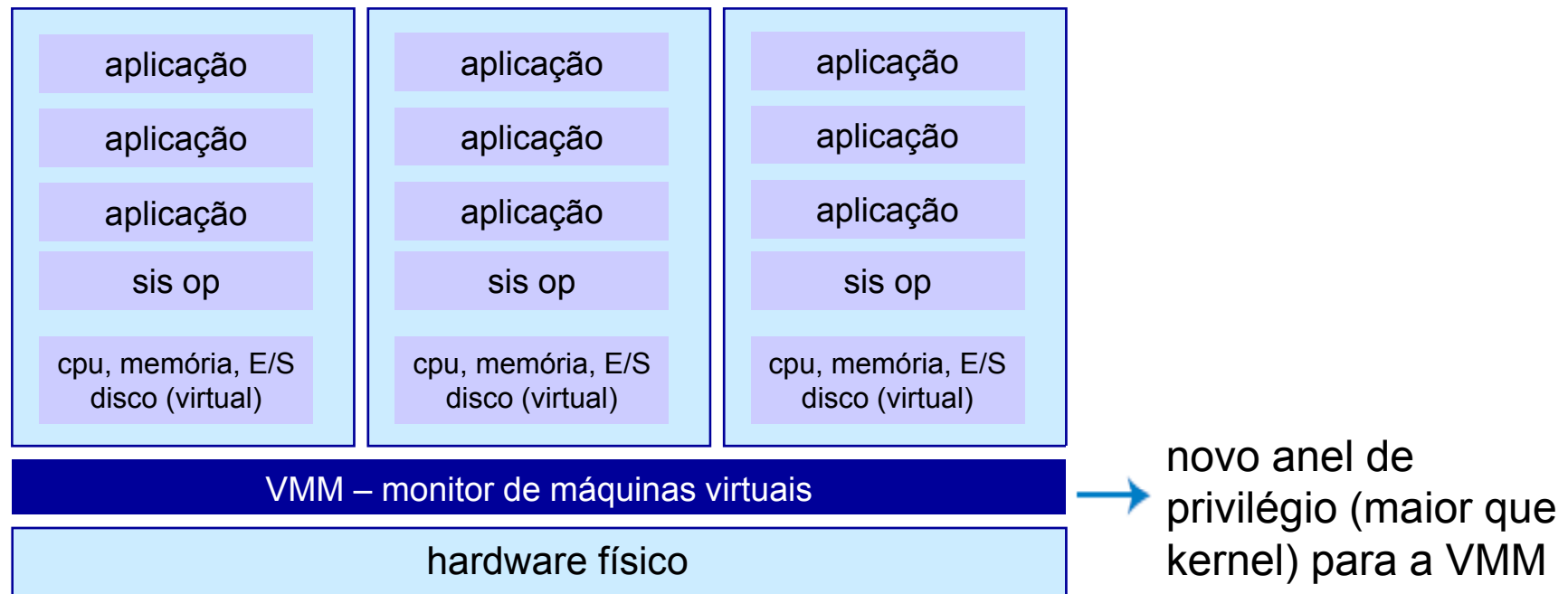


Itanium 2

Virtualização Intel

VM (máquina virtual):
isolamento de falhas por hardware
possibilidade de implementar *failover* na mesma máquina

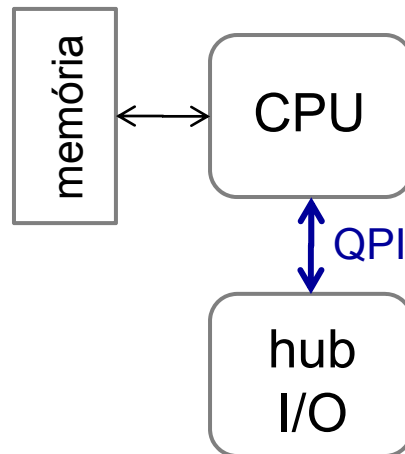
Enhanced Virtualization on Intel®
Architecture based Servers - White
paper, Intel 2004



VMM – software que opera como árbitro no acesso aos recursos físicos e hospeda as máquinas virtuais

Intel i7 e Xeon

- ✓ Intel® QuickPath Interconnect (QPI)
 - ✓ detecção de erro com CRC
 - ✓ correção de erro usando *Link level retry*
 - ✓ Intel® Interconnect Built In Self Test



✓ MCA Recovery

- ✓ permite recuperação do sistema
- ✓ sinaliza o erro para o SO ou VMM que podem então recuperar o erro (se for possível) sem derrubar todo o sistema

herança do Itanium

Intel White Paper: Intel® Xeon® Processor E7 Family: Reliability, Availability, and Serviceability - Advanced data integrity and resiliency support for mission-critical deployments. 2011

Intel Xeon processor E7

1 PROCESSOR/SYSTEM

- Corrupt Data Containment Mode
- Electronically Isolated Partitioning
- Processor Sparing and Migration*
- Core (Socket) Disable for Fault Resilient Boot
- Machine Check Architecture Recovery (MCA Recovery)*
- CPU Hot Add*
- PCIe Express Hot Plug
- Corrected Machine Check Interrupt (CMCI) for Preventive Failure Analysis*

*Requires operating system support.

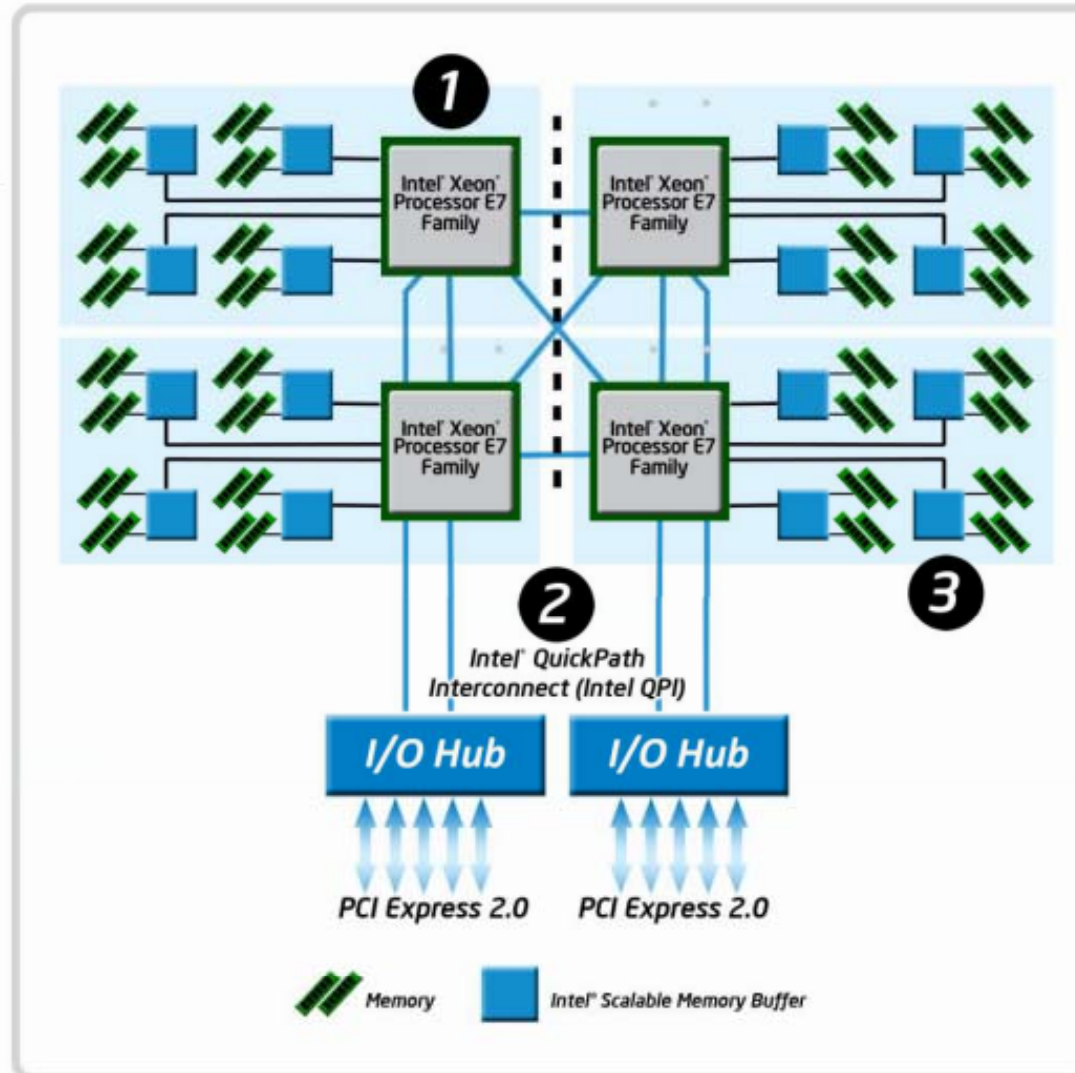
2 INTEL® QPI

- Intel QPI Protocol Protection via CRC
- QPI Viral Mode
- Intel QPI Self-healing
- QPI Clock Failover
- QPI Packet Retry

3 MEMORY

- ECC
- Memory Address Parity Protection
- Memory Demand and Patrol Scrub
- Memory Thermal Throttling
- Enhanced DRAM Single Device Data Correction (SDDC)
- Enhanced DRAM Double Device Data Correction (DDDC+1)
- Fine Grained Memory Mirroring
- Memory Sparing
- Memory Migration
- Intel Scalable Memory Interconnect (SMI) Lane Failover
- Intel SMI Clock Failover
- Intel SMI Packet Retry
- Failed DIMM Identification
- Memory Hot Add*

*Requires operating system support.



Micros recentes

✓ IBM Power5 / Power6/ Power7



✓ vários outros

✓ ARM Cortex R Series

✓ SPARC64 V

✓ SUN Niagara II

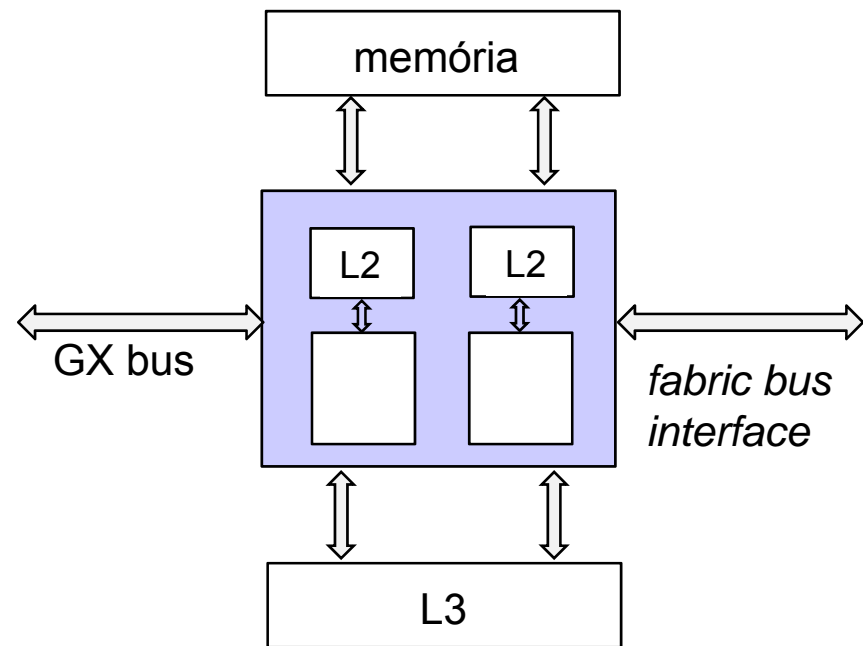
✓ AMD Opteron

✓ Texas Hercules

IBM Power

✓ ECC comum ao Power5 e Power6

- ✓ sinais internos ao chip
- ✓ todos pinos entre o chip e:
 - ✓ L3 cache
 - ✓ memória
 - ✓ GX bus que conecta o chip ao I/O hub
 - ✓ *fabric bus interface* para comunicação entre chips e entre nodos
- ✓ L2 e L3 caches
- ✓ memória



as L1 são internas aos cores e protegidas por **paridade**

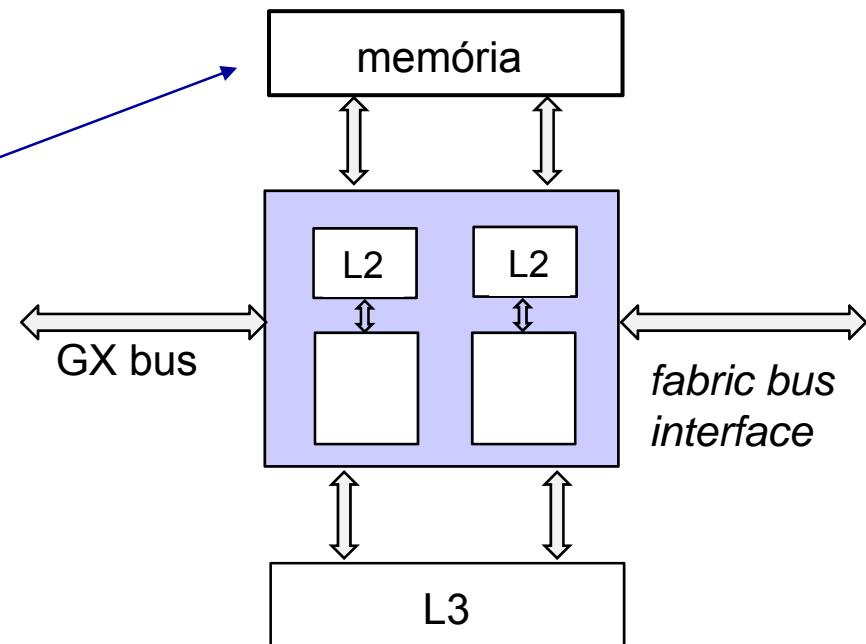


POWER6™
Built on Power™

IBM Power

comum ao Power 5 e 6

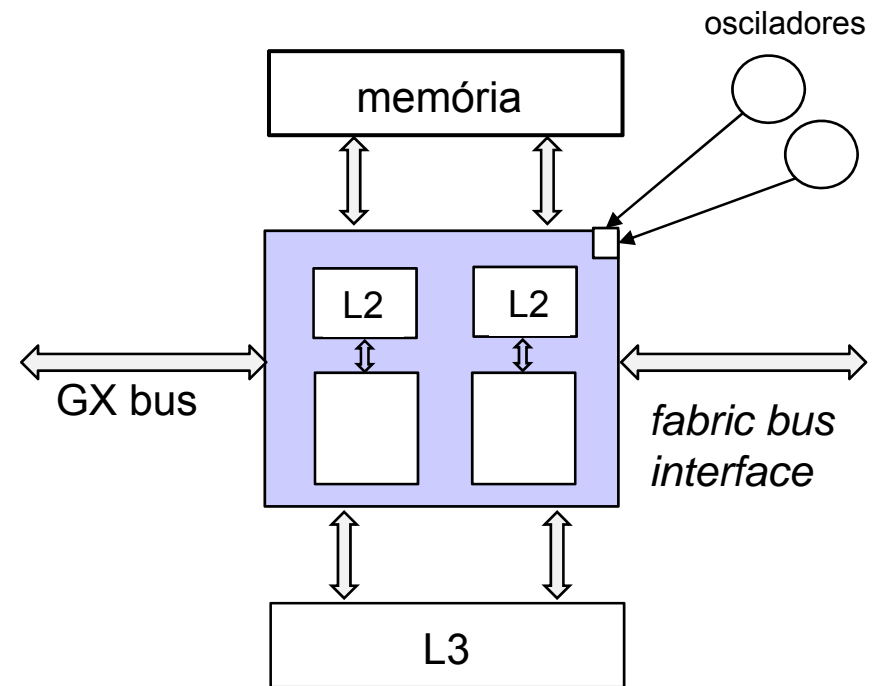
- ChipKill
- *scrubbing* (limpeza) assistida por hardware
- redundância dinâmica e *bit steering* (troca por estepe)
- tratamento para *special uncorrectable error* (SUE)



IBM Power6

novas técnicas no **Power6**:

- reparo dinâmico no barramento de memória,
- failover dinâmico de oscilador (clock)
- recuperação de cache
- recuperação para outro processador
- partition isolation for core checkstops
- *instruction retry* para erros detectados nos cores



IBM Power6 - RU

- ✓ unidade de recuperação - RU

- ✓ faz **checkpoint** do estado do sistema no final da execução de um grupo de instruções
- ✓ array de checkpoints – protegido por ECC
 - ✓ cuidados especiais são tomados para que o checkpoint não registre erros

- ✓ recuperação é realizada para todo o core

- ✓ não apenas para uma thread
- ✓ um pequeno período de tempo após a recuperação, o core fica sem operação em pipeline (slow mode)

Jude A. Rivers and Prabhakar Kudva. Reliability Challenges and System Performance at the Architecture Level. IEEE Design & Test of Computers. 2009. p. 62-72

FAULT-TOLERANT DESIGN OF THE IBM POWER6 MICROPROCESSOR” , K Reick, PN Sanda, S Swaney, JW Kellington, Michael Mack, Michael Floyd e D. Henderson ,publicado na IEEE Micro, 2008, pg. 30 a 38

IBM Power6 - IRR

✓ IRR

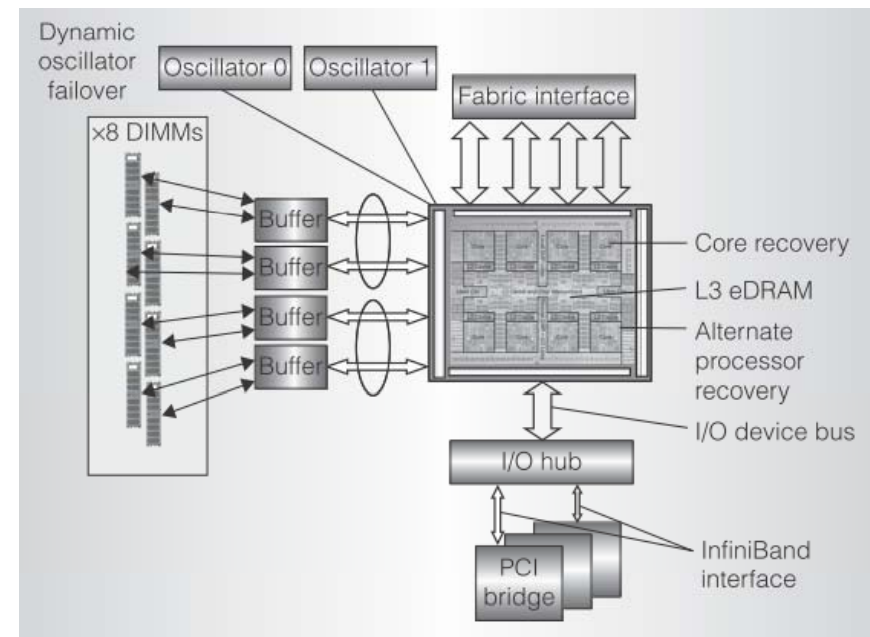
- ✓ instruction retry recovery
- ✓ o RU (recovery unit) dispara o IRR
- ✓ se foi falha transitória, a recuperação é bem sucedida
- ✓ se foi permanente, é escalado um checkstop
 - ✓ no checkstop vai ser feita uma recuperação de alto nível para um core alternativo

Jude A. Rivers and Prabhakar Kudva. Reliability Challenges and System Performance at the Architecture Level. IEEE Design & Test of Computers. 2009. p. 62-72

FAULT-TOLERANT DESIGN OF THE IBM POWER6 MICROPROCESSOR", K Reick, PN Sanda, S Swaney, JW Kellington, Michael Mack, Michael Floyd e D. Henderson ,publicado na IEEE Micro, 2008, pg. 30 a 38

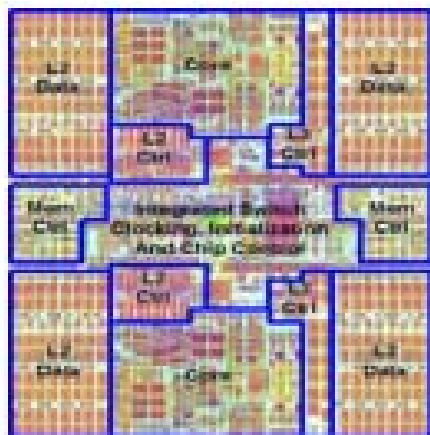
Power 7

- ✓ todas as características RAS do Power 6
- ✓ adicionais
 - ✓ algoritmo ECC 64 bits para a memória
 - ✓ permite a correção de 8 bits (ou seja um chip de memória)
 - ✓ estepe para os chips de buffer de memória
 - ✓ espelhamento de memória seletivo



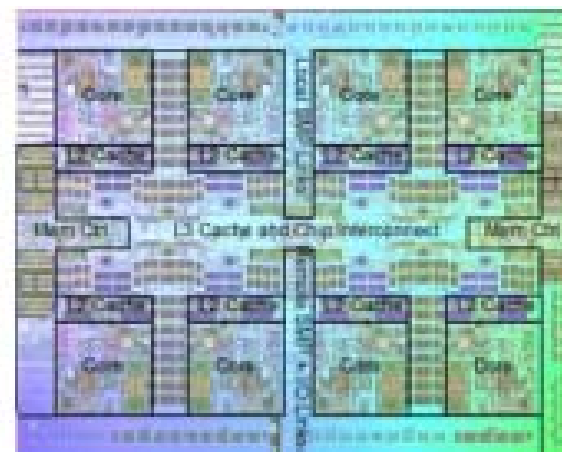
Ron Kalla, B. Sinharoy, W. J. Starke, M. Floyd.
POWER7: IBM'S NEXT-GENERATION SERVER
PROCESSOR. IEEE Micro. 2010.

Power6 e Power7



POWER6 (2007)

- ♦ 65 nm technology – 341 mm²
- ♦ 0.79B transistors
- ♦ 2 Cores
 - 2 SMT threads/core
- ♦ 9 execution units/core
 - 2 integer and 2 binary floating-point units
 - 1 vector and 1 decimal floating-point unit
 - 2 load/store, 1 branch
- ♦ Integrated L2 cache
- ♦ L3 directory & controller (off chip L3 cache)
- ♦ 2 memory controllers



POWER7 (2010)

- ♦ 45nm technology – 567 mm²
- ♦ 1.2B transistors
- ♦ 8 Cores
 - 4 SMT threads/core
- ♦ 12 execution units/core
 - 2 integer and 4 binary floating-point units
 - 1 vector and 1 decimal floating-point unit
 - 2 load/store, 1 branch, 1 condition register
- ♦ Integrated L2 cache
- ♦ Integrated L3 cache
- ♦ 2 memory controllers

Bibliografia

- ✓ capítulos de livros

- ✓ SIEWIOREK, D. Architecture of fault-tolerant computers, cap 2. **Fault-Tolerant System Design**. Prentice Hall, New Jersey, 1996

- ✓ livros

- ✓ SURI, N.; WALTER, C.J.; HUGUE, M.M. **Advances in ultra-dependable distributed systems**. IEEE Computer Society Press. Los Alamitos. 1995.

Bibliografia

✓ artigos

- ✓ AVIZIENIS, A. **Fault Tolerance Infrastructure for Dependable Computing with High-Performance COTS Components**. DSN, IEEE 2000
- ✓ Cristian Constantinescu, **Trends and Challenges in VLSI Circuit Reliability**. IEEE Micro, 2003
- ✓ R. Iyer et al. **Recent Advances and New Avenues in Hardware-Level Reliability Support**, IEEE Micro, vol. 25, pp. 18-29, 2005.
- ✓ K Reick, PN Sanda, S Swaney, JW Kellington, Michael Mack, Michael Floyd e D. Henderson. **Fault-tolerant Design of The IBM Power6 Microprocessor**. IEEE Micro, 2008, pg. 30 a 38
- ✓ Daniel Henderson, Jim Mitchell, and George Ahrens **POWER7 System RAS Key Aspects of Power Systems Reliability, Availability, and Serviceability** . October 3, 2012. IBM Systems and Technology Group

Bibliografia

✓ artigos

- ✓ Ron Kalla, Balaram Sinharoy, William J. Starke, Michael Floyd. **POWER7: IBM'S NEXT-GENERATION SERVER PROCESSOR**. IEEE Micro. 2010. March/ april. 7-15
- ✓ Google Inc.; Schroeder, Bianca; Pinheiro, Eduardo; and Weber, Wolf-Dietrich. "**DRAM Errors in the Wild: A Large-Scale Field Study**." SIGMETRICS/Performance '09, Seattle, WA, June 15-19, 2009.
- ✓ Intel White Paper: Intel® Xeon® Processor E7 Family: Reliability, Availability, and Serviceability - Advanced data integrity and resiliency support for mission-critical deployments. 2011

Arquiteturas tolerantes a falhas

Alta disponibilidade e ultra dependabilidade

Taisy Silva Weber
UFRGS

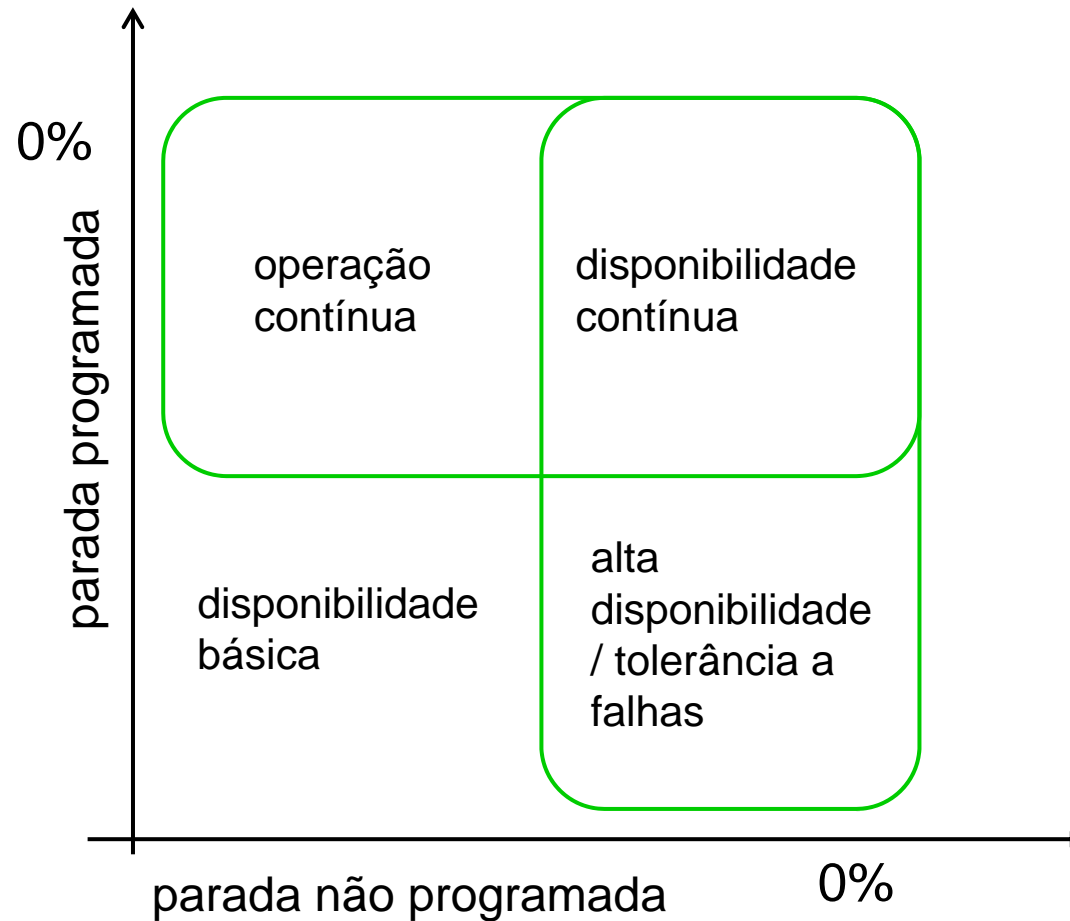
Arquiteturas para alta disponibilidade

- ✓ alta disponibilidade
 - ✓ possível intervalos pequenos para reparo
- ✓ importante
 - ✓ redução do **MTTR**
- ✓ áreas de aplicação
 - ✓ sistemas de transação
 - ✓ redes
 - ✓ telefonia
- ✓ medidas usuais
 - ✓ 7 x 24 (operação contínua)
 - ✓ 5 noves (uptime)

99,999 %

sistemas usados em aplicações em que falhas podem provocar enormes **prejuízos financeiros** como transações bancárias, bolsas de valores, reservas aéreas, e-commerce, transações on-line, ...

disponibilidade: alta X contínua



alta disponibilidade,
disponibilidade contínua
e tolerância a falhas
podem aparecer como
sinônimos, mas alguns
autores fazem distinção

Wendy Bartlett e Lisa Spainhower.
Commercial Fault Tolerance: **A Tale of Two Systems**, IEEE TRANS. ON
DEPENDABLE AND SECURE
COMPUTING, Jan-Mar 2004.

disponibilidade: alta X contínua

Availability Level		Average Yearly Downtime
Conventional	0,99	87 hours, 40 minutes
High Availability	0,999	8 hours, 46 minutes
	0,9995	4 hours, 23 minutes
	0,9999	52 minutes, 36 seconds
Continuous Availability	0,99999	5 minutes, 16 seconds
	0,999999	31.6 seconds

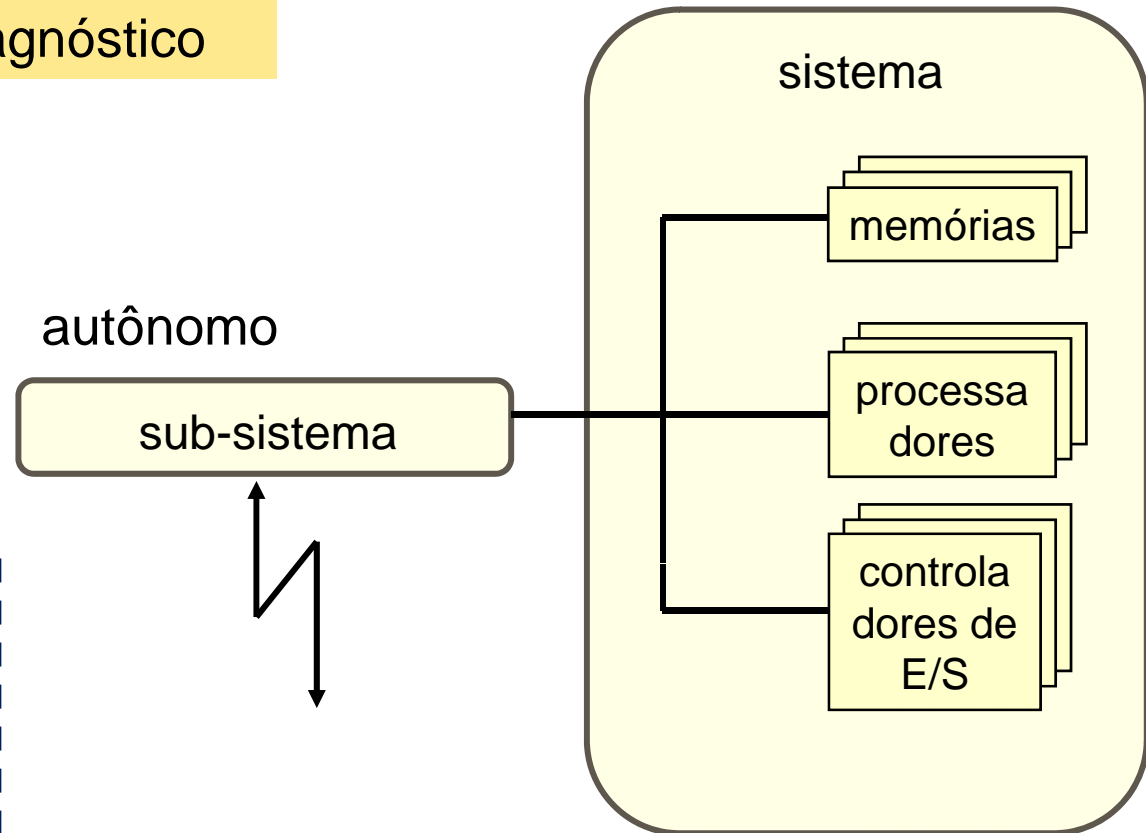
<http://www.stratus.com/About/UptimeMeter.aspx>

sub-sistema de manutenção

supervisão, detecção e diagnóstico

não pode ser o
componente menos
confiável do sistema

não dispensa outras
técnicas de TF:
códigos de correção e
detecção de erros;
recuperação de erros
transitórios



Sistemas comerciais

- ✓ tolerantes a falha

- ✓ computadores de grande porte específicos para aplicações comerciais tolerantes a falhas (sistemas de transações)

- ✓ exemplos:

- ✓ Tandem: fundada em 1976

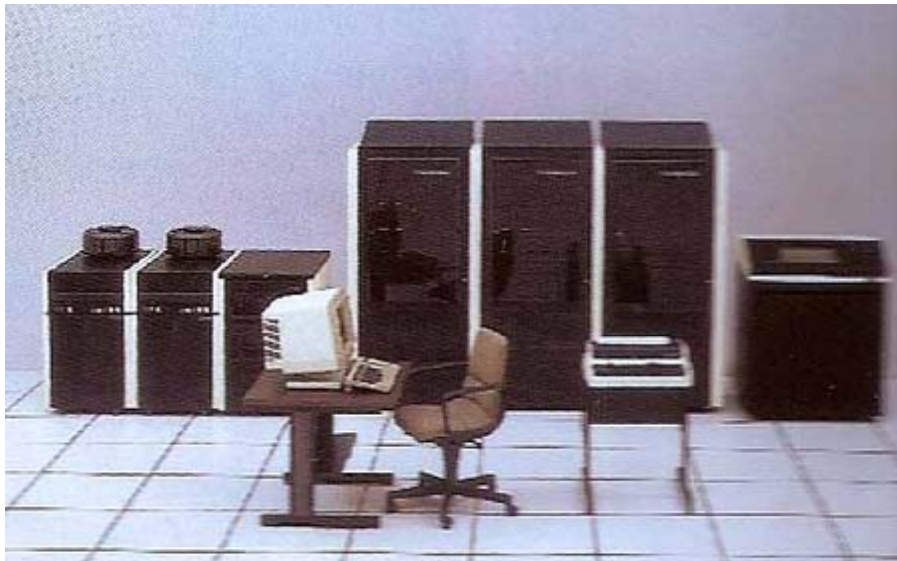
- ✓ mecanismos de TF implementados em software

- ✓ Stratus: fundada em 1980

- ✓ mecanismos de TF implementados em hardware

técnicas ainda usadas para servidores e computadores de alta disponibilidade

NonStop computers



Tandem

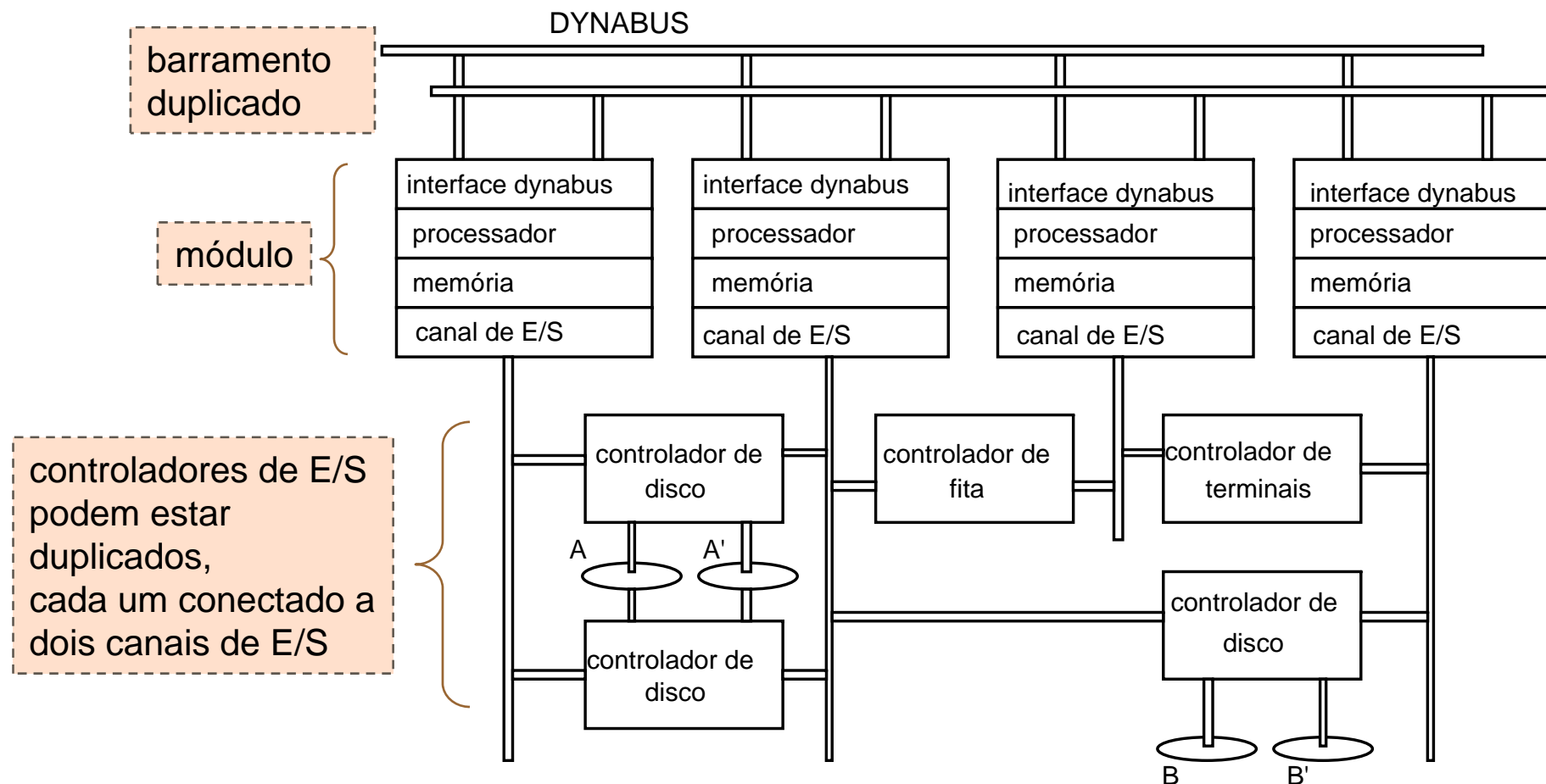


HP



primeiro sistema comercial
modular e escalável projetado
especificamente para
disponibilidade contínua

Tandem NonStop

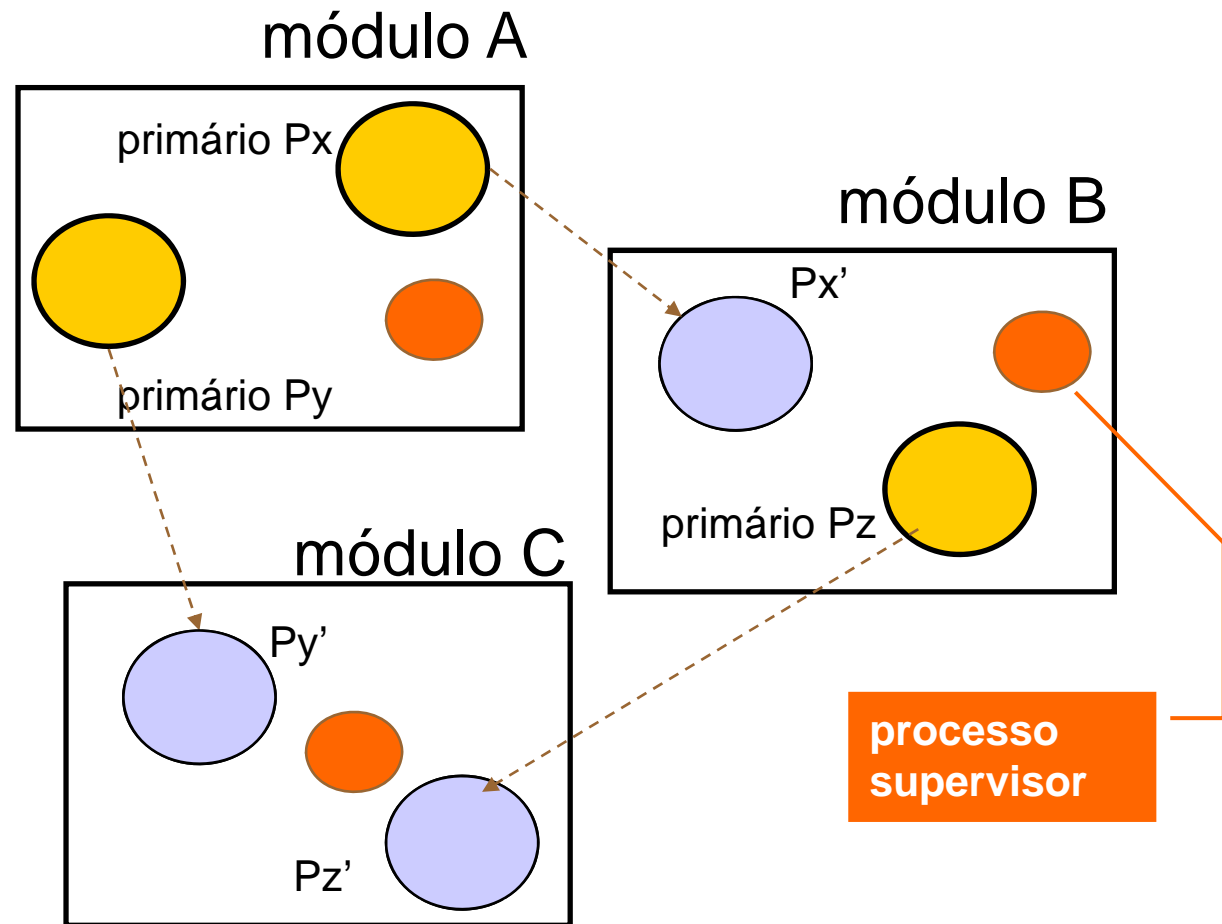


Primários e backups

redundância
dinâmica em
software

pares para
processos do
sistema e do usuário:
processo **primário**
ativo & processo
substituto passivo
(**backup**)

primário envia
pontos de
recuperação para o
backup



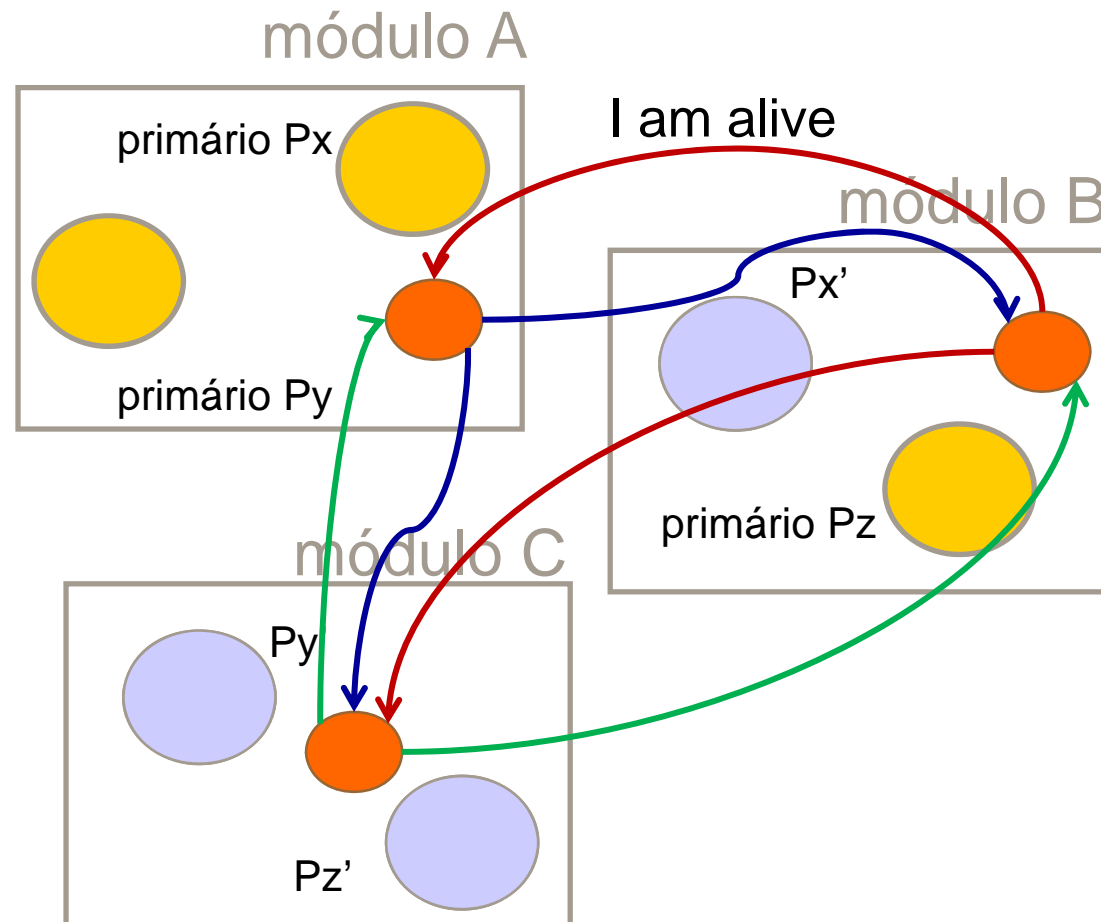
Detecção

assumido *fail-stop*

a cada 1 s: cada supervisor envia **signal de vida** a todos os outros módulos

a cada 2 s: supervisor verifica sinal dos outros módulos

falta de um sinal: **módulo falhou**



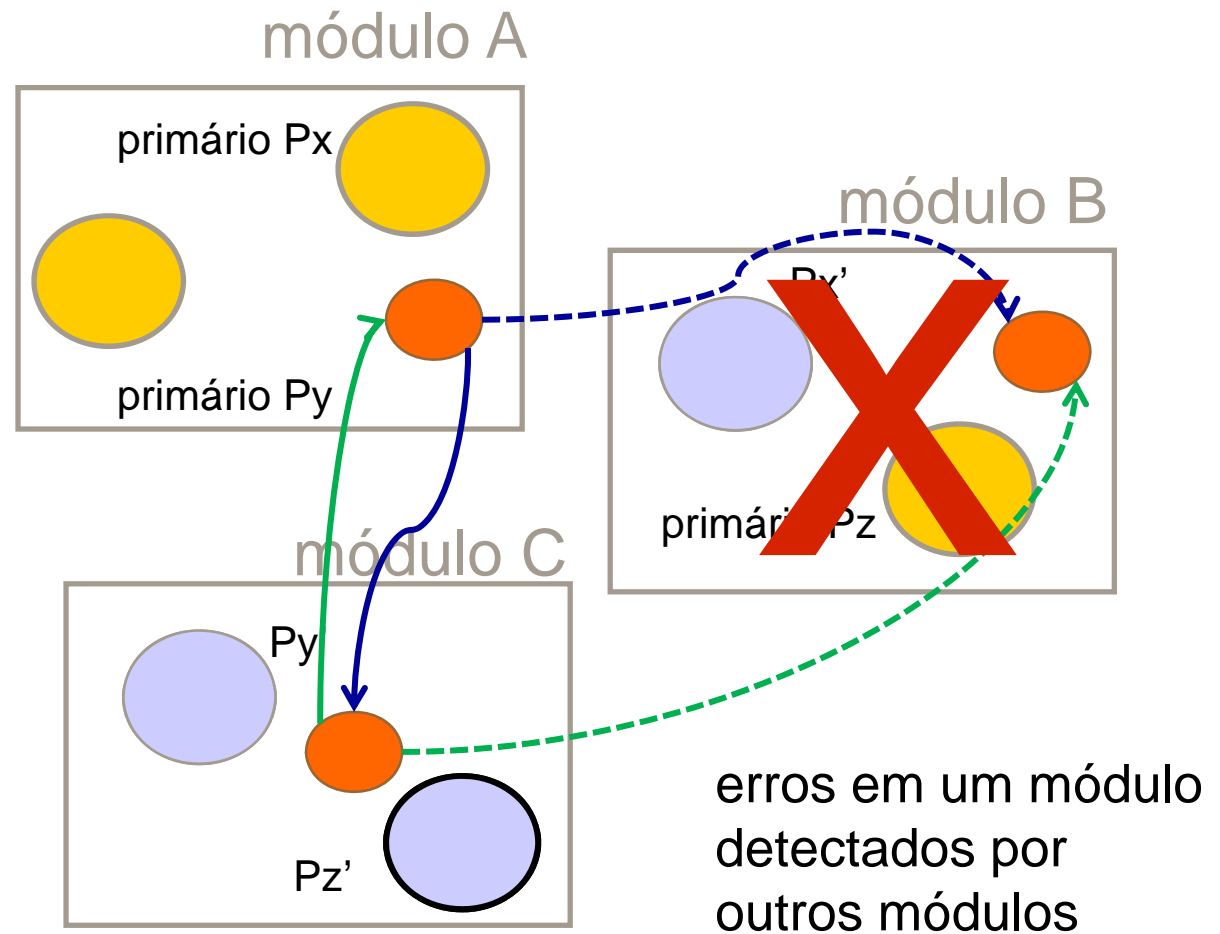
assumido *fail-stop*

Detecção

módulo B parou

supervisores de A e C não recebem sinal de B

supervisores de A e C concluem que B falhou



Tandem: detecção e recuperação

- ✓ operações de entrada e saída

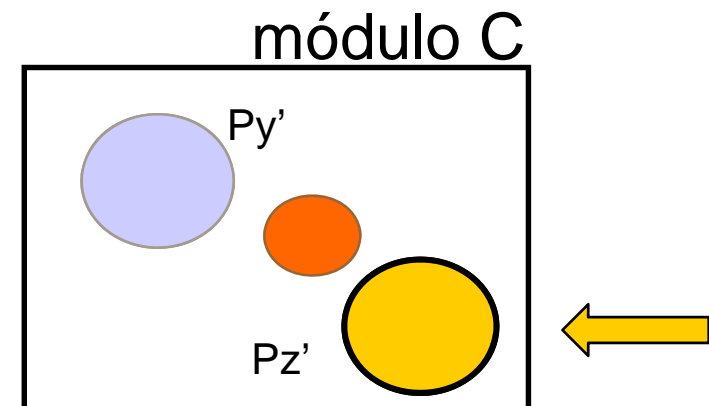
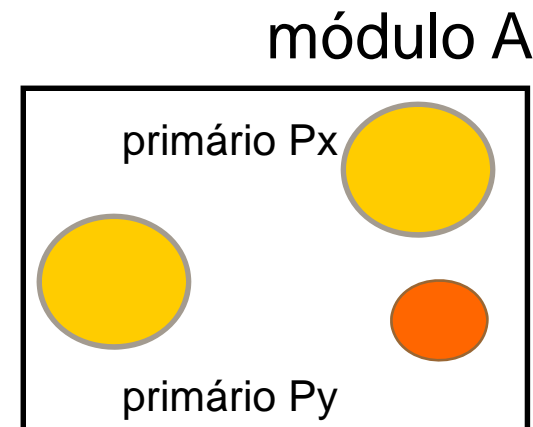
- ✓ detecção por time-out

processo de E/S substituto
entra em operação

- ✓ recuperação

- ✓ processos backup rolam para o último PR e são ativados como primários

- ✓ sistema é reconfigurado
 - ✓ após reparo do módulo novos primários criam substitutos no módulo



Stratus

modelos atuais (2010)



Continuum Series



Stratus® ftServer® 4300 System

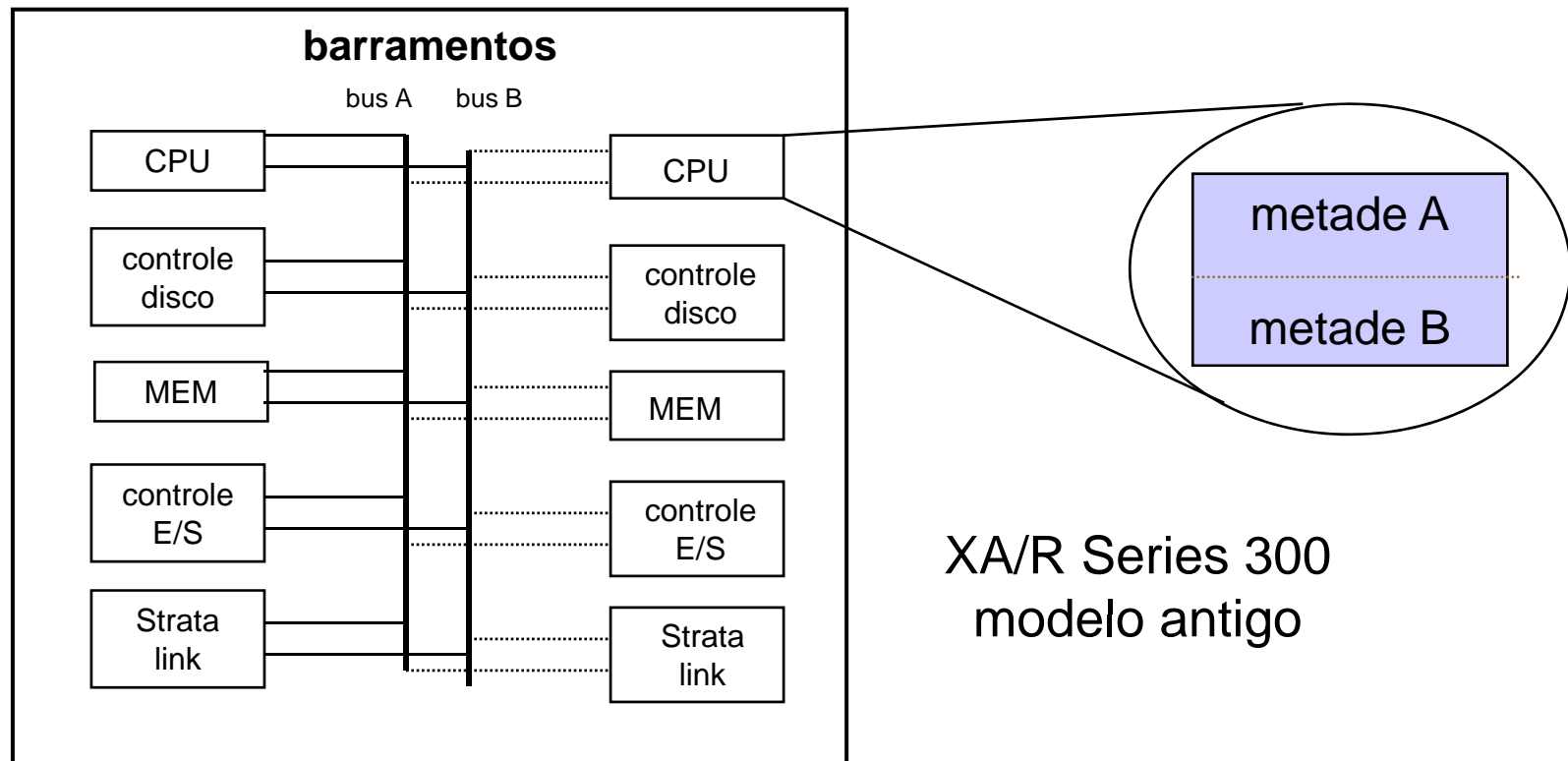


TOTAL AVAILABILITY

abordagem pair-and-spare

Stratus

CPU e memória com componentes duplicados
com comparador (redundância estática)



módulo com 2 boards

XA/R Series 300
modelo antigo

módulos em rede
local (Strata Link)

Servidores HA

- ✓ fornecedores atuais de soluções para servidores de alta disponibilidade

- ✓ Sun (oracle)
- ✓ Dell
- ✓ HP
- ✓ IBM
- ✓ Stratus

pesquisar no site dos fabricantes para descobrir modelos, características, técnicas de TF empregadas e custo

- ✓ evtl outros fabricantes

- ✓ Bull
- ✓ Fujitsu
- ✓ Hitashi



<http://www.stratus.com/About/UptimeMeter.aspx>

- ✓ **reconfiguração automática**
 - ✓ reinicia imediatamente após defeito, isolando automaticamente o componente falho
 - ✓ testa os componentes do sistema
 - ✓ a cada vez que o equipamento é ligado
 - ✓ ou quando é gerada uma interrupção externa
- ✓ fontes de energia e ventilação **redundantes**
- ✓ **monitoramento ambiental**
 - ✓ proteção contra temperaturas extremas, falta de fluxo de ar ou flutuações de energia

- ✓ códigos de detecção e correção de erros
 - ✓ ferramentas de monitoramento
 - ✓ registradores para monitorar o sistema e implementar *cão de guarda*
 - ✓ acesso por *software especial* para extrair dados sobre *desempenho* e para a *manutenção*
 - ✓ ferramentas de diagnóstico *online*
 - ✓ *hot swap*
 - ✓ troca de componentes com o sistema em operação
- módulos de energia/ventilação, placas de CPU/memória e de I/O

Sistemas de telefonia

- ✓ electronic switching systems (ESS)
 - ✓ técnica usual: duplicação e comparação

antigamente eram classificados como críticos, hoje ESS são geralmente classificados como HA ou disponibilidade contínua

sistemas de telefonia são mais antigos que sistemas de computação

início: 1878



Sistemas críticos

ultra dependabilidade

alta dependabilidade

- ✓ primeira área de computadores TF
- ✓ controle de aeronaves e satélites
- ✓ exemplos:
 - ✓ satélite OAO
 - ✓ (Orbiting Astronomical Observatory)
 - ✓ um dos últimos computadores construídos com componentes discretos
 - ✓ Apollo
 - ✓ TMR para processadores
 - ✓ espelhamento de memória e ECC

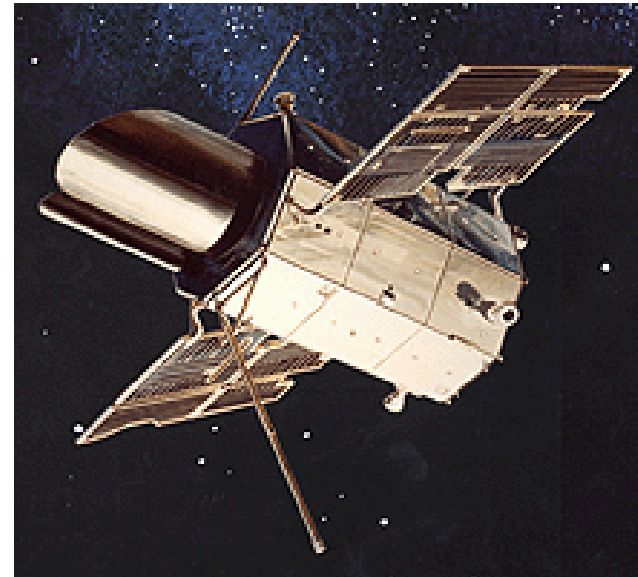
início década de 60

1963 - 1972

Satélites

OAQ - 1

lançado em 8 de abril de 1966;
funcionou perfeitamente
durante 77 minutos;
defeito foi atribuído a falha
no sistema de potência
(fonte)



OAQ - 3

Agosto de 1972 a Fevereiro
de 1981 (9.5 anos)

Fly-by-wire control systems

na NASA, o primeiro avião *fly-by-wire* foi o F-8C Crusader (em 1972)

o A320 foi o primeiro avião comercial *fly-by-wire* digital (em serviço: 1988)

Airbus fly-by-wire: A total approach to dependability. P Traverse, I Lacaze, J Souyris - Building the Information Society, 2004 - Springer

cabine de um airbus A321
(projeto 1989 – primeiro vôo 1993)



Computadores de bordo

ultra dependabilidade

Fly-by-wire control systems

aplicação crítica de tempo real

✓ controle de aeronaves

perda de controle de poucos milisegundos pode ser fatal

- ✓ tempo real
 - ✓ com tempo de atuação curto
 - ✓ interrupção no funcionamento: inadmissível
- ✓ reparo:
 - ✓ possível apenas durante os intervalos de vôo
- ✓ confiabilidade:
 - ✓ da ordem de 10^{-9} ou 10^{-10} falhas por hora para um vôo de 10 horas

1,14 milhões de anos de operação

dois computadores desenvolvidos a partir da mesma especificação

FTMP e SIFT

✓ década de 70 (NASA)

- ✓ redundância modular tripla (TMR)

✓ FTMP

- ✓ Fault Tolerant Multi-Processor

✓ SIFT

- ✓ Software Implemented Fault Tolerance

SRI International, 1972

- ✓ votador implementado em hardware,

- ✓ todos processadores sincronizados

- ✓ relógio central é tolerante a falhas

- ✓ votação por software,

- ✓ processadores são assíncronos,

- ✓ não há relógio central

- ✓ sincronização de resultados para votação garantido por software

SIEWIOREK, D. Architecture of fault-tolerant computers, cap 2.
Fault-Tolerant System Design. Prentice Hall, New Jersey, 1996

- ✓ 5 barramentos redundantes

- ✓ processadores e módulos de memória ligados ao sistema de barramento por interfaces especiais

BGs - bus guardians

- ✓ tríade – 3 processadores + 3 memórias

- ✓ elementos da tríade executam a **mesma tarefa** e comunicam-se através de 3 dos 5 barramentos
 - ✓ BGs votam sobre dados da tríade colocados nos 3 barramentos
 - ✓ **falha mascarada** em processador, memória ou barramento

- ✓ tríades diferentes executam tarefas diferentes
- ✓ processadores e módulos de memória estepe
 - ✓ objetivo:
 - ✓ substituir um elemento de uma tríade que falhou
- ✓ distribuição dinâmica de tarefas entre as tríades
 - ✓ objetivo:
 - ✓ reconhecer falhas nos votadores
- ✓ reconfiguração periódica
 - ✓ objetivo:
 - ✓ reconhecer falhas no mecanismo de reconfiguração

- ✓ módulos processadores interligados por barramento redundante
 - ✓ processadores operam assincronamente em relação aos demais
- ✓ sincronização de resultados para votação por software
- ✓ uma tarefa é alocada sempre a 3 módulos:
 - ✓ cada módulo envia seu resultado aos outros 2 usando o barramento redundante
 - ✓ cada módulo realiza votação majoritária por software

votação majoritária: (2-em-3)

Comparação FTMP e SIFT

- ✓ ambos com **alta confiabilidade** para aplicações críticas tempo-real
- ✓ **FTMP:**
 - ✓ esquema de votação mais eficiente (hardware)
 - ✓ tolerância a falhas não é visível a partir da aplicação
- ✓ **SIFT:**
 - ✓ esquema de votação em software
 - ✓ mais versátil
 - ✓ elimina necessidade de clock tolerante a falhas
 - ✓ tolerância a falhas é visível a partir da aplicação

Space Shuttle



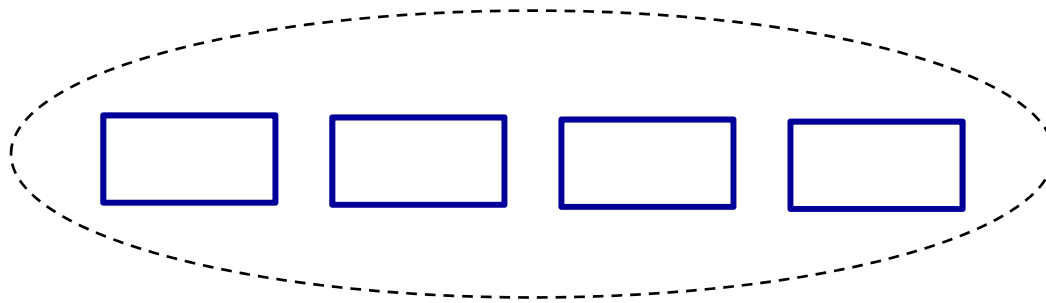
primeiro vôo: 13/04/1981

últimos vôos: 2011

Computers in Spaceflight: The NASA Experience (James E. Tomayko - 1987)
Chapter Four - Computers in the Space Shuttle Avionics System
<http://www.hq.nasa.gov/office/pao/History/computers/Ch4-4.html>

Columbia †
Challenger †
Discovery
Atlantis
Endeavour

Space Shuttle computer



4 computadores atuam nas fases críticas
mascaramento NMR
detecção cruzada (cada computador ouve
todos os demais)



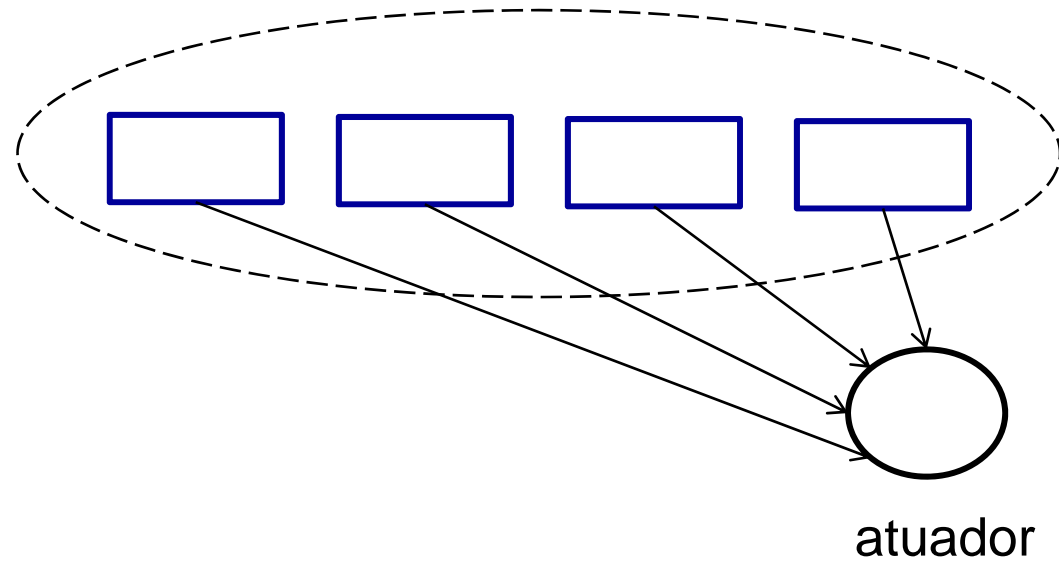
quinto computador realiza
operações não críticas,
atua como backup dos
demais,
software diversitário

suporta até duas falhas

abordagem semelhante a SIFT (1976)

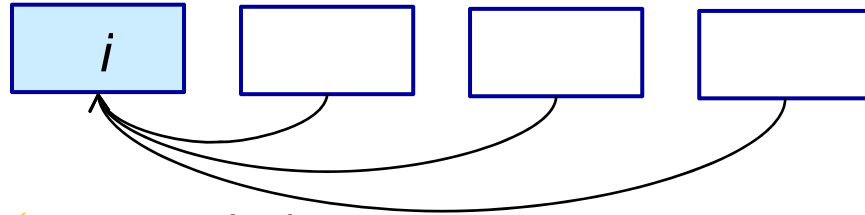
software primário é da IBM, o diversitário da Rockwell

Space Shuttle: operação

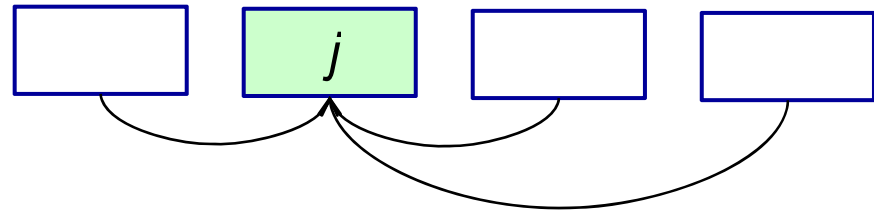


- ✓ mascaramento
 - ✓ saídas são votadas nos atuadores

Space Shuttle: detecção de falha



- ✓ para todo i
 - ✓ i escuta as saídas dos demais
 - ✓ i compara com suas saídas (por software)
 - ✓ se i detectar **desacordo**: avisa o computador suspeito j



- ✓ em cada computador j
 - ✓ sinais de **desacordo** recebidos por j são votados em um circuito especial (**gerente de redundância**)
 - ✓ se o voto for positivo, o suspeito j é desativado

- ✓ controle de superfícies de vôo
 - ✓ controle elétrico, ativação hidráulica
 - ✓ A320/A330/A340
 - ✓ A340 entrou em operação em 2002
 - ✓ controle elétrico, ativação elétrica (*power-by-wire*)
 - ✓ A380, A400M
- ✓ cada computador: canal de comando e canal de monitoramento
 - ✓ cada canal é diferente (diverso) em hardware e software
- ✓ 5 computadores
 - ✓ cada um capaz de pleno controle da aeronave, 3 seriam suficientes para atender safety
- ✓ microprocessadores COTS



norma DO 178B - nível A (software)

computador
com 2 canais

canal de comando

canal de monitoramento

controle
pleno

canal de comando

canal de monitoramento

garantia de
safety

canal de comando

canal de monitoramento

canal de comando

canal de monitoramento

canal de comando

canal de monitoramento

Airbus fly-by-wire: A total approach to dependability. P Traverse, I Lacaze, J Souyris - Building the Information Society, 2004 - Springer

Bibliografia

✓ capítulos de livros

- ✓ SIEWIOREK, D. Architecture of fault-tolerante computers, cap 2. **Fault-Tolerant System Design**. Prentice Hall, New Jersey, 1996

✓ artigos

- ✓ AVIZIENIS, A. **Fault Tolerance Infrastructure for Dependable Computing with High-Performance COTS Components**. DSN, IEEE 2000
- ✓ Cristian Constantinescu, **Trends and Challenges in VLSI Circuit Reliability**. IEEE Micro, 2003
- ✓ Wendy Bartlett e Lisa Spainhower. Commercial Fault Tolerance: **A Tale of Two Systems**, IEEE TRANS. ON DEPENDABLE AND SECURE COMPUTING, Jan-Mar 2004.
- ✓ K Reick, PN Sanda, S Swaney, JW Kellington, Michael Mack, Michael Floyd e D. Henderson .**FAULT-TOLERANT DESIGN OF THE IBM POWER6 MICROPROCESSOR**. IEEE Micro, 2008, pg. 30 a 38
- ✓ P Traverse, I Lacaze, J Souyris. **Airbus fly-by-wire: A total approach to dependability** - Building the Information Society, 2004 - Springer

✓ livros

- ✓ SURI, N.; WALTER, C.J.; HUGUE, M.M. **Advances in ultra-dependable distributed systems**. IEEE Computer Society Press. Los Alamitos. 1995.

Segurança funcional

Integridade de segurança

Taisy Weber

?



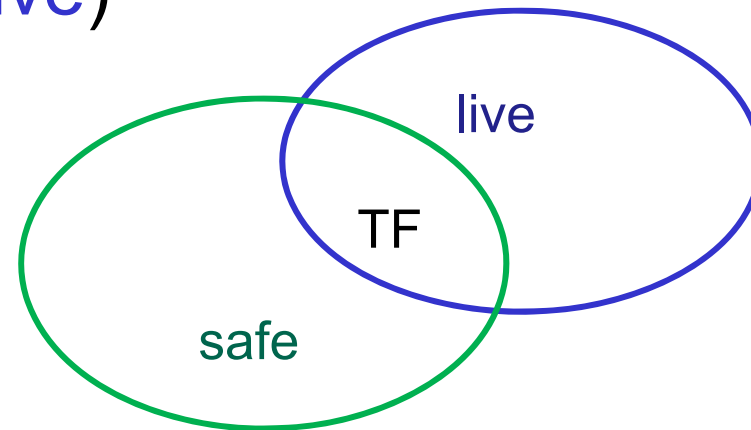
safety

não relacionado a **security**

- garantia de que uma “coisa ruim” jamais vai acontecer
 - por exemplo: elevador despencar
- não garante que um **coisa boa** possa acontecer (vivacidade – liveness)
 - por exemplo: elevador se movimentar

safety

- sistema tolerante a falhas deve ser seguro (**safe**) e vivaz (**live**)



- termo “safety”
 - enfatizado também em outras áreas da computação (eng. software, especificação formal)
 - muito usado na área de controle de processos

safety versus dependabilidade

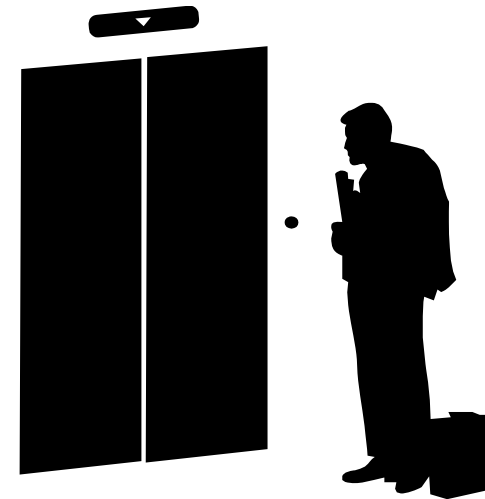
- safety é um **atributo** de dependabilidade
 - aparece frequentemente com a medida da capacidade *fail-safe* do sistema
 - relacionado a *detecção de falhas*

um sistema seguro não apresenta defeito:
ou está fornecendo as saídas corretas ou está
em um estado seguro

a medida numérica (probabilidade)
indica o quão próximo o sistema está
do ideal

fail-safe versus fail-operate

- muitos sistemas reais são *fail-safe*
 - ou seja, possuem um estado **não operacional** seguro e fácil de alcançar em caso de falha interna
- exemplos
 - elevador
 - sinais de trânsito
 - máquina de corte
 - escadas rolantes



fail-safe versus fail-operate

- **mas** muitos outros sistemas reais não podem parar em caso de falha
 - aviões no ar
 - processos contínuos que não podem ser subitamente interrompidos
- são chamados *fail-operate*
 - ou seja, tolerantes a falhas



sistemas críticos

- sistemas críticos exigem segurança
 - são denominados *safety-critical*
- money-critical
 - prejuízos a propriedade
 - ou a economia
- life-critical
 - prejuízos a vida humana
 - ou ao ambiente do qual dependemos

exemplo: acidentes ambientais

problemas podem ser graves quando computadores sujeitos a apresentar defeitos são usados no controle desses sistemas

exemplos de sistemas críticos

- controle do espaço aéreo
- controle de usinas nucleares
- controle de segurança em plataformas de petróleo

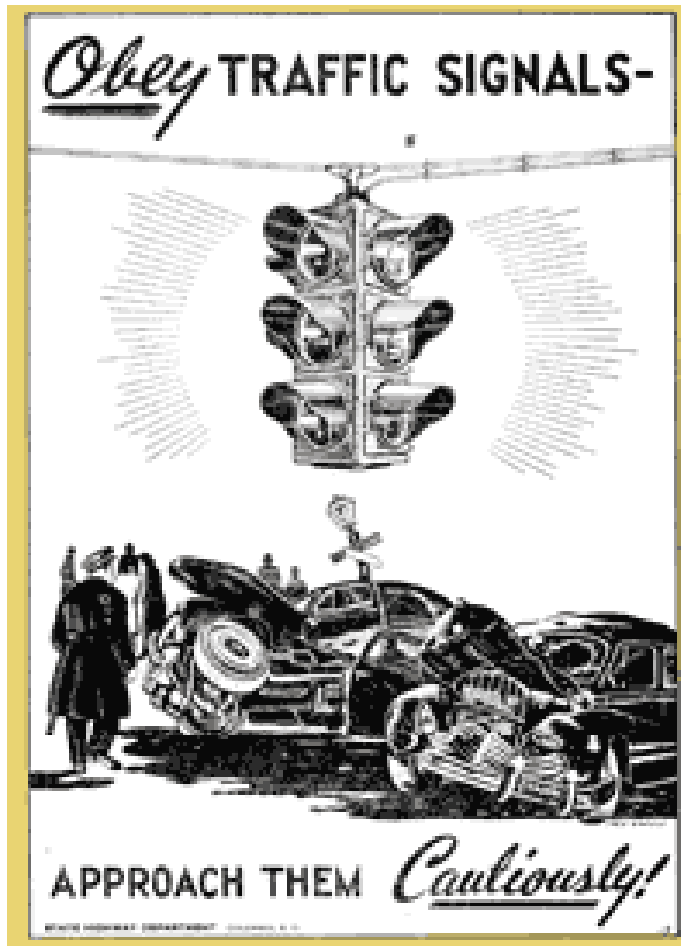


exemplos de sistemas críticos

- controle de vazamento de gases tóxicos
- controle de processos químicos
- sistemas de suporte de vida em hospitais



exemplos de sistemas críticos



- controle de transporte terrestre (trens, metrô subterrâneo e de superfície)
- sistemas embarcados (ex: automóveis)
- controle de robôs

fracassos

- causas de projetos mal sucedidos
 - conhecimento incompleto do que torna um sistema seguro
 - desconhecimento do sistema geral onde o controlador ou computador vai atuar
 - ignorância dos pontos de defeito críticos (*single-points-of-failure*) do sistema
- impossível sistema perfeito
 - falhas são inevitáveis
 - mas os riscos podem ser reduzidos consideravelmente

hazards e riscos

- *hazards* ameaça, perigo
 - fonte potencial de perigo
 - análise de hazards e estimativa dos riscos
- riscos
 - danos ao negócio, a pessoas, à propriedade
 - riscos:
 - inaceitáveis
 - aceitáveis (ou toleráveis)

hazards e riscos

Exemplos de riscos:

acidentes em estradas	100	cpm
dirigir um carro	150	cpm
queda de aviões	0.02	cpm
mordida de inseto ou cobra	0.1	cpm
fumar 20 cigarros por dia	5000	cpm

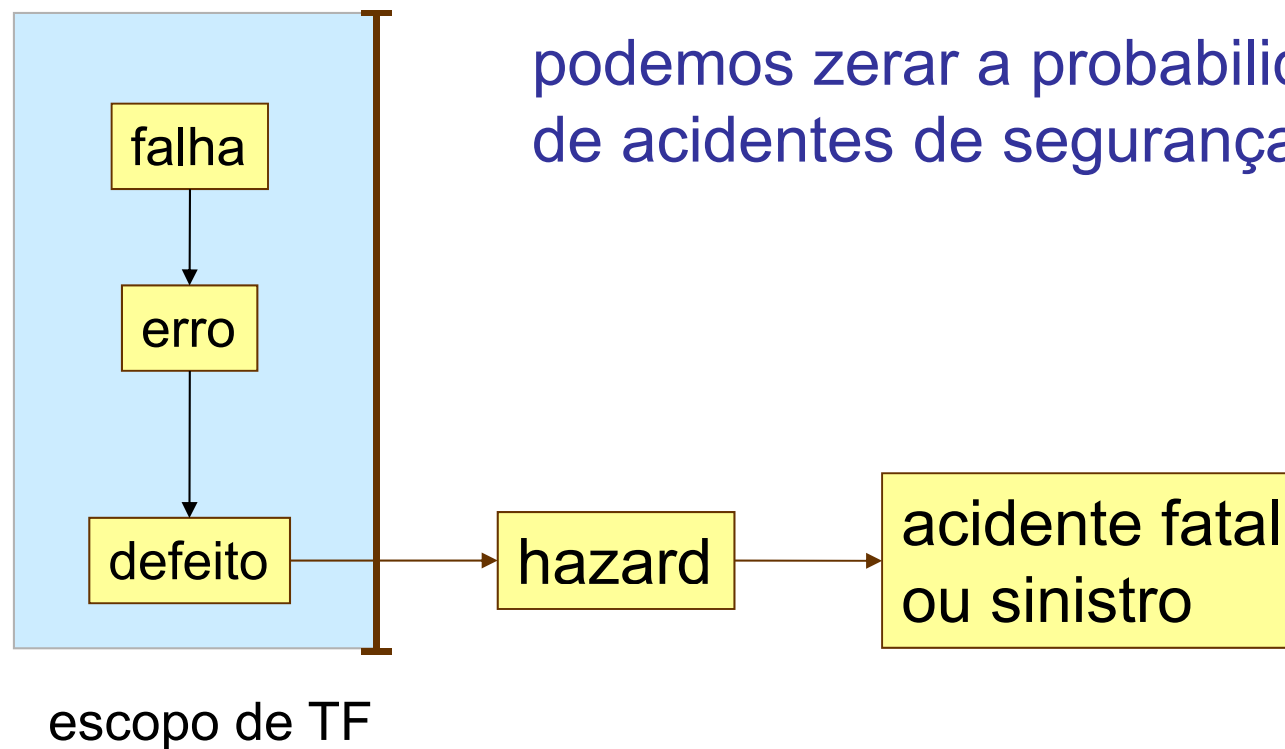
cpm - chances por
milhão por ano

hazards

- ameaça, perigo (*hazard*)
- definida como alguma condição real ou potencial que pode causar
 - ferimento, doença, ou morte de pessoas;
 - danos a (ou perda de) um sistema, um equipamento ou uma propriedade;
 - danos ao ambiente



hazards x falhas



elementos associados ao risco

estimativa de risco

- risco de que?
 - consequência ou severidade
- quanto provável é o risco?
 - frequência ou probabilidade de ocorrência por unidade de tempo
 - probabilidade de ser morto por raio é 10^{-7} por ano;
 - probabilidade de um avião cair durante aterrissagem devido a defeito nos instrumentos é $< 10^{-7}$ por aterrissagem.

riscos baixos são frequentemente ignorados;
riscos a longo prazo são frequentemente ignorados

elementos associados ao risco

- exemplo:
 - defeito de uma válvula química pode resultar em uma explosão que pode matar 10 pessoas,
 - se esse componente falhar **uma vez a cada 10.000 anos**, qual o **risco** associado a esse componente?
- solução:
 - **taxa de defeito** da válvula é 0,0001 defeitos por ano.
 - então, o risco de uma fatalidade é $10 \times 0,0001 = 0.001$ mortes por ano.

pergunta: é aceitável ou não?
a resposta não é técnica

risco

- risco aceitável
 - deve fazer parte da especificação do sistema
 - variação de 10^{-2} a 10^{-10} incidentes por hora
 - geralmente definido por leis ou órgãos reguladores no interesse da população
- normas
 - **várias** normas estabelecem padrões de segurança para diferentes setores
 - médico-cirúrgico
 - nuclear
 - transportes, ...



<http://www.iec.ch/>

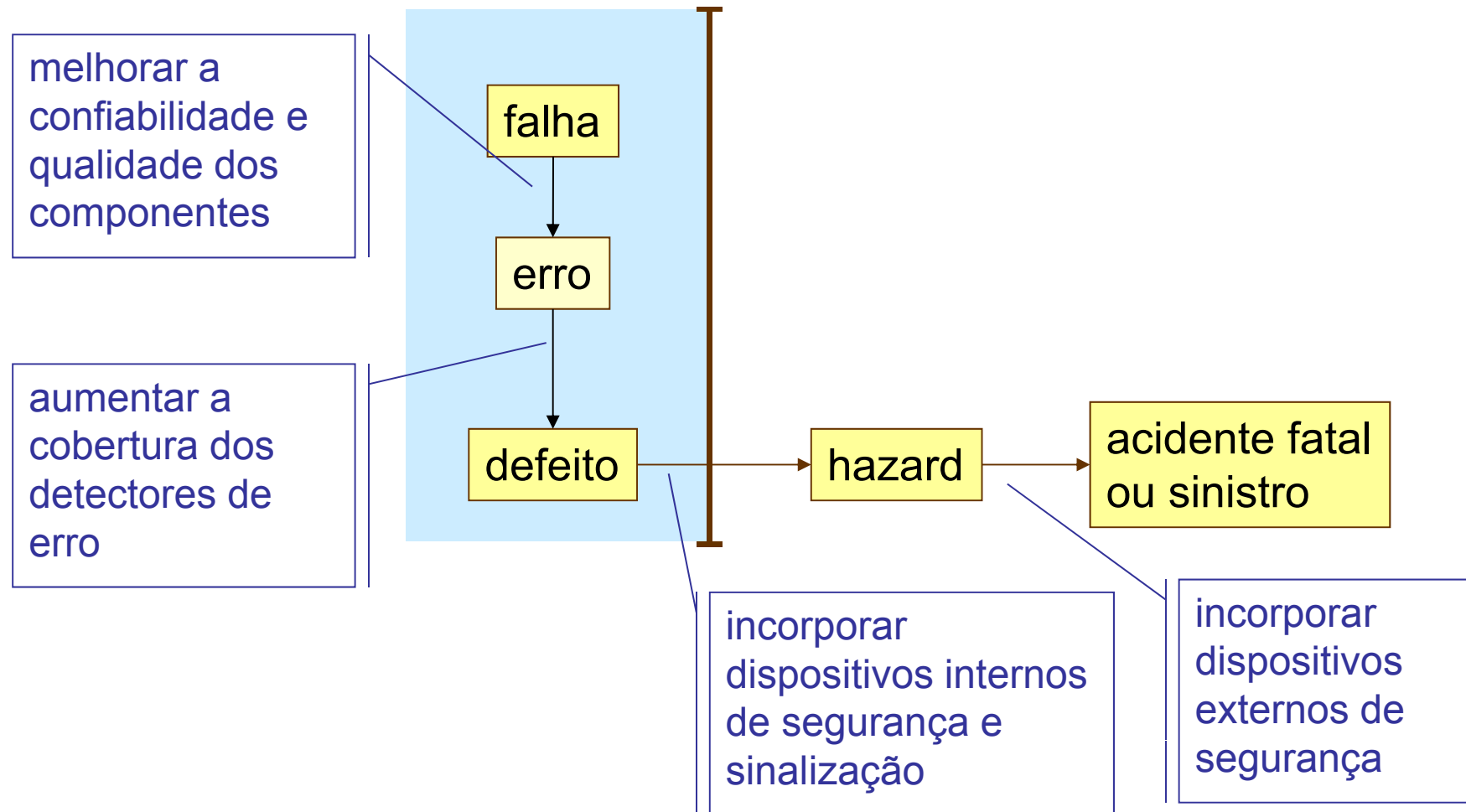
normas de segurança

- exemplos:
 - IEC 61508 (industrial)
 - Mil-Std-882D (militar)
- normas relacionadas
 - IEC 61511 (process industry)
 - IEC 62061 (machinery control systems),
 - IEC 61513 (nuclear power plants),
 - IEC 60601 (medical equipment),
 - ISO/CD 26262 (automotive systems)



<http://www.iec.ch/>

redução de risco



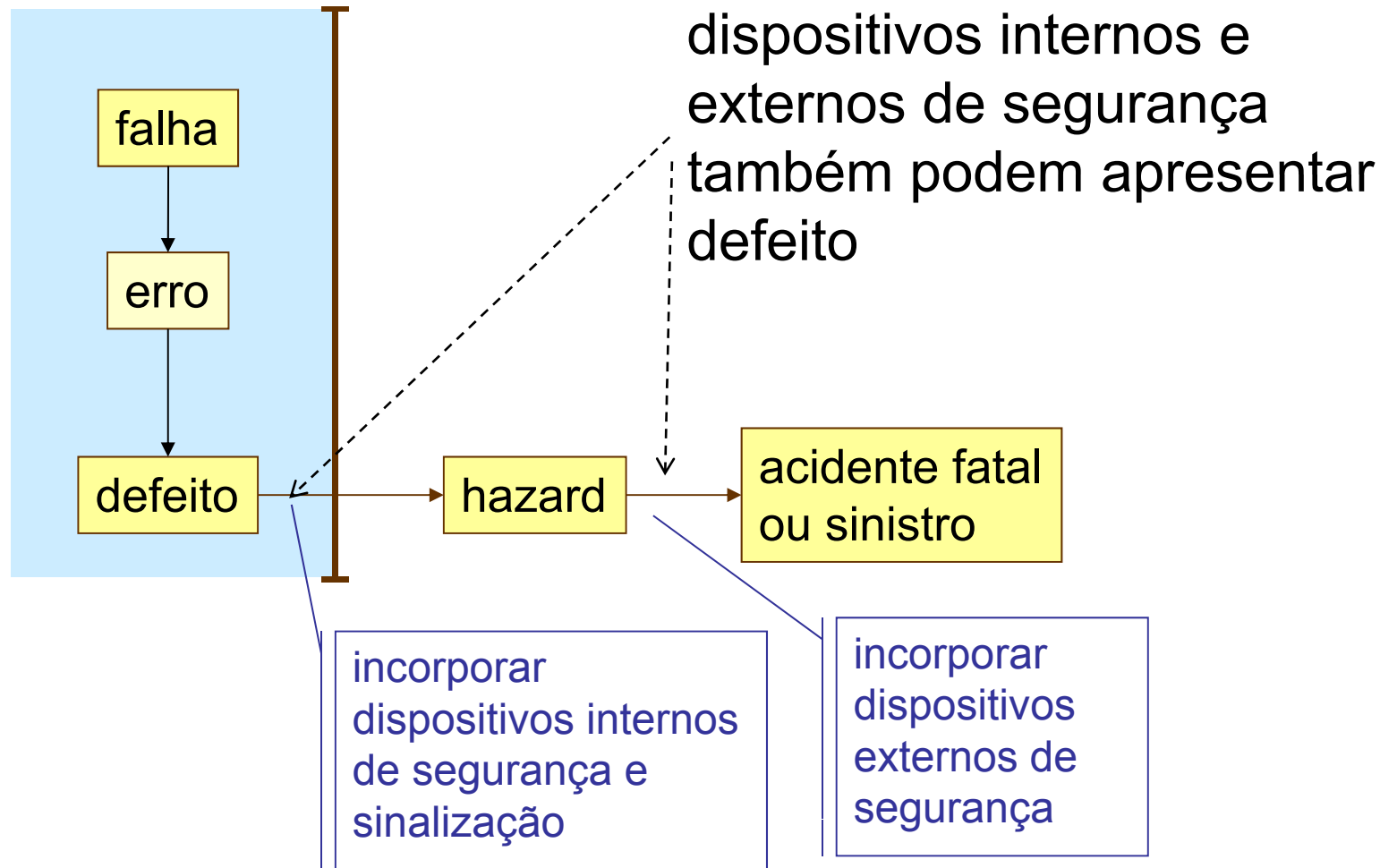
confiabilidade e safety



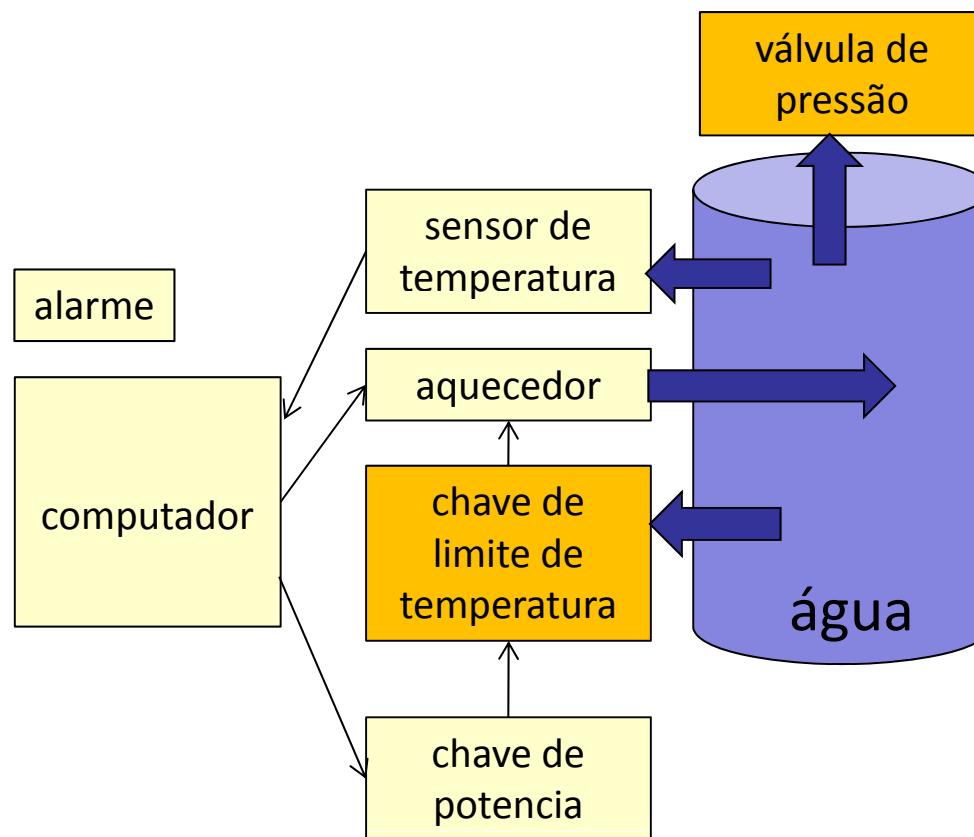
componente altamente
confiável
implementando função
de segurança

"Larry here is one of our most reliable men.
He's in charge of safety!"

dispositivos de segurança



exemplo



hazards:
sobreaquecimento da água
e explosão do tanque

defeitos:
devidos a falhas no sensor,
no aquecedor, no
computador, no software,
no operador

dispositivos de segurança

- devem ser construídos com componentes confiáveis
 - baixa taxa de defeito
- devem possuir baixa probabilidade de falha
 - se forem dispositivos eletrônicos programáveis, devem ser **tolerantes a falhas**
 - um defeito em um dispositivo de segurança não deve provocar dano ao ambiente ou usuários
- devem também ser seguros
 - provar sua **integridade de segurança**

norma de segurança?



**“Do you carry
safety books for dummies?”**

- norma para segurança funcional

- independente do domínio da aplicação
- equipamentos elétricos, eletrônicos e eletrônicos programáveis

E/E/PE *electrical/electronic/ programmable electronic*

- **sugere** ou **impõe** técnicas, estratégias e métodos para o projeto de sistemas seguros
- usada em aplicações críticas (principalmente sistemas de segurança, mas pode ser usada em sistemas de controle)
- permite **certificação** por organizações internacionais

safety e segurança funcional

- safety (no contexto da norma IEC)
 - sistemas livres de riscos inaceitáveis envolvendo prejuízos físicos ou danos à saúde de pessoas resultantes direta ou indiretamente de danos a propriedades ou ao ambiente

obs: ênfase nos riscos inaceitáveis

- *segurança funcional (functional safety)*
 - é parte da segurança global de um sistema
 - depende do sistema ou equipamento operar corretamente em resposta a suas entradas

essa definição não implica em estado seguro

segurança funcional x fail safe

- a norma considera o conceito de *fail-safe* inadequado para sistemas computacionais complexos
 - *fail-safe* só indicado para sistemas de baixa complexidade com modo de defeito bem definido
 - a norma visa um escopo maior de sistemas (fail-operate)
 - *fail-operate* = tolerante a falhas

na prática os estados seguros precisam ser identificados e defeitos seguros separados de perigosos

IEC 61508: objetivos

- permitir usar a tecnologia para aumentar segurança sem prejudicar desenvolvimento futuro,
 - aproveitamento do potencial da tecnologia E/E/PE para melhorar a segurança e aspectos econômicos
 - desenvolvimento tecnológico considerando segurança global
- estabelecer **padrão genérico**,
- prover meios para **usuários e reguladores** terem confiança em controladores computacionais
- abordagem flexível e baseada em **risco presumido**

IEC 61508: abordagem

- dois conceitos fundamentais
 - ciclo de vida de segurança (SLC)
 - análise probabilística de falhas para quantificar nível de integridade de segurança (SIL)
- ciclo de vida
 - reduzir ou eliminar defeitos devido a erros sistemáticos
- cálculo de SIL
 - determinação de probabilidade de defeitos randômicos

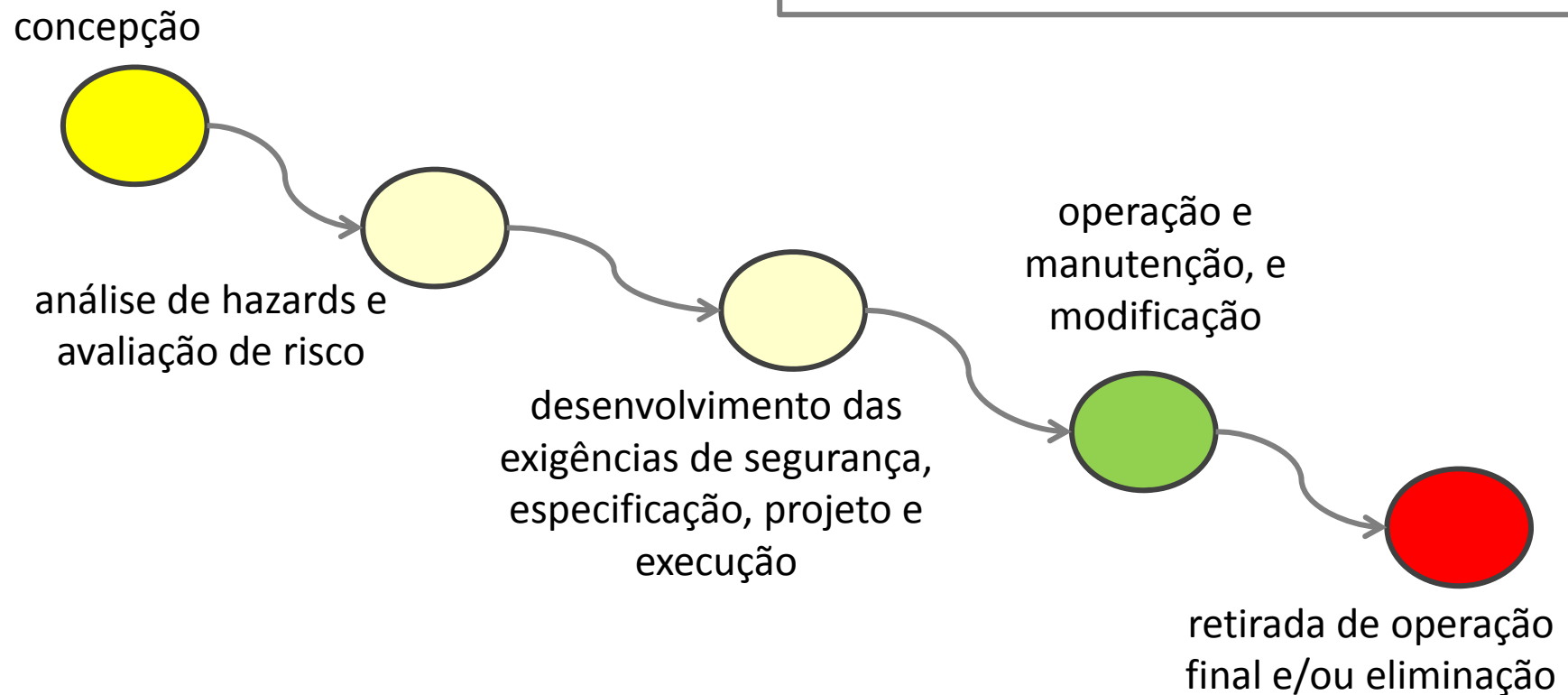
IEC 61508: ciclo de vida

- modelagem do ciclo de vida
 - a norma cobre todas as atividades do ciclo de vida de safety:
 - concepção,
 - análise de hazard e avaliação de risco,
 - desenvolvimento das exigências de segurança, especificação, projeto e execução, operação e manutenção, e também modificação,
 - retirada de operação final e/ou eliminação.
- compreende aspectos do sistema e de mecanismos de tratamento de falhas

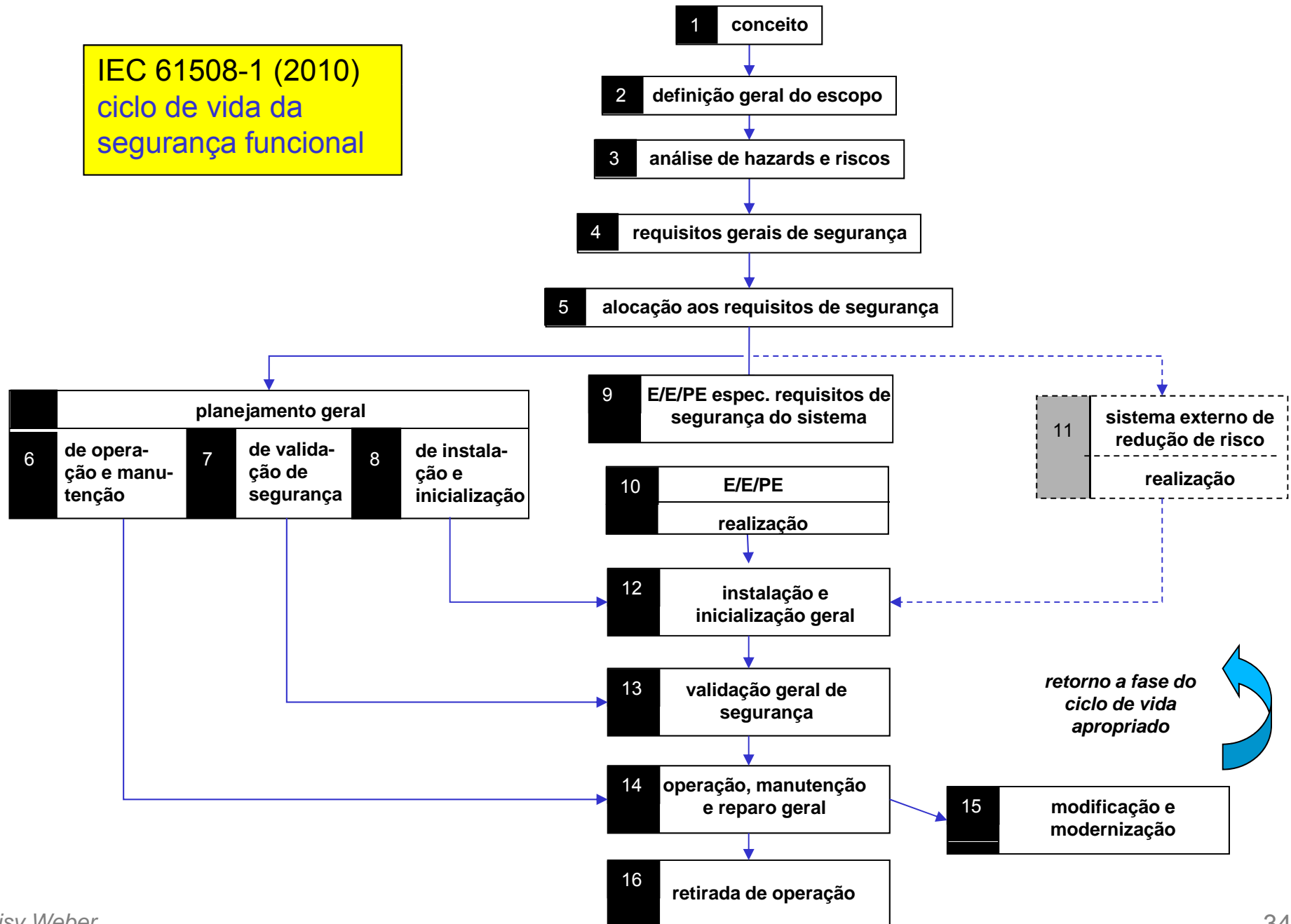
IEC 61508: características

- modelagem do ciclo de vida de safety

compreende aspectos do sistema e de mecanismos de [tratamento de falhas](#)



IEC 61508-1 (2010)
ciclo de vida da
segurança funcional

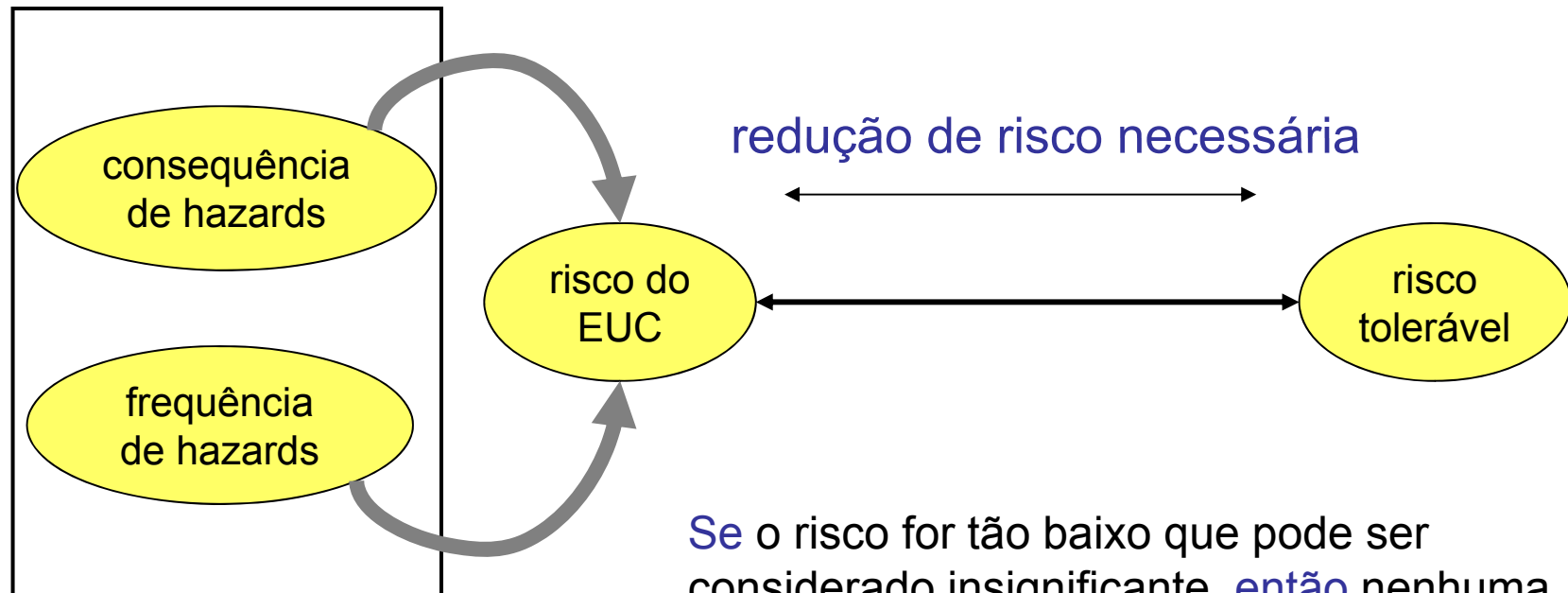


IEC 61508: características

- prevenção e controle de defeitos
 - exigências para impedir defeitos
 - evitar a introdução de falhas - prevenção
 - exigências para controlar defeitos
 - garantir segurança mesmo quando falhas estão presentes
- técnicas e medidas
 - a norma especifica as técnicas e as medidas que são necessárias para conseguir a integridade de segurança requerida

tolerância a falhas

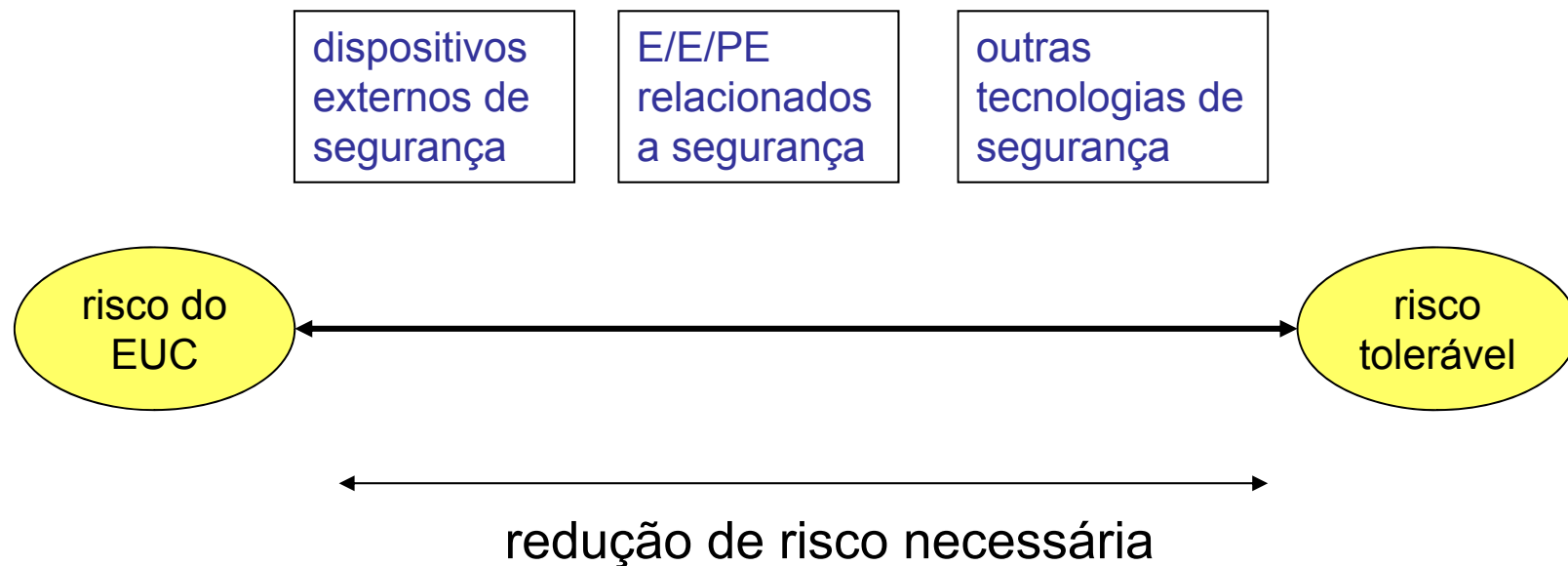
IEC 61508 – redução de risco



EUC (equipamento sob controle) e controle do EUC

Se o risco for tão baixo que pode ser considerado insignificante, **então** nenhuma ação adicional é requerida
Se não for o caso, **então** características de segurança devem ser incorporadas no projeto para reduzir o risco a um nível tolerável para a aplicação.

IEC 61508 – redução de risco



sistemas relacionados a segurança

na literatura técnica: **SIS** – sistemas instrumentados de segurança

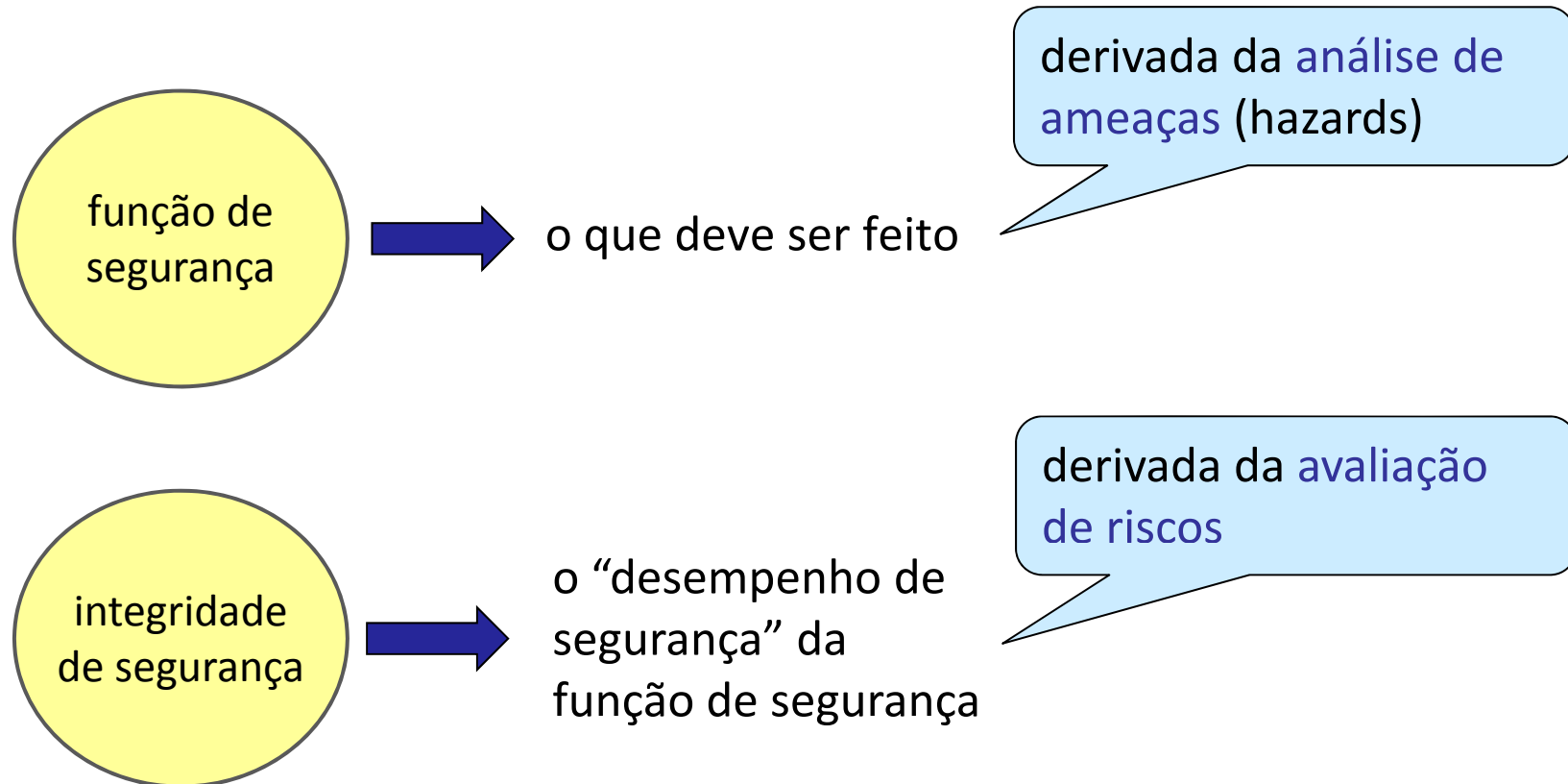
- aplicados para garantir que os riscos serão mantidos em um nível aceitável
- executam funções de segurança
- qualquer sistema que realiza funções de segurança é um SIS
 - pode estar integrado a equipamentos ou sistemas de controle (**interno**)
 - pode estar separado deles
 - dispositivo **externo** de segurança

funções de segurança

- requisitos para as funções de segurança
 - o que a função de segurança efetivamente deve realizar
 - derivados da **análise de ameaças** (hazards)
- integridade de segurança
 - relacionado a **probabilidade** que a função de segurança seja executada satisfatoriamente (sob condições definidas e durante um dados período de tempo)
 - requisitos derivados da **estimativa de riscos**

o conceito de **integridade** aparece também na literatura como a capacidade de detecção de falhas de um sistema

funções de segurança vs integridade



Exemplo

- função de segurança
 - para prevenir a ruptura de um reservatório X sob pressão, a válvula Y deve abrir em 2 segundos quando a pressão no reservatório alcançar 2.6 bar
- integridade de segurança
 - a integridade de segurança da função de segurança deve ser SIL 2

Bell, R.: *Introduction to IEC 61508*. In Proc. of the 10th Australian Workshop on Safety Critical Systems and Software, 3-12. (2006)

funções de segurança

- funções de segurança executadas por sistemas eletrônicos programáveis

CLPs e computadores

- alta complexidade
 - falhas e erros que levam a defeitos
 - impossível determinar com precisão o modo de defeito
 - impossível teste exaustivo
- desafio:
 - projetar sistemas que previnem defeitos ou controlam falhas (ou erros) quando eles aparecem

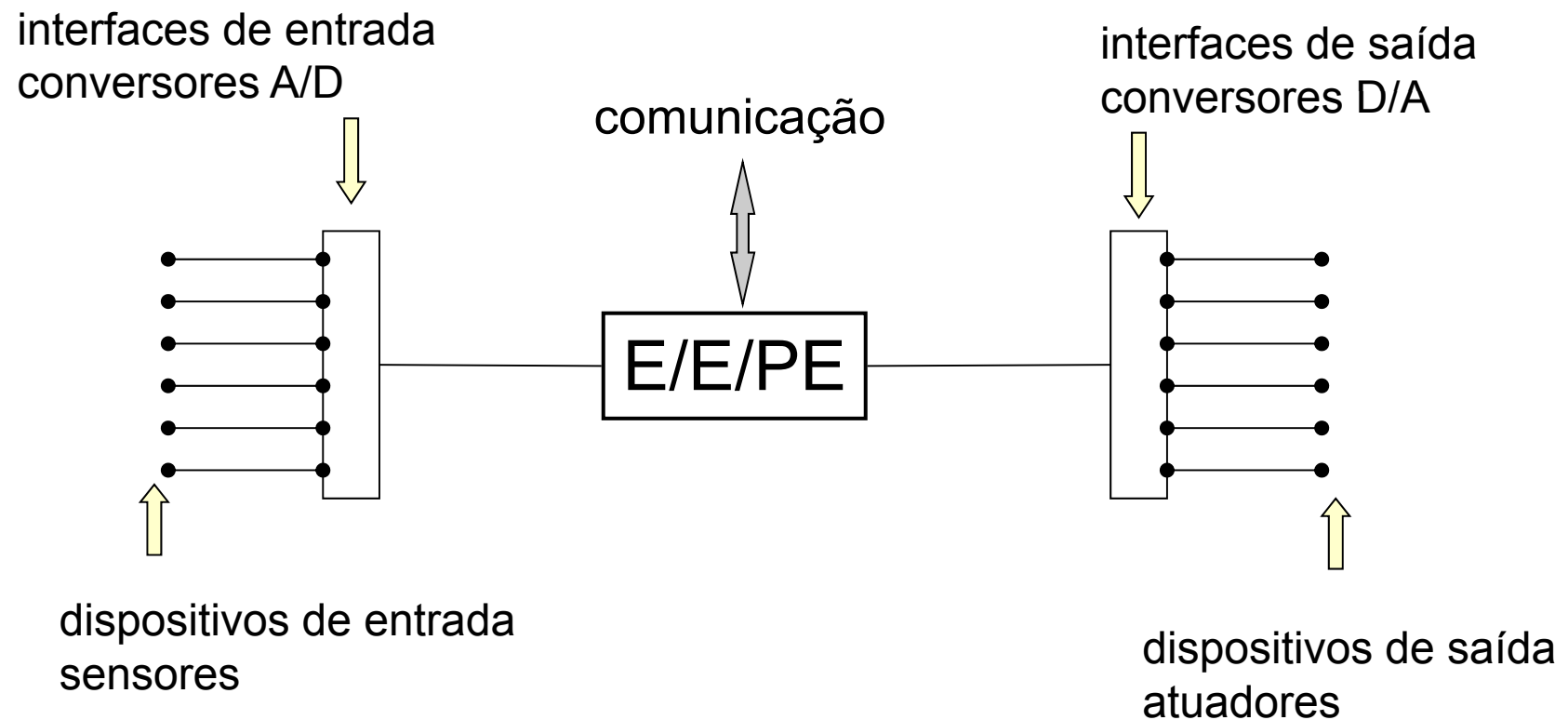
aqui entra-se na área de dependabilidade

exemplos de sistemas E/E/PE relacionados a segurança

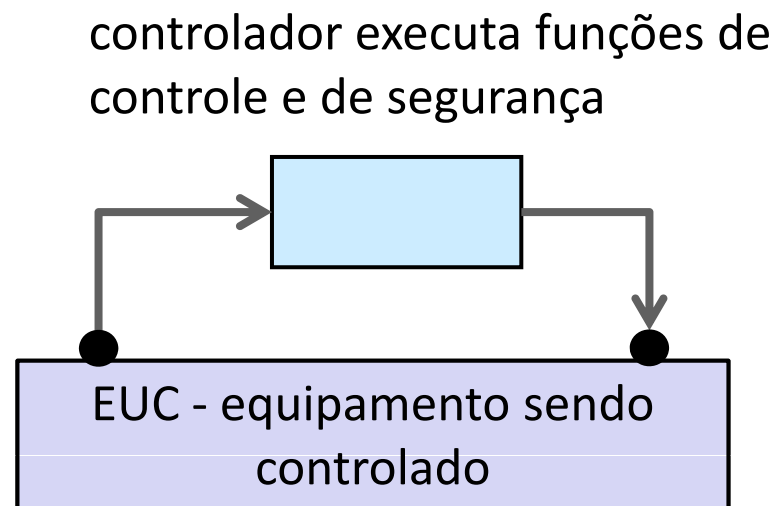
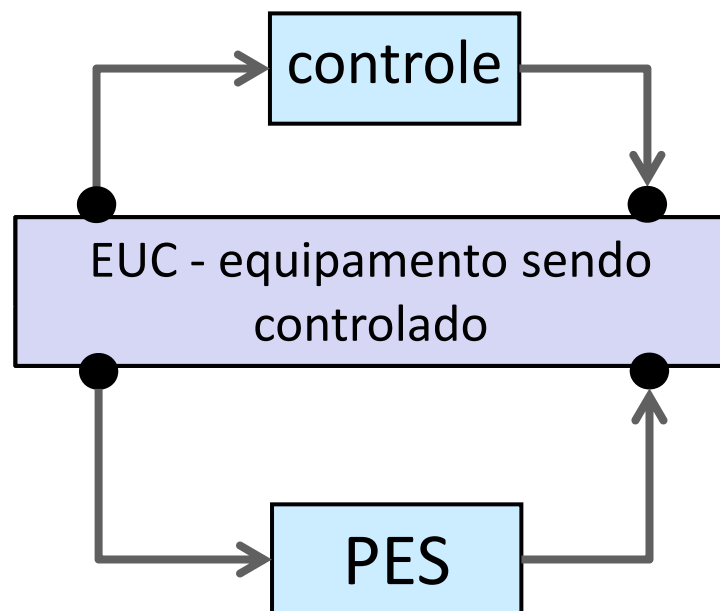
- sistemas de emergência (shut-down) em uma planta de processamento químico;
- indicador de segurança de carga em guindastes;
- sistema de sinalização em ferrovias;
- sistema de proteção e parada de emergência em máquinas e equipamentos ;
- sistemas de controle de exposição a doses de radiação (radioterapia médica);
- luzes de indicação de um automóvel, freio anti-bloqueante e sistemas de gerência de motor;
- monitoração remota, operação ou programação de uma planta através da rede.

PES

programmable electronic system (PES)



arquiteturas de sistemas de segurança



causa de defeitos

- especificação incorreta
 - do sistema, do hardware ou do software;
 - omissões na especificação dos requisitos de *safety*;
- defeitos de hardware **randômicos**;
- defeitos de hardware **sistemáticos**
- erros de software (qualquer tipo de bug);
- causas comuns de defeitos;
- erros humanos;
- influências do ambiente:
 - eletromagnéticas, temperatura, mecânicas, atmosféricas
- distúrbios na fonte de alimentação:
 - falta de energia, tensões reduzidas, reconexão de fonte.

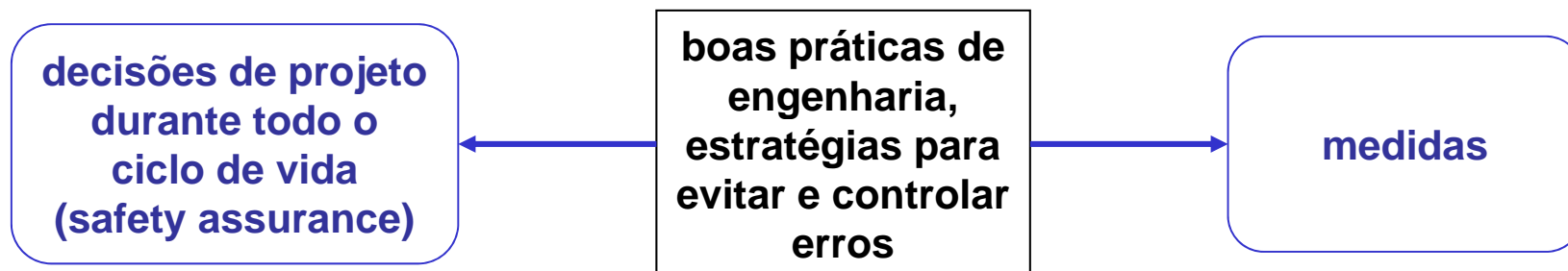
tipos de defeitos

defeitos de hardware **randômicos**

Reliability, Availability &
Maintainability



defeitos **sistemáticos** (incluindo erros de software)



SIL - Safety integrity levels

- IEC 61508 especifica 4 níveis de integridade de segurança para uma função de segurança.
 - safety integrity level 1 (SIL1) é o menor
 - safety integrity level 4 (SIL4) é o maior
- IEC 61508 detalha os requisitos necessários para alcançar cada nível.

os requisitos são mais rigorosos nos níveis mais altos de integridade para garantir menor probabilidade de defeitos perigosos

hardware vs software

- hardware

$$PFH_{G,1001} = \lambda_{DU}$$

- quantitativo

- taxa de defeitos de cada componente, forma de conexão no sistema, cobertura de falhas dos mecanismos de detecção, falhas correlacionadas (causa comum), taxa de reparo, ..

- qualitativo (para evitar introduzir falhas)

- ciclo de vida do projeto de hardware
 - conjunto de regras e recomendações
 - tabelas com técnicas
 - indicação de técnica **não recomendada**, **recomendada** e **altamente recomendada** para cada nível de SIL

NR, R, HR

hardware vs software

- software (qualitativo)
 - ciclo de vida do software
 - conjunto de regras e recomendações
 - tabelas com técnicas
 - indicação de técnica não recomendada, recomendada e altamente recomendada para cada nível de SIL

NR, R, HR

não são usadas fórmulas para cálculo de confiabilidade de software (falhas não são consideradas randômicas)

taxa de defeitos e SIL na norma IEC

- relaciona **taxas de defeitos** a níveis de integridade **derivados de medidas como FIT, MTTF**
- estabelece limites mínimos para taxas de defeito
 - baixa demanda - *probabilidade média de falhar* ao realizar função prevista sob demanda
 - por exemplo: PFD de 10^{-5}
 - significa 1 vez a cada 100000 vezes
 - alta demanda ou **modo de operação contínua** - *probabilidade de um defeito perigoso* por hora
 - PFH : 10^{-9}

baixa demanda



probabilidade de falhar
ao realizar função
prevista sob demanda

por exemplo: PFD de 10^{-5}
significa 1 vez a cada 100000 vezes

tabela SIL

baixa demanda -
probabilidade média de falhar ao realizar função prevista sob demanda

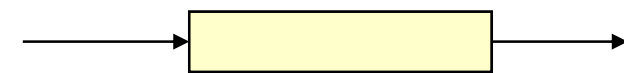
alta demanda ou modo de operação contínua -
probabilidade de um defeito perigoso por hora

modo de operação		
SIL	baixa demanda PFD	alta demanda PFH
4	$\geq 10^{-5}$ a $< 10^{-4}$	$\geq 10^{-9}$ a $< 10^{-8}$
3	$\geq 10^{-4}$ a $< 10^{-3}$	$\geq 10^{-8}$ a $< 10^{-7}$
2	$\geq 10^{-3}$ a $< 10^{-2}$	$\geq 10^{-7}$ a $< 10^{-6}$
1	$\geq 10^{-2}$ a $< 10^{-1}$	$\geq 10^{-6}$ a $< 10^{-5}$

defeitos

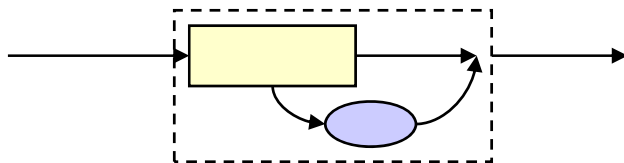


Arquiteturas MooN



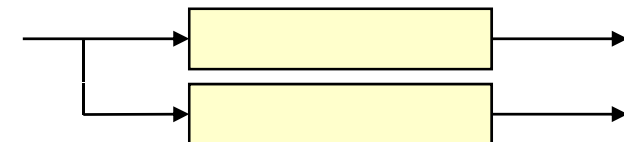
1oo1

componentes
confiáveis e em
pequeno número



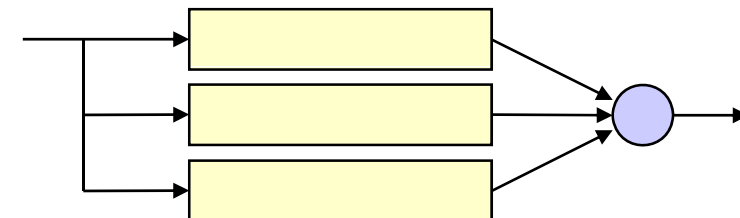
1oo1D

diagnóstico exige circuitos
extras ou algoritmos extras



1oo2

ideal para fail-safe



2oo3

ideal para períodos
curtos de missão

É possível alcançar alta integridade de segurança sem
necessidade de canais redundantes. Qual o segredo?

PFH para 1001

$$PFH_{G,1001} = \lambda_{DU}$$

λ_{DU} : λ devido a defeitos perigosos não detectáveis

PFH - probabilidade média de defeito por hora

onde se obtém λ ?

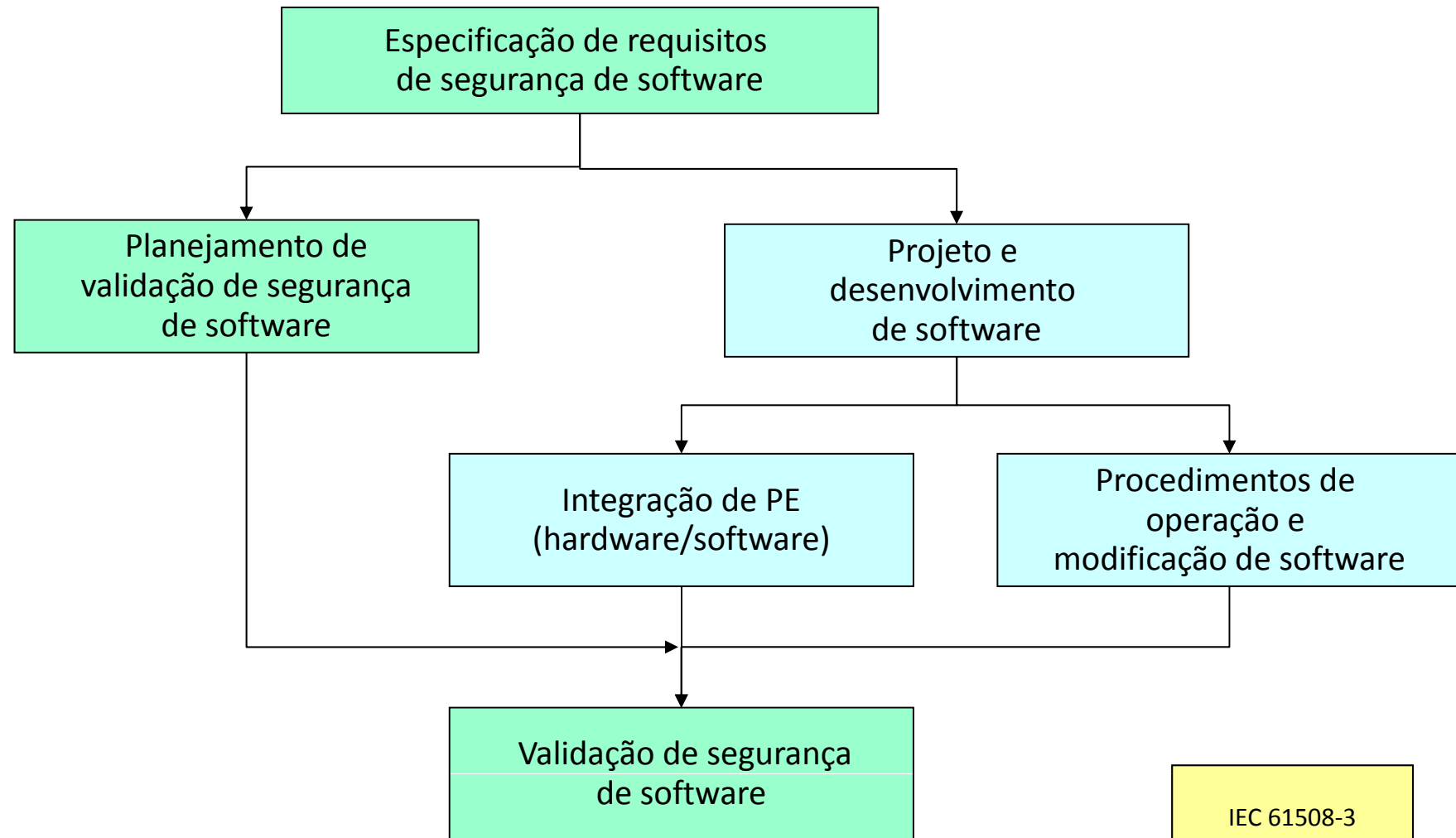
especificação de cada componente de hardware usando base de dados de componentes

causa comum: defeitos correlacionados

- importante nos sistemas 1oo2 ou 2oo3
 - causa comum - falhas que afetam canais redundantes
 - β - fator (peso) para defeitos perigosos não-detectáveis
 - β_D - fator para defeitos perigosos detectáveis

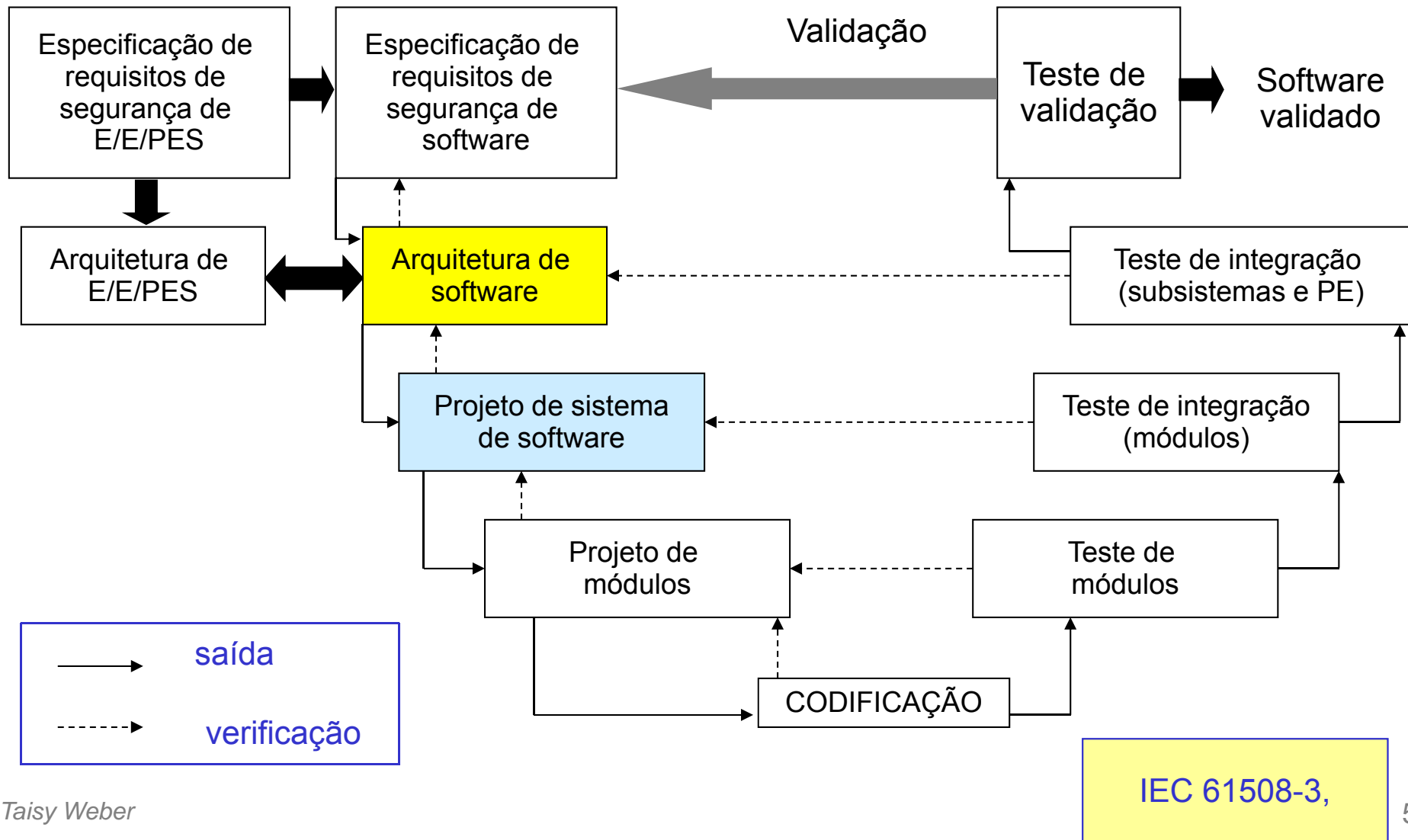
para sistemas 1oo2 ou 2oo3 o cálculo do λ_{DU} é mais complicado do que o cálculo de 1oo1 pois envolve β e esquemas série paralelo de composição de taxa de defeitos

Ciclo de vida de segurança de software



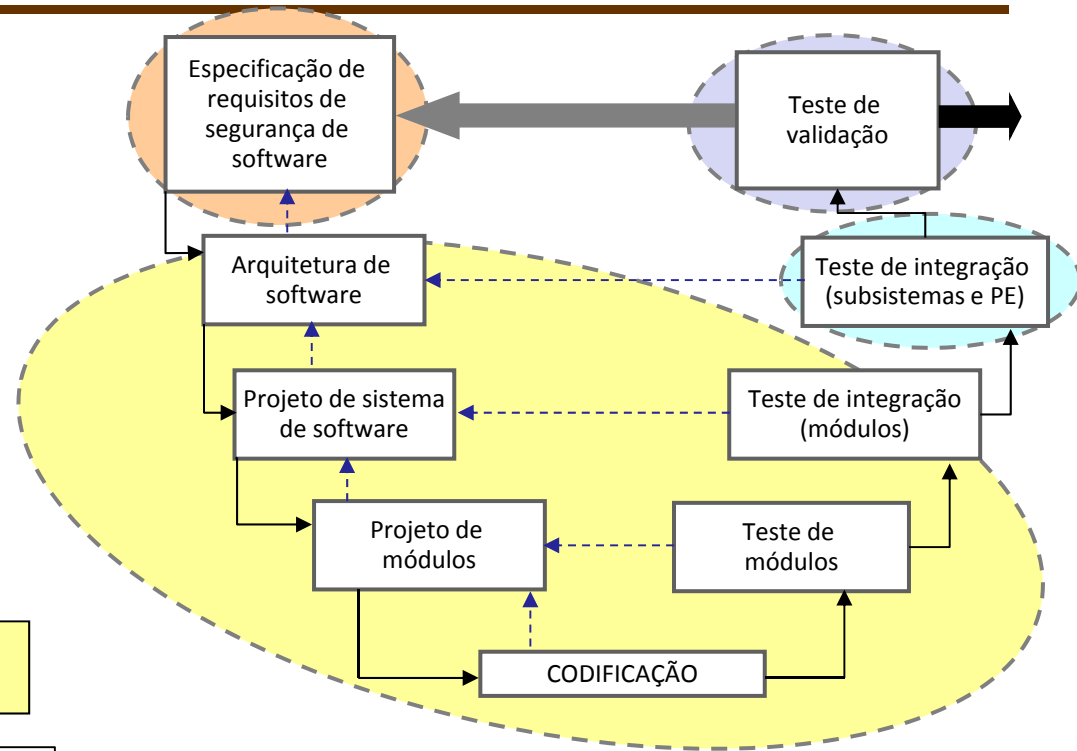
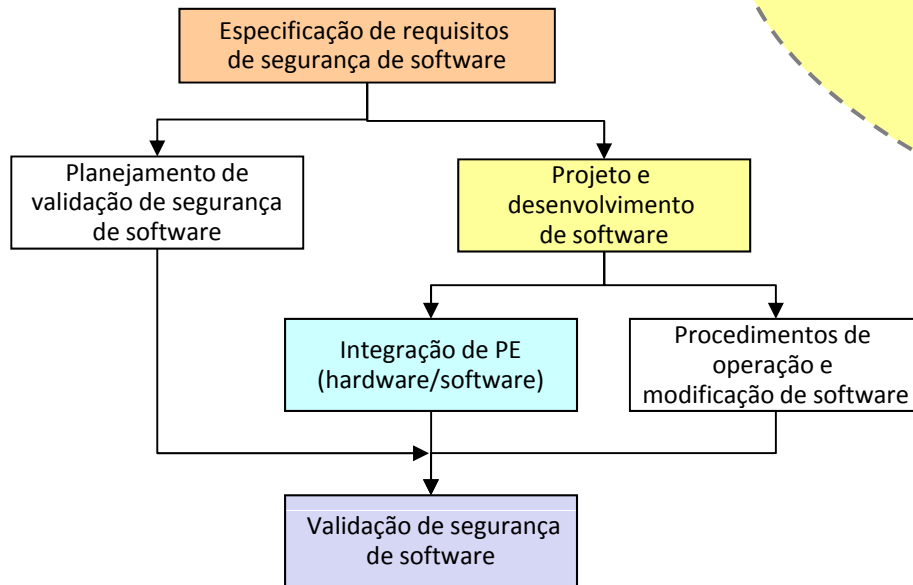
IEC 61508-3

modelo em V: software



Relação entre os modelos

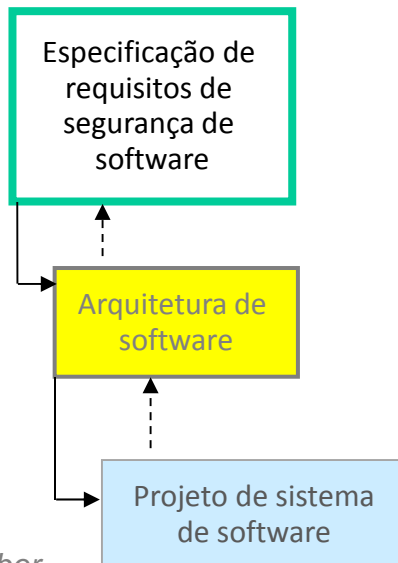
ciclo de vida de software



modelo V

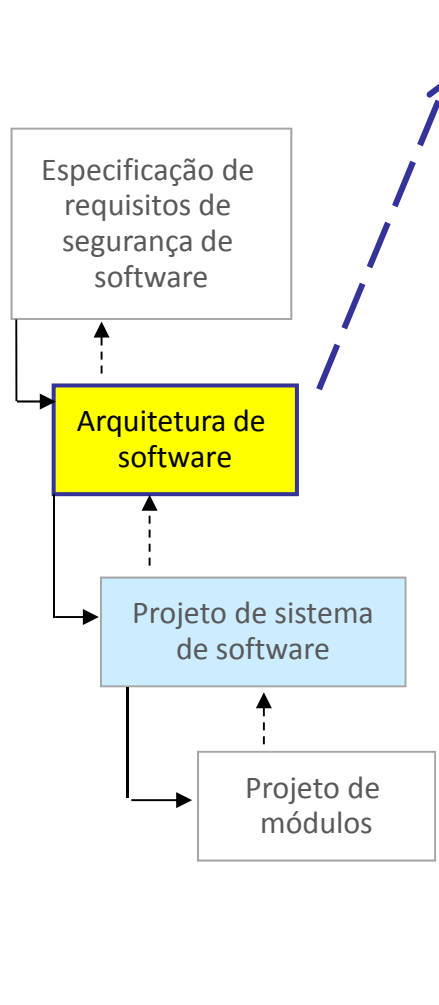
Nova tabela A.1 (IEC61508-3 2010)

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1a	Semi-formal methods	Table B.7	R	R	HR	HR
1b	Formal methods	B.2.2, C.2.4	---	R	R	HR
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	R	R	HR	HR
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	R	R	HR	HR
4	Computer-aided specification tools to support appropriate techniques/measures above	B.2.4	R	R	HR	HR



traceability foi introduzida na versão de 2010

exemplo de tabela: A.2 - arquitetura





		1	2	3	4
1	Fault detection and diagnosis	---	R	HR	HR
2	Error detecting and correcting codes	R	R	R	HR
3a	Failure assertion programming	R	R	R	HR
3b	Safety bag techniques	---	R	R	R
3c	Diverse programming	R	R	R	HR
3d	Recovery block	R	R	R	R
3e	Backward recovery	R	R	R	R
3f	Forward recovery	R	R	R	R
3g	Re-try fault recovery mechanisms	R	R	R	HR
3h	Memorising executed cases	---	R	R	HR
4	Graceful degradation	R	R	HR	HR
5	Artificial intelligence - fault correction	---	NR	NR	NR
6	Dynamic reconfiguration	---	NR	NR	NR
7a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	HR	HR	HR	HR
7b	Semi-formal methods	R	R	HR	HR
7c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	---	R	R	HR

Nova A.2

Technique/Measure *		Ref.	SIL 1	SIL 2	SIL 3	SIL 4
	Architecture and design feature					
1	Fault detection	C.3.1	---	R	HR	HR
2	Error detecting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	---	R	R	----
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	---	R	R	HR
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	---	---	---	R
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	---	---	R	HR
3f	Backward recovery	C.3.6	R	R	---	NR
3g	Stateless software design (or limited state design)	C.2.12	---	---	R	HR
4a	Re-try fault recovery mechanisms	C.3.7	R	R	---	---
4b	Graceful degradation	C.3.8	R	R	HR	HR

Arquitetura de software

Nova A.2

5	Artificial intelligence - fault correction 	C.3.9	---	NR	NR	NR
6	Dynamic reconfiguration 	C.3.10	---	NR	NR	NR
7	Modular approach	Table B.9	HR	HR	HR	HR
8	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
11a	Structured diagrammatic methods **	C.2.1	HR	HR	HR	HR
11b	Semi-formal methods **	Table B.7	R	R	HR	HR
11c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
11d	Automatic software generation	C.4.6	R	R	R	R
12	Computer-aided specification and design tools	B.2.4	R	R	HR	HR

Arquitetura de
software

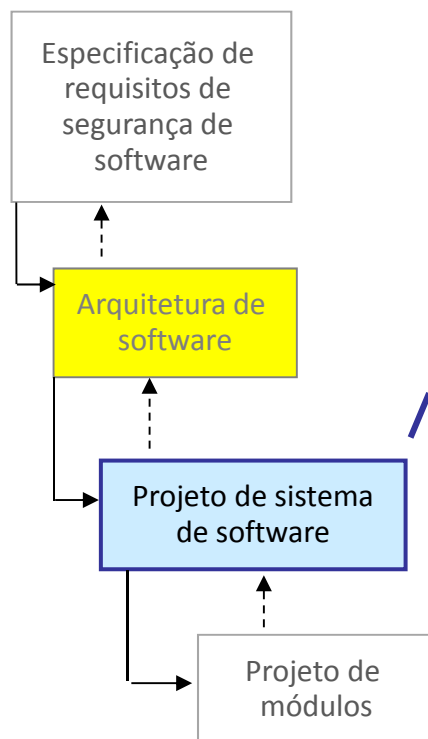
Nova A.2

13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	R	HR	HR	HR
13b	Time-triggered architecture	C.3.11	R	HR	HR	HR
13c	Event-driven, with guaranteed maximum response time	C.3.11	R	HR	HR	-
14	Static resource allocation	C.2.6.3	-	R	HR	HR
15	Static synchronisation of access to shared resources	C.2.6.3	-	-	R	HR

os itens 13 só se aplicam para sistemas com requisitos de temporização de segurança

Arquitetura de
software

exemplo de tabela: projeto de software



		1	2	3	4
1	Suitable programming language	HR	HR	HR	HR
2	Strongly typed programming language	HR	HR	HR	HR
3	Language subset	---	---	HR	HR
4a	Certificated tools	R	HR	HR	HR
4b	Tools: increased confidence from use	HR	HR	HR	HR
5a	Certificated translator	R	HR	HR	HR
5b	Translator: increased confidence from use	HR	HR	HR	HR
6	Library of trusted/verified software modules and components	R	HR	HR	HR

documentos IEC

2550,-

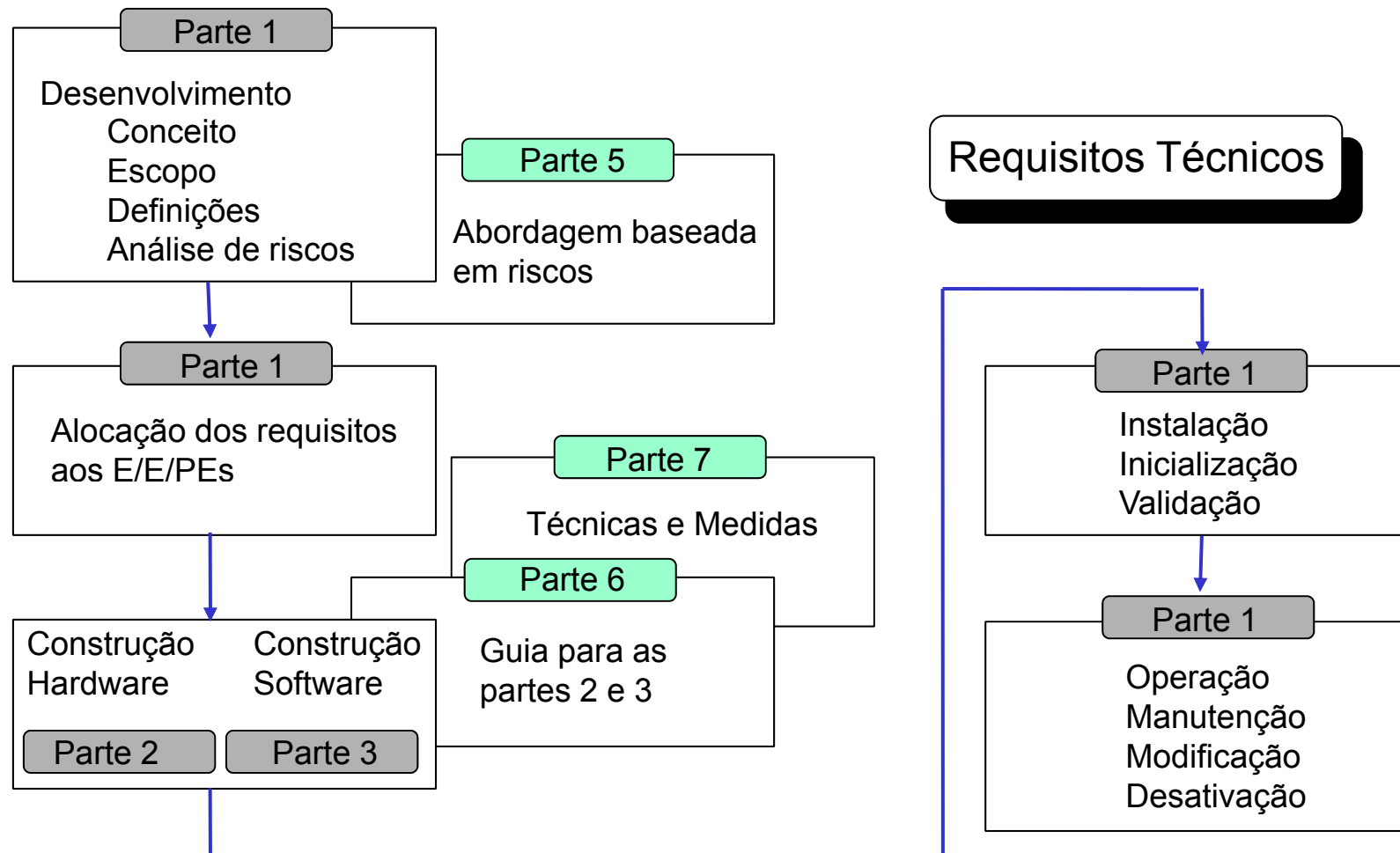
- IEC 61508-0 - Functional safety and IEC 61508
- IEC 61508-1 - General requirements
- IEC 61508-2 - Requirements for electrical/electronic/
programmable electronic safety-related systems
- IEC 61508-3 - Software requirements
- IEC 61508-4 - Definitions and abbreviations
- IEC 61508-5 - Examples of methods for the determination of safety integrity
levels
- IEC 61508-6 - Guidelines on the application of IEC 61508-2 and IEC 61508-3
- IEC 61508-7 - Overview of measures and techniques

<http://www.iec.ch/functionalsafety/standards/>

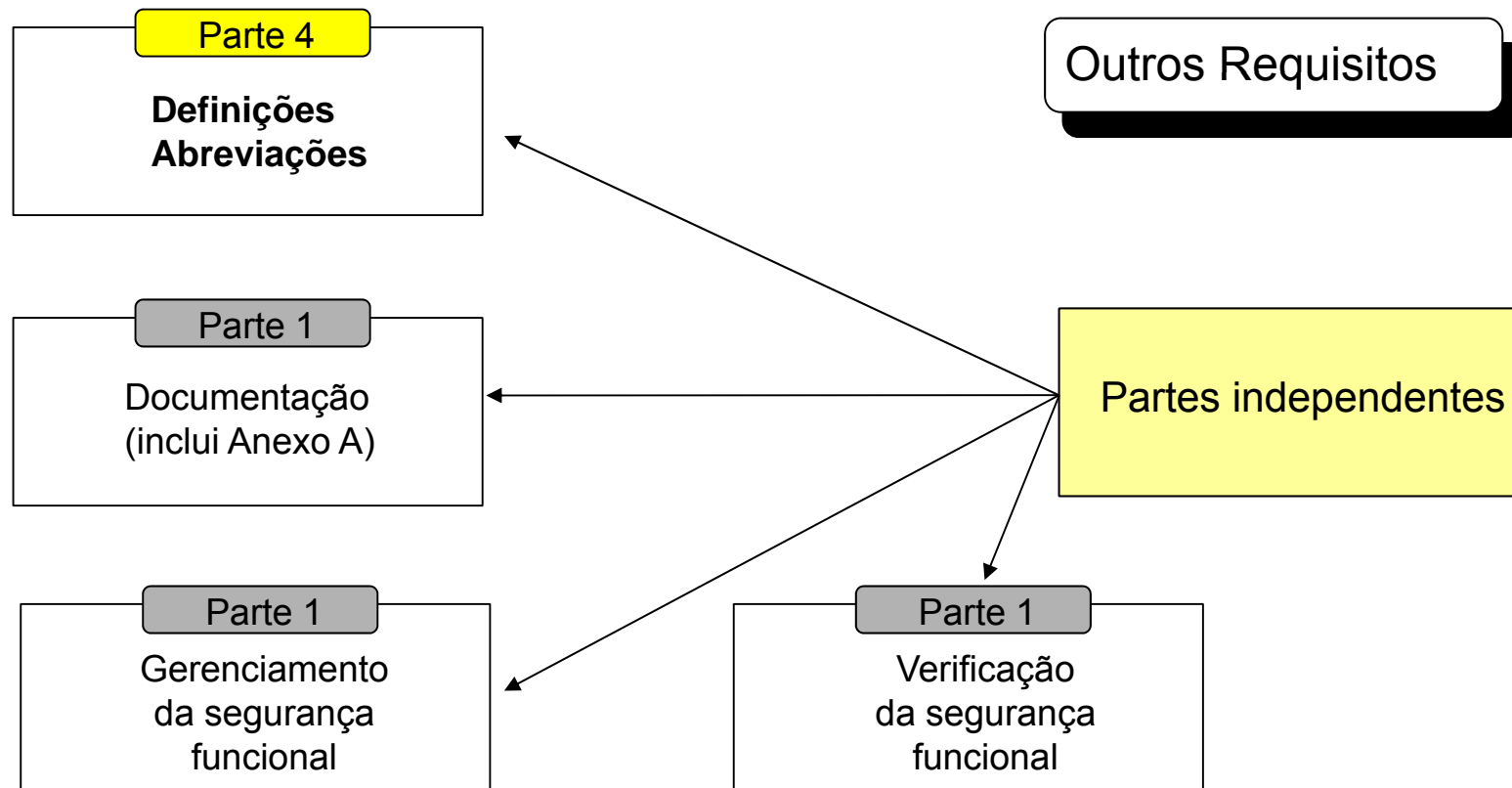


exceto a parte zero, de 2005, todas as demais são de 98 a 2000 e foram revisadas em abril de 2010

IEC 61508



IEC 61508



tolerância a falhas e safety (norma)

- conceitos da área de dependabilidade não se tornaram obsoletos com a norma
 - falha, erro e defeito
 - confiabilidade e suas medidas
 - modelagem de confiabilidade
 - teste e validação
- os termos e técnicas usuais na área de tolerância a falhas continuam aplicáveis
 - as diferenças são apenas relativas a área de aplicação (segurança crítica)
 - dano, perigo, risco

bibliografia

- *Functional safety and IEC 61508: A basic guide*
 - November 2002 (IEC web site)
- Dunn, William R. *Designing Safety-Critical Computer Systems*. Computer, 2003
- Bell, R.: *Introduction to IEC 61508*. In Proc. of the 10th Australian Workshop on Safety Critical Systems and Software, 3-12. (2006)

Abreviaturas

- E/E/PE Electrical/electronic/programmable electronic
- E/E/PES Electrical/electronic/programmable electronic system
- EUC Equipment under control
- PES Programmable electronic system
- PLC Programmable logic controller
- SIL Safety integrity level

Tolerância a falhas em sistemas distribuídos

UFRGS

Taisy Silva Weber

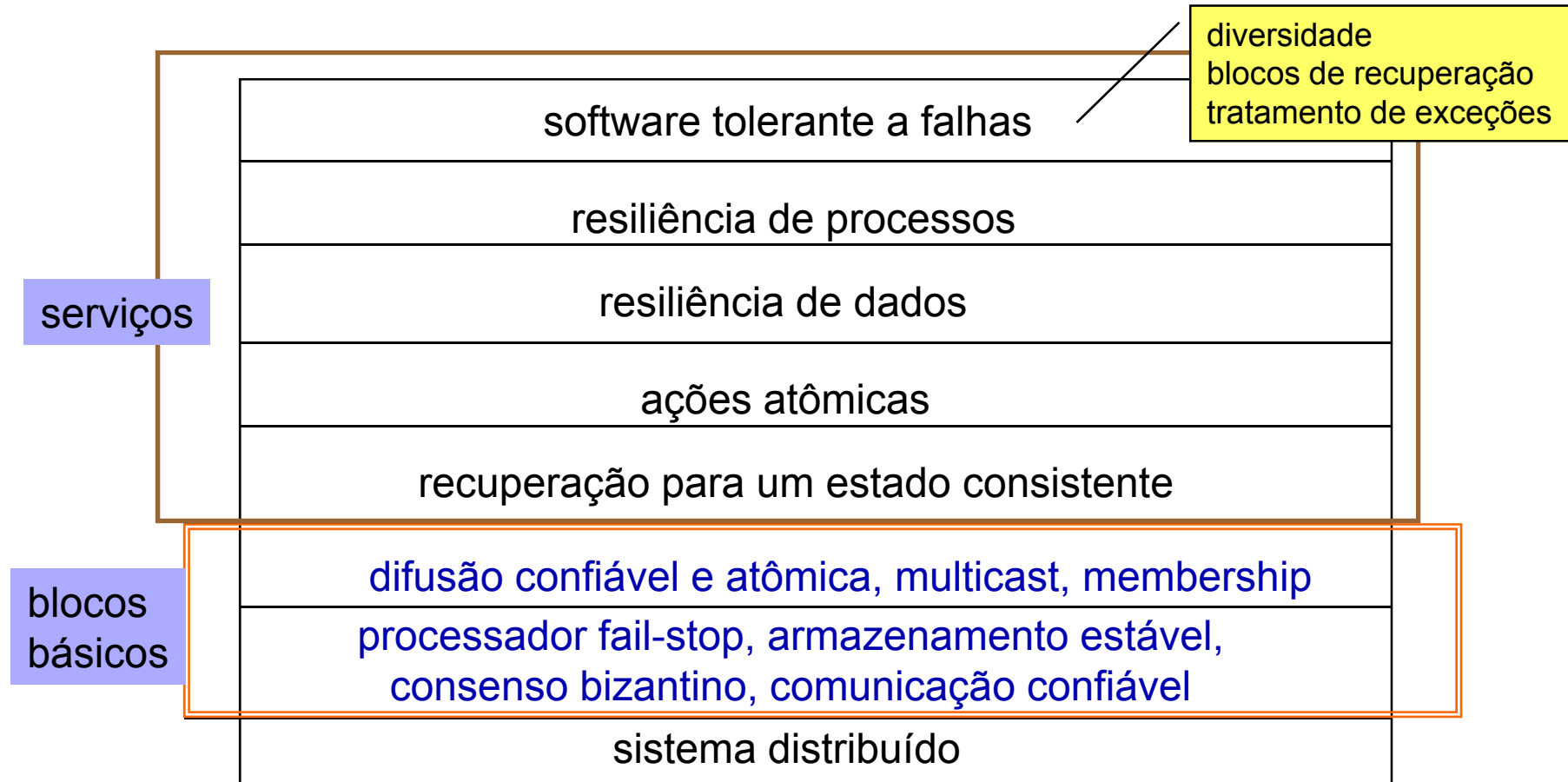
Redundância implícita

- ✓ sistemas distribuídos são naturalmente redundantes
 - ✓ mas redundância sozinha não aumenta a confiabilidade do sistema



Níveis

JALOTE, P. Fault tolerance in distributed systems. Prentice Hall, Englewood Cliffs, New Jersey, 1994



Nesta disciplina

- ✓ sistemas distribuídos
 - ✓ processador fail-stop
 - ✓ armazenamento estável
 - ✓ consenso bizantino
 - ✓ comunicação confiável
 - ✓ difusão confiável
 - ✓ recuperação para um estado consistente
 - ✓ replicação de dados
-
- blocos básicos
- serviços

Exemplos de sistemas distribuídos

- ✓ sistemas de pequena escala
 - ✓ LANs
 - ✓ clusters
- ✓ sistemas de grande escala (ou larga escala)
 - ✓ Internet
 - ✓ Grids
 - ✓ nuvens
- ✓ propriedades fortes de tolerância a falhas não **escalam** adequadamente
 - ✓ custo muito alto para manter **consenso**, **consistência** de réplicas e **estado global distribuído** em sistemas de larga escala sujeitos a falhas

clusters, grids, clouds

	Clusters	Grids	Clouds
Size/ scalability	centenas	milhares	centenas a milhares
OS	Linux, Windows	Unix	VM e múltiplas OSs
Ownership	único	múltiplo	único
Inter- connection	Dedicated, high- end with low latency and high bandwidth	Mostly Internet with high latency and low bandwidth	Dedicated, high-end with low latency and high bandwidth
Security/ privacy	login/password Medium level of privacy.	Public/private key pair based authentication and mapping a user to an account. Limited support for privacy.	Each user/application is provided with a virtual machine. High security/privacy.
Discovery	Membership services	Centralised indexing and decentralised info services	Membership services

clusters, grids, clouds

	Clusters	Grids	Clouds
Resource management	Centralized	Distributed	Centralized/ Distributed
Standards/ inter-operability	Virtual Interface Architecture	Some Open Grid Forum standards	Web Services (SOAP and REST)
Capacity	Stable and guaranteed	Varies, but high	Provisioned on demand
Failure management (Self-healing)	Limited (often failed tasks/ applications are restarted).	Limited (often failed tasks/ applications are restarted).	Strong support for failover and content replication. VMs can be migrated from one node to other

Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg, e Ivona Brandic. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". *Future Generation Computer Systems* 25, nº. 6 (junho 2009): 599-616.

Sistemas distribuídos

- ✓ sem memória compartilhada
- ✓ sem relógio global

- ✓ modelos

- ✓ modelo físico

nodos e rede

processador
relógio local
memória local volátil
armazenamento não volátil
interface de rede
software

- ✓ modelo lógico

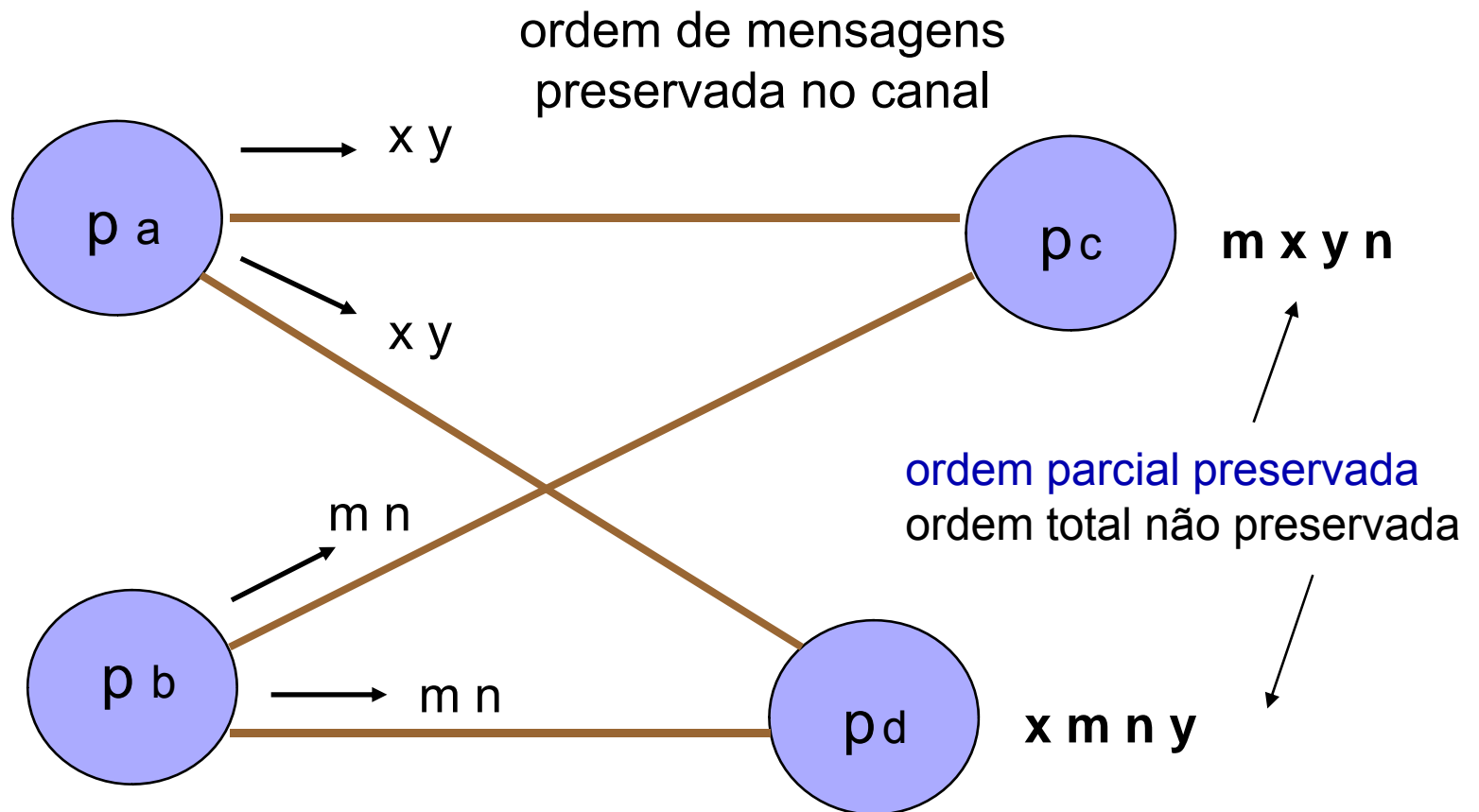
processos e canais

rede completamente conectada

existe um **canal** entre quaisquer dois processos que interagem,
buffer infinito e livre de erros

canais entregam mensagens na ordem que foram enviadas

Canal



Modelo lógico

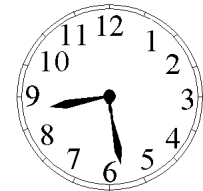
Modelos de tempo

- ✓ sistema assíncrono
 - ✓ não existem limites de tempo



- ✓ sistema síncrono
 - ✓ existe um limite de tempo finito e conhecido

sistema correto opera dentro desse limite

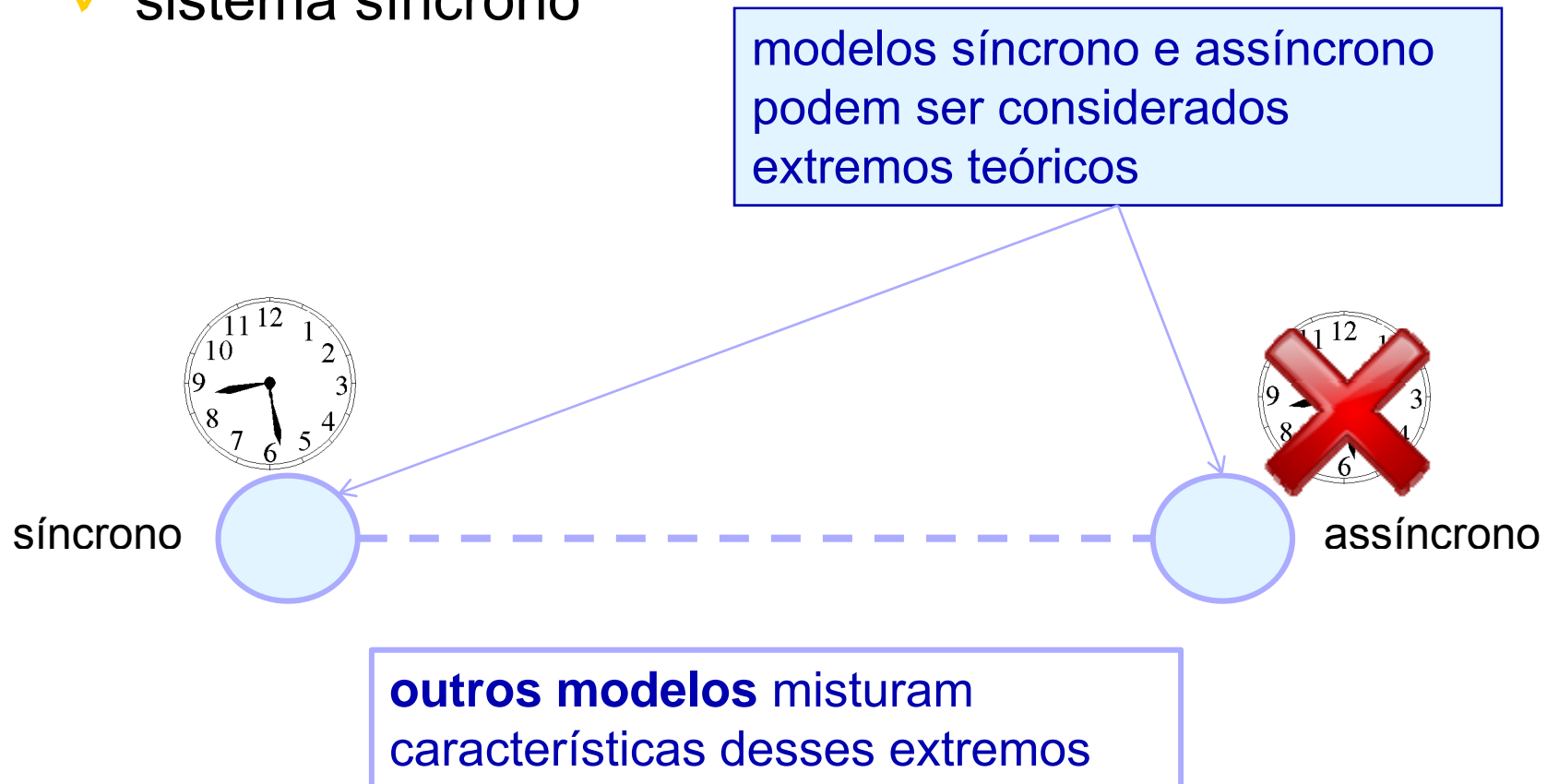


- ✓ falha de componente pode ser deduzida pela ausência de resposta
- ✓ timeout

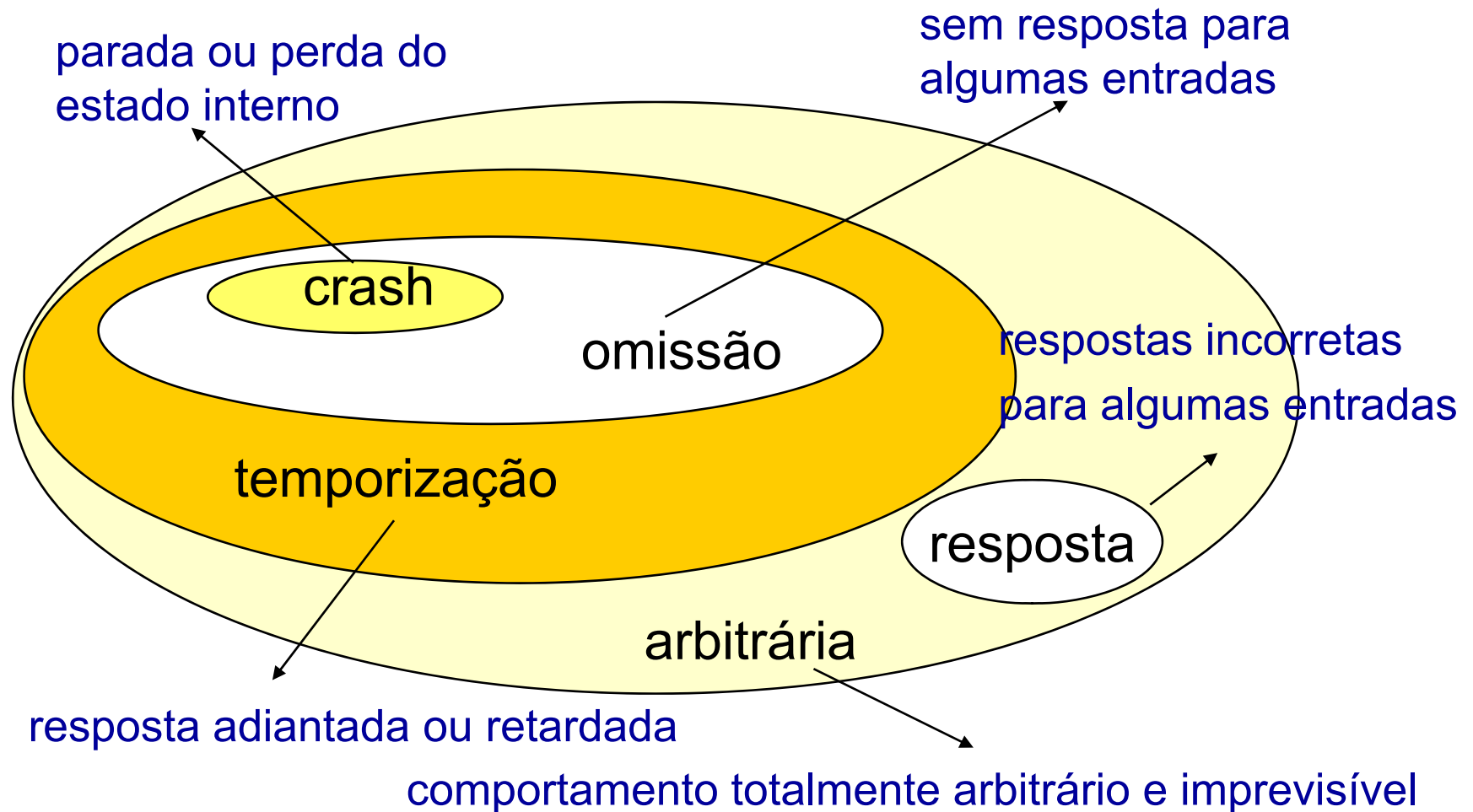
detecção de defeitos em nodos e perdas de mensagens

Modelos de tempo

- ✓ sistema assíncrono
- ✓ sistema síncrono



Classificação de falhas (Cristian)

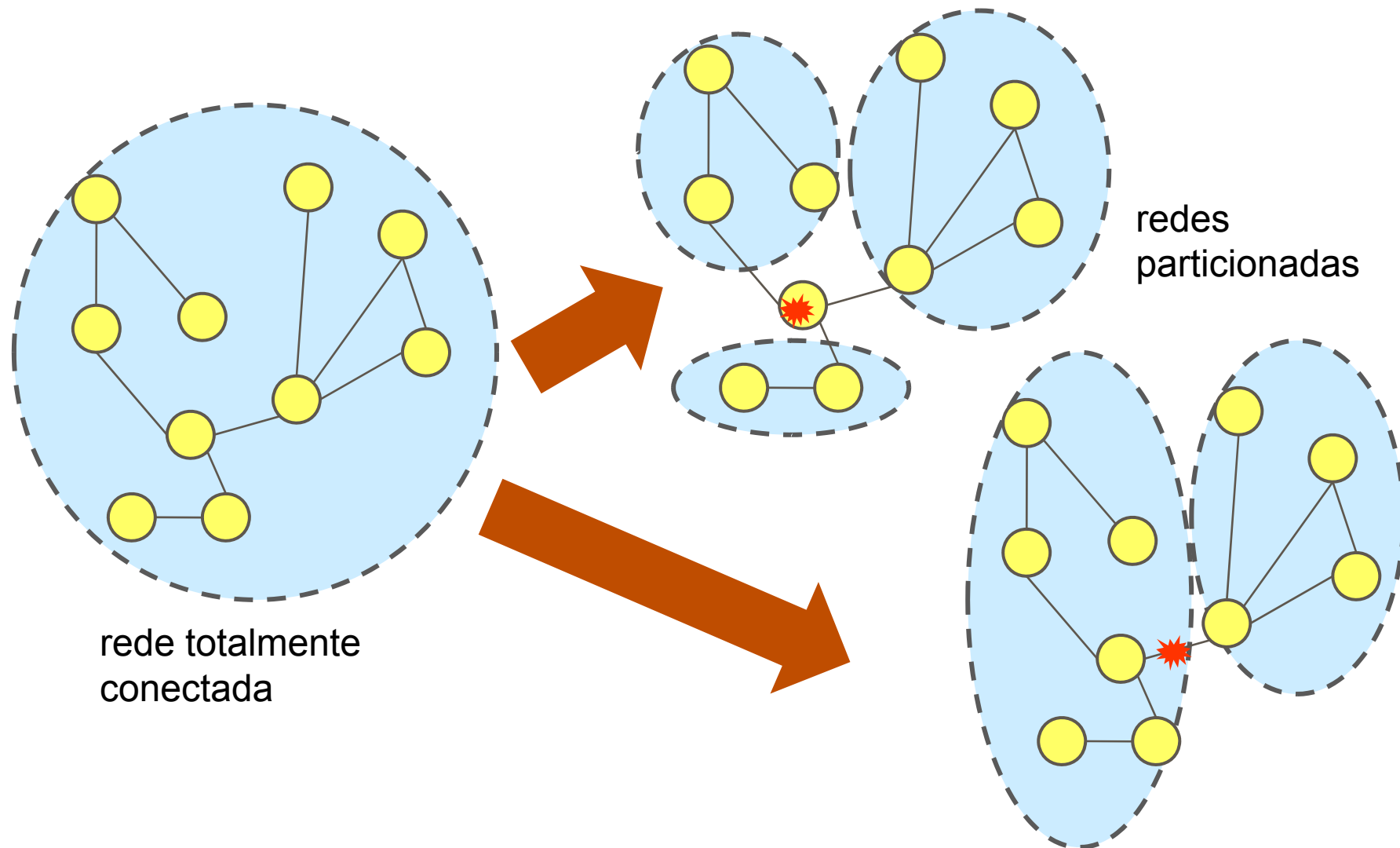


Exemplos de falhas

- ✓ processador:
 - ✓ crash ou bizantinas
- ✓ rede de comunicação:
 - ✓ todos os tipos
- ✓ clock:
 - ✓ temporização ou bizantinas
- ✓ meio de armazenamento
 - ✓ temporização, omissão ou resposta
- ✓ software:
 - ✓ resposta

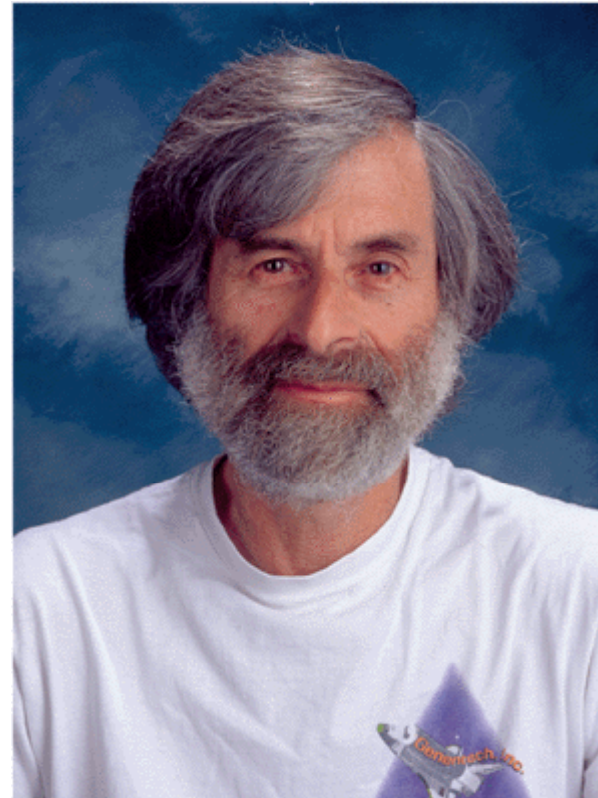
O modelo de falhas é uma simplificação da realidade. Na literatura aparecem vários outros modelos de falhas para sistemas distribuídos. Alguns incluem particionamento de rede.

Particionamento



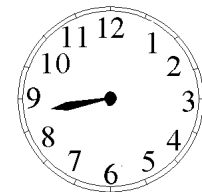
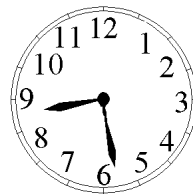
Lamport

- ✓ clocks lógicos
- ✓ generais bizantinos
- ✓ LaTeX
- ✓ Paxos (consenso)



Ordenação de eventos

- ✓ determinar a ordem de eventos que ocorrem em nodos diferentes, medidos por relógios diferentes



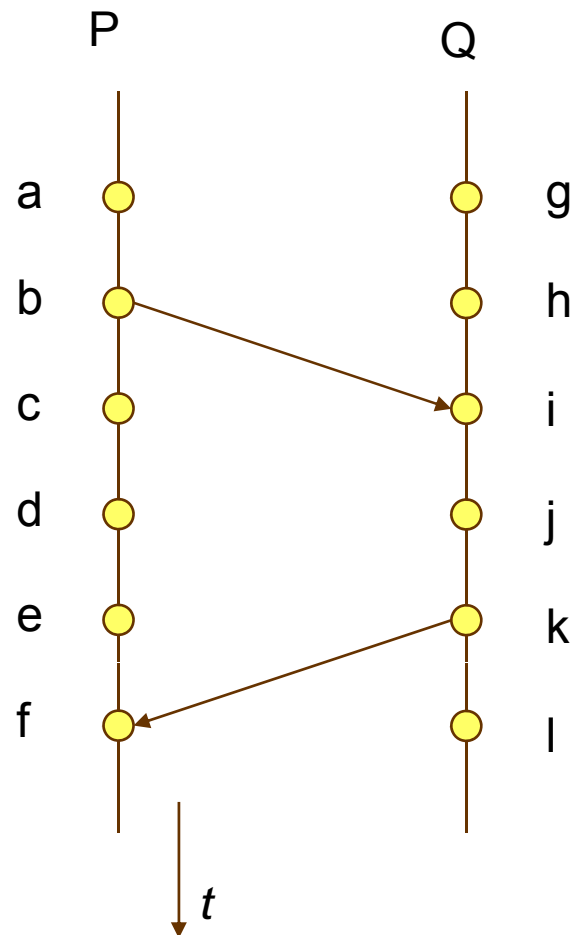
- ✓ relação:

- ✓ a “aconteceu antes de” b : $a \rightarrow b$

ordem de ordem parcial

nem todos os eventos podem
ser ordenados

Ordenação de eventos



$a \rightarrow b$

$b \rightarrow i$

$h \rightarrow i$

$a \rightarrow b$ e $b \rightarrow i$ então $a \rightarrow i$

nem $a \rightarrow h$, nem $h \rightarrow a$

- ✓ se **a** e **b** são eventos do mesmo processo e **a** é executado antes de **b** então $a \rightarrow b$
- ✓ se **a** é **send** e **b** é **receive** da mesma msg então $a \rightarrow b$
- ✓ $a \rightarrow b$ e $b \rightarrow c$ então $a \rightarrow c$
- ✓ eventos concorrentes: nem $a \rightarrow b$, nem $b \rightarrow a$

Ordenação de eventos

- ✓ sistema de clock lógico
 - ✓ um sistema de clock lógico é correto se é consistente com a relação \rightarrow

exemplo: timestamp T

- ✓ clock lógico carimba um evento de forma que a relação de ordem parcial é mantida

é possível estabelecer uma ordem total não temporal com relógios lógicos

Concordância bizantina

- ✓ consenso
 - ✓ problema recorrente em sistemas distribuídos
 - ✓ solução trivial para sistemas livres de falhas (ou seja perfeitos)
 - ✓ não trivial para sistemas com falhas arbitrárias e assíncronos

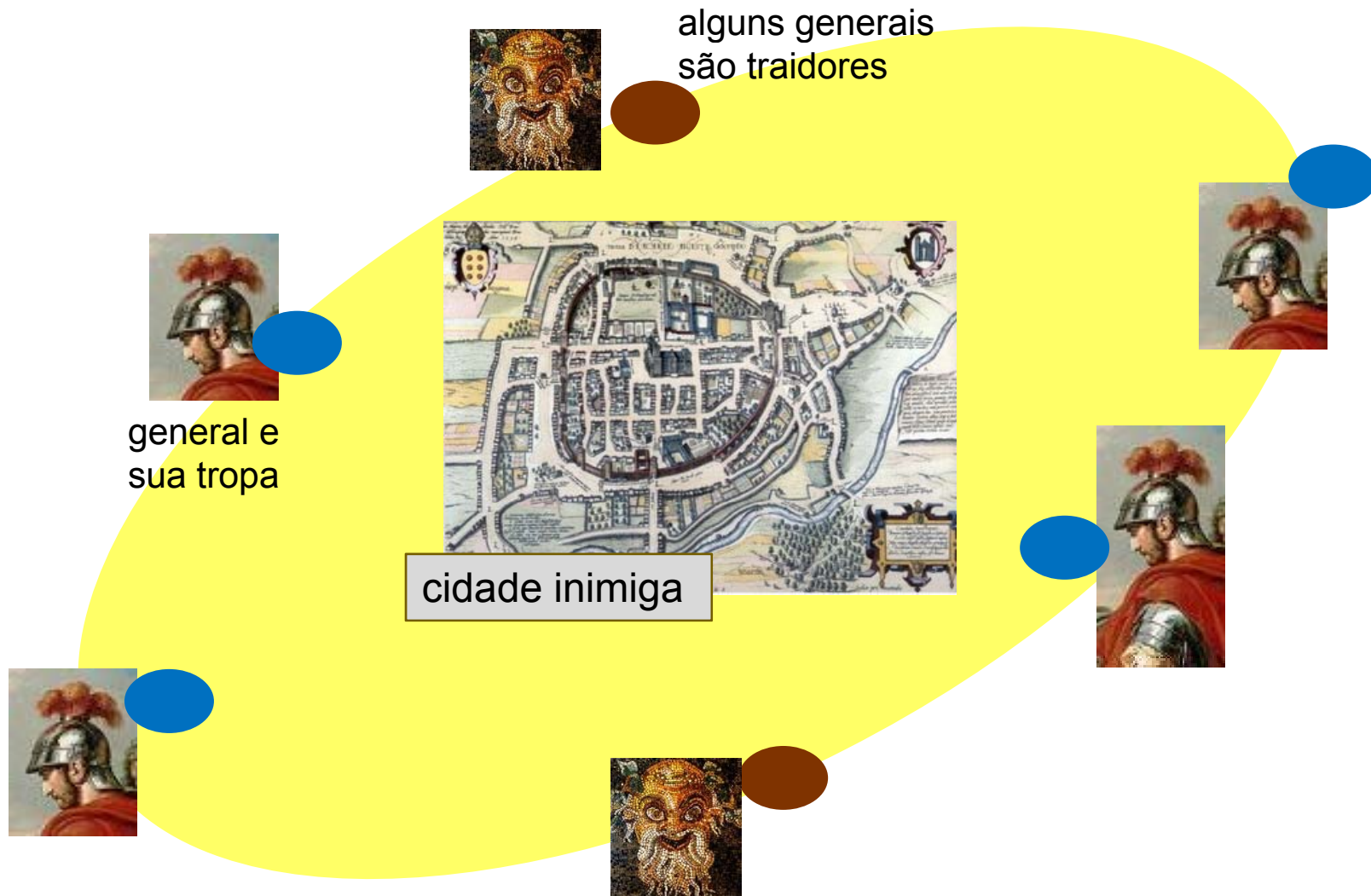
Israel Koren, C. Mani Krishna - *Fault-Tolerant Systems* - Morgan Kaufmann, 2007
pg 42

Concordância bizantina

- ✓ alcançar **consenso** na presença de traidores
 - ✓ defeitos bizantinos
 - ✓ comportamento **arbitrário**
 - ✓ nodo pode enviar informações diferentes para os diferentes componentes com quem se comunica
- ✓ problema dos **generais bizantinos**

Lamport, Shostak e Pease, 82

Generais bizantinos



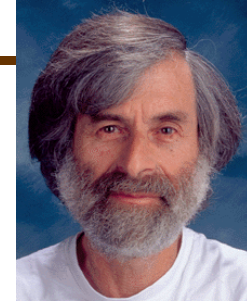
Generais bizantinos

traidores não podem
provocar divisão (alguns
atacam e outros
recuam)



não traidores devem
chegar a **consenso** sobre
atacar ou recuar

Algoritmos de Lamport



- ✓ Lamport 82 (Lamport, Shostak e Pease)

- ✓ solução para:

- ✓ sistemas síncronos
 - ✓ totalmente conectado
 - ✓ dois algoritmos :
 - ✓ msgs **orais**
 - ✓ msg **assinadas**

Orais: comuns, sem assinatura (o emissor é conhecido, impossível verificar a integridade do conteúdo, um nodo traidor pode alterar o conteúdo)

- ✓ relação entre traidores (**m**) e não traidores (**n-m**)

mensagens orais: $n \geq 3m + 1$

mensagens assinadas: $n \geq m + 2$

- ✓ grande número de rodadas: **m + 1**
 - ✓ grande número de mensagens: **$O(n^m)$**

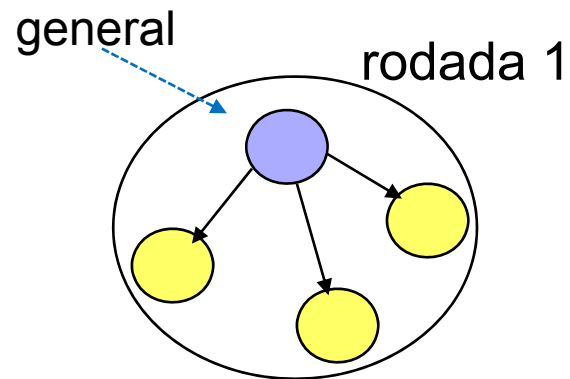
Algoritmo para mensagens orais

consistência interativa (ICA)

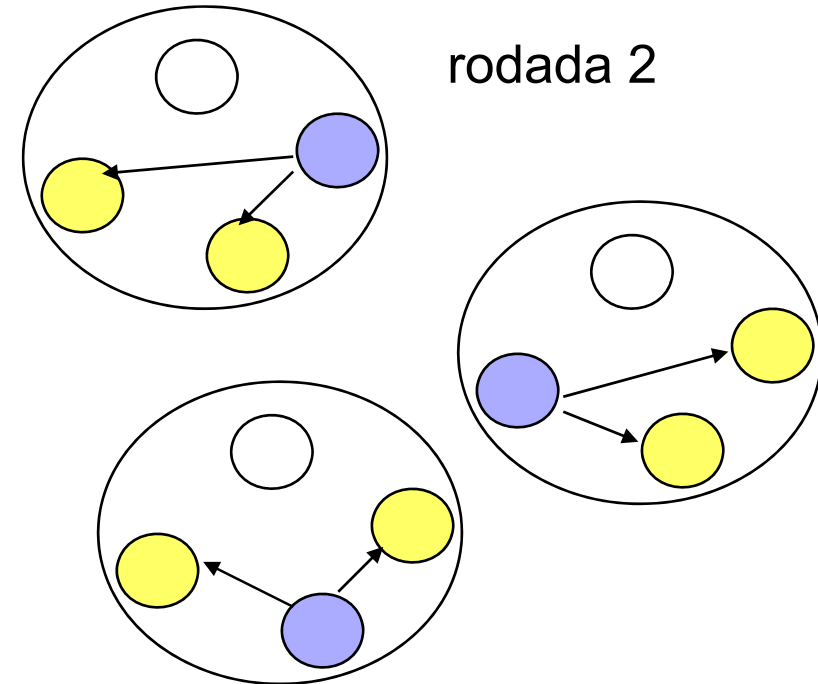
- ✓ ICA(0)
 - ✓ o general envia o seu valor para os outros $n - 1$ nodos
 - ✓ cada nodo usa valor recebido, ou default (se nada recebeu)
- ✓ (fim da recursão)
- ✓ ICA(m), $m > 0$
 - ✓ o general envia o seu valor para os outros $n - 1$ nodos
 - ✓ nodo i: $v(i)$ (valor recebido pelo nodo i ou default),
 - ✓ nodo i atua como general em ICA(m-1) enviando $v(i)$ para os demais $n - 2$ nodos (confirmação do valor)
 - ✓ para cada nodo i:
 - ✓ $v(j)$ valor recebido do nodo j ($j \neq i$)
 - ✓ nodo i usa valor *maioria*($v(1), \dots, v(n-1)$)

ICA(m)

ICA(m) deve ser usado por todos os nodos para alcançar consenso



usando ICA(m) em cada nodo,
cada nodo do sistema terá o
mesmo valor assumido para
todos os outros nodos



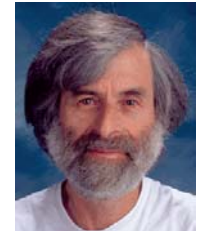
ICA(1)

no máximo
um traidor

toda mensagem enviada é recebida
corretamente;
o receptor sabe quem enviou a mensagem
a ausência de uma mensagem pode ser
detectada (time-out)

Concordância bizantina

- ✓ algoritmos de Lamport
 - ✓ ótimos em relação ao número de rodadas
 - ✓ mas exigem número exponencial de mensagens
- ✓ outros algoritmos foram propostos
 - ✓ alguns mais eficientes em relação ao número de mensagens
 - ✓ alguns suportam sistemas não totalmente conectados
 - ✓ alguns suportam sistemas assíncronos através de procedimentos randômicos
 - ✓ comunicação epidêmica:
 - ✓ garante certa probabilidade de todos os nodos recebem uma mensagem difundida



Impossibilidade de consenso

- ✓ consenso é impossível em sistemas assíncronos (Fischer, 1985) - conhecido como **problema FLP**
 - ✓ mesmo com um único traidor
 - ✓ mesmo quando só ocorrem falhas de crash

Lamport não chegou a provar para qualquer falha, apenas para falhas bizantinas

- ✓ entretanto o problema pode ser solúvel se alguma forma fraca de sincronismo for introduzido (Dolev 1987; Dwork 1988; Chandra e Toueg, 1996)
- ✓ esforços foram direcionados para detectores de falhas não confiáveis (Chandra e Toueg)

Paxos

- ✓ Lamport 1988
 - ✓ ilha grega Paxos
 - ✓ sistema de consenso ficcional
- ✓ consenso
 - ✓ sistemas assíncronos
 - ✓ não garante progresso (FLP)
 - ✓ mas garante *safety* (livre de inconsistências)
- ✓ usado em sistemas de réplicas de arquivos e banco de dados

[usado pelo google, Microsoft, IBM e outras empresas]

Armazenamento estável

essencial em vários esquemas de suporte a tolerância a falhas

- ✓ parte do estado do sistema permanece disponível mesmo após defeito do sistema
- ✓ conteúdo é preservado apesar de falhas
 - ✓ um disco magnético **não** é armazenamento estável
- ✓ exemplos de implementação:
 - ✓ sombreamento de disco
 - ✓ imagens idênticas em dispositivos separados
 - ✓ 2 discos - espelhamento
- ✓ RAID: **R**edundant **A**rray of **I**nexpensive **D**isks

Tandem

I pode ser também Independent

Exemplo de implementação

- ✓ **Redundant Array of Inexpensive Disks**

1987

- ✓ propostos inicialmente para diminuir **custos** de armazenamento e prover alta **velocidade**
 - ✓ **atualmente:** confiabilidade e desempenho

- ✓ bit interleaving: entrelaçamento de bits

- ✓ perda de um disco pode comprometer todo o array
- ✓ colocando mais discos ou mais código de recuperação de erros esse problema pode ser contornado

<http://en.wikipedia.org/wiki/RAID>

RAID

- ✓ RAID 0: sem redundância
- ✓ RAID 1
 - ✓ dois discos idênticos espelhados
 - ✓ método mais caro (100% redundância)
- ✓ RAID 2
 - ✓ bits entrelaçados + palavra código
 - ✓ número de discos depende do algoritmo de correção de erros
- ✓ RAID 3
 - ✓ bytes entrelaçados + disco extra para paridade
 - ✓ cada byte sequencial está num disco diferente
- ✓ RAID 4
 - ✓ como RAID 3, mas com **setores** entrelaçados
 - ✓ vantagem para o SO
 - ✓ paridade em disco extra

RAID 5 e 6

existem combinações

✓ RAID 5

o mais popular

- ✓ como RAID 3 mas sem disco de paridade
 - ✓ **paridade é distribuída** pelos discos do sistema
- ✓ pode ser implementado a partir de dois discos

além da paridade distribuída

✓ RAID 6

- ✓ como RAID 5, mas com **mais um** disco de paridade
 - ✓ dois discos podem falhar sem perda de dados
 - ✓ pode trocar drive defeituoso com sistema em operação
- ✓ degrada para RAID 5 quando 1 disco está defeituoso

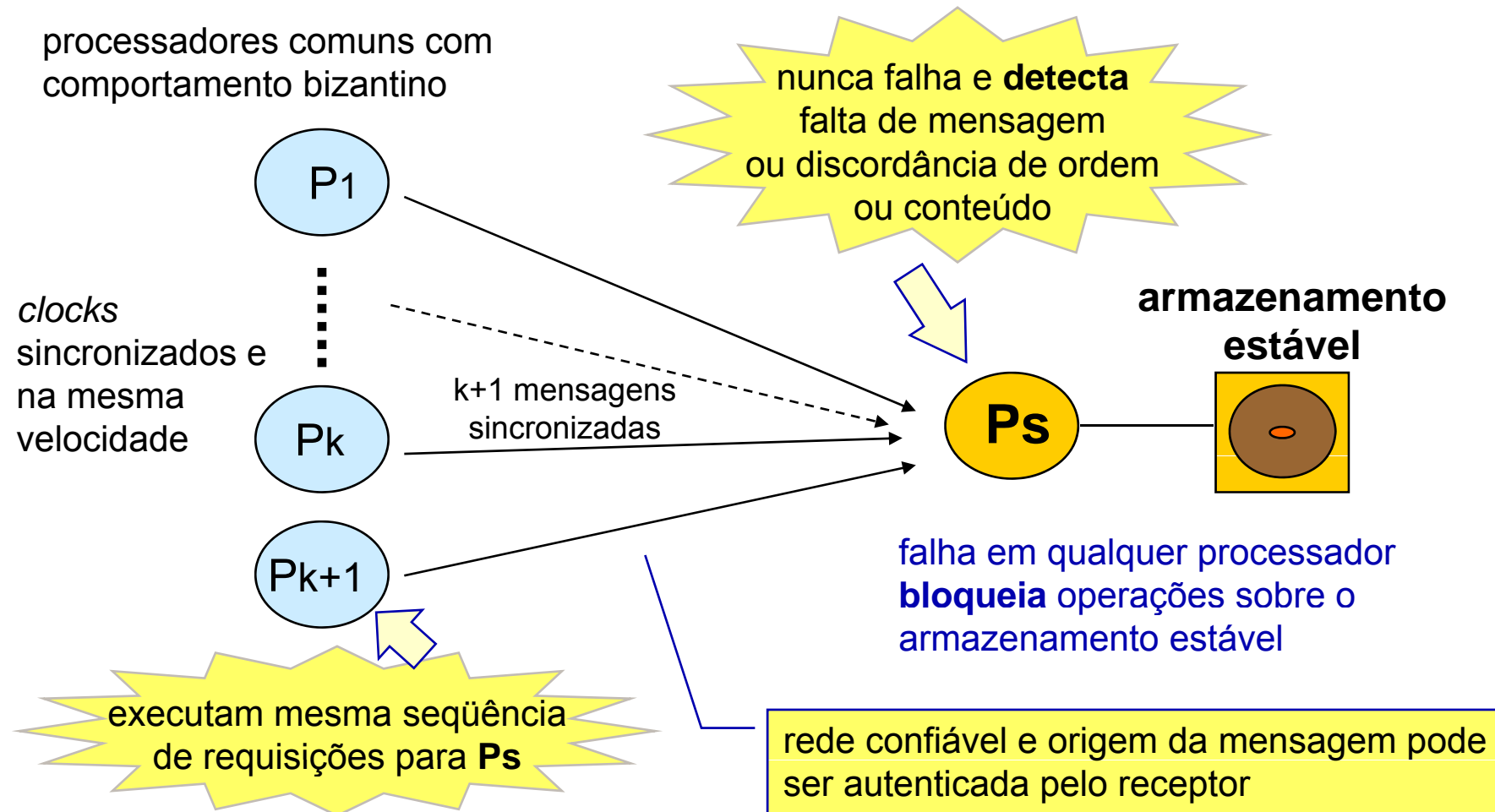
Processadores fail-stop

- ✓ em caso de defeito, nodo cessa operação sem realizar qualquer ação incorreta
 - ✓ comportamento **fail-stop** assumido por grande parte dos esquemas de TF
- ✓ processadores reais não são por natureza **fail-stop**
 - ✓ processadores reais com defeito se comportam de maneira **arbitrária**
 - ✓ nodos **aproximadamente fail-stop** podem ser construídos a partir de processadores reais (**k fail-stop**)

exemplo: grande parte dos servidores tolerantes a falhas

k fail-stop

comporta-se como um processador fail-stop a menos que **k+1** ou **mais** componentes falhem



Tipos de difusão

- ✓ broadcast

- ✓ envio de mensagens a **todos** os nodos do sistema

- ✓ multicast

multicast envolve o conceito de grupo

- ✓ envio de mensagens a **alguns** nodos do sistema

- ✓ sensível a falhas de nodo e comunicação

em broadcast ou multicast sobre comunicação **ponto a ponto**: um nodo pode falhar após ter iniciado difusão, assim alguns nodos podem ter recebido a mensagem e outros não

também existem problemas com redes de broadcast e multicast não confiável

Propriedades na difusão

✓ confiabilidade valem tanto para broadcast como multicast

✓ mensagem deve ser recebida por todos os nodos operacionais

✓ ordenamento consistente

✓ diferentes mensagens enviadas para nodos diferentes são entregues na mesma ordem em todos os nodos

ordenamento consistente é diferente de ordenamento temporal

✓ preservação de causalidade

✓ a ordem na qual mensagens são entregues é consistente com a relação causal de envio das mensagens

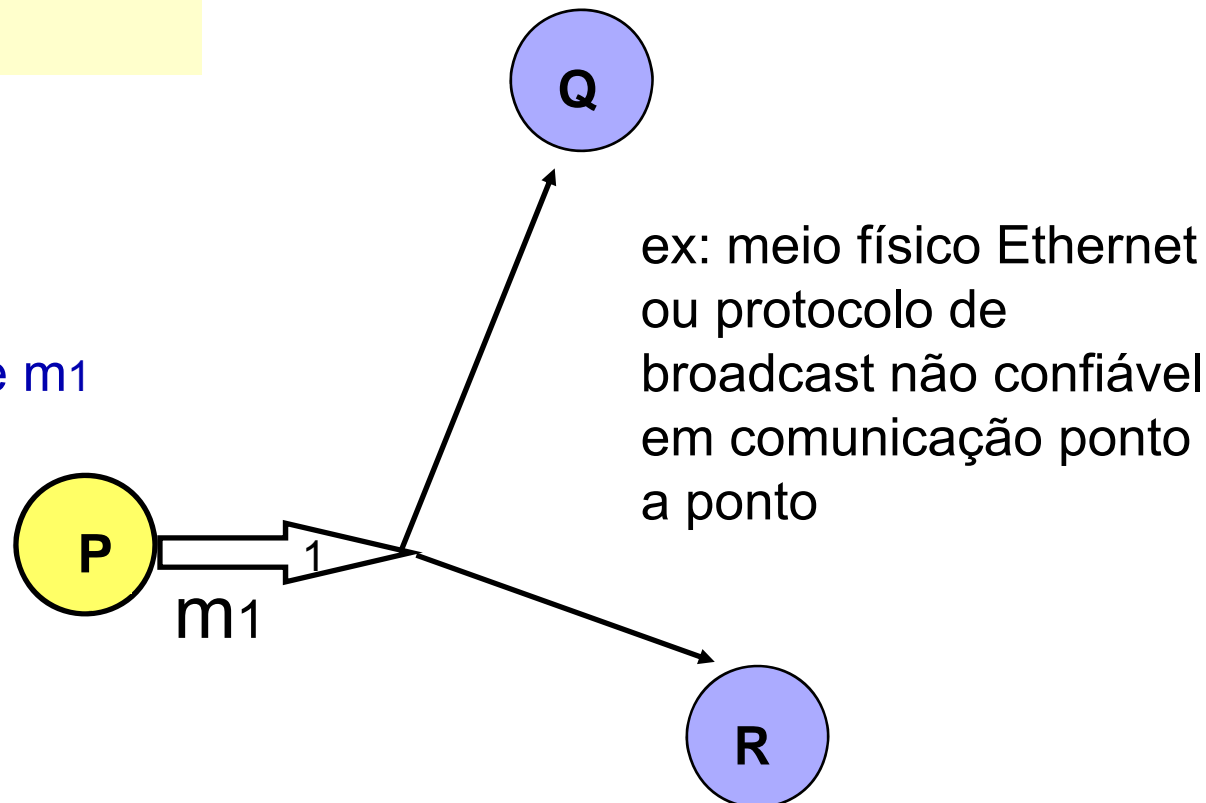
mensagens sem relação causal poderiam ser entregues em qualquer ordem

Trans

- ✓ primitiva confiável baseada em broadcast não confiável

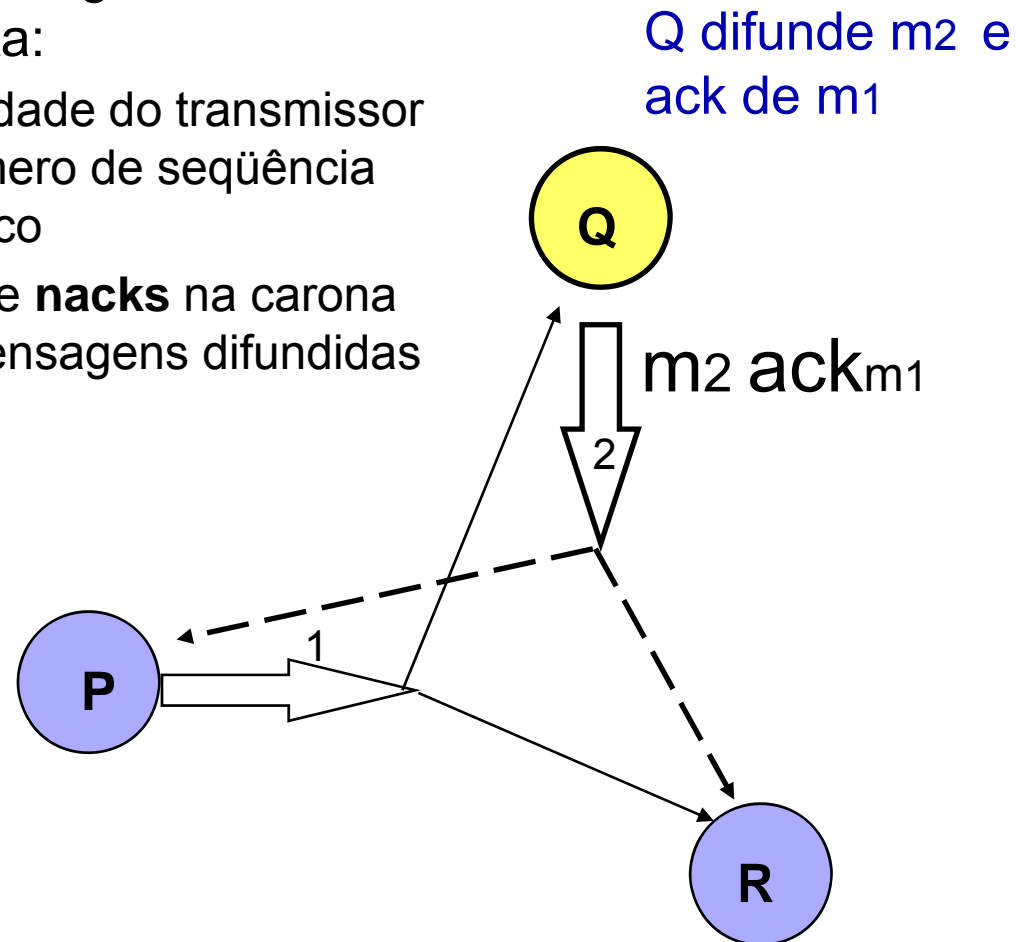
Melliar-Smith, Moser e
Agrawala (1990)

P difunde m1



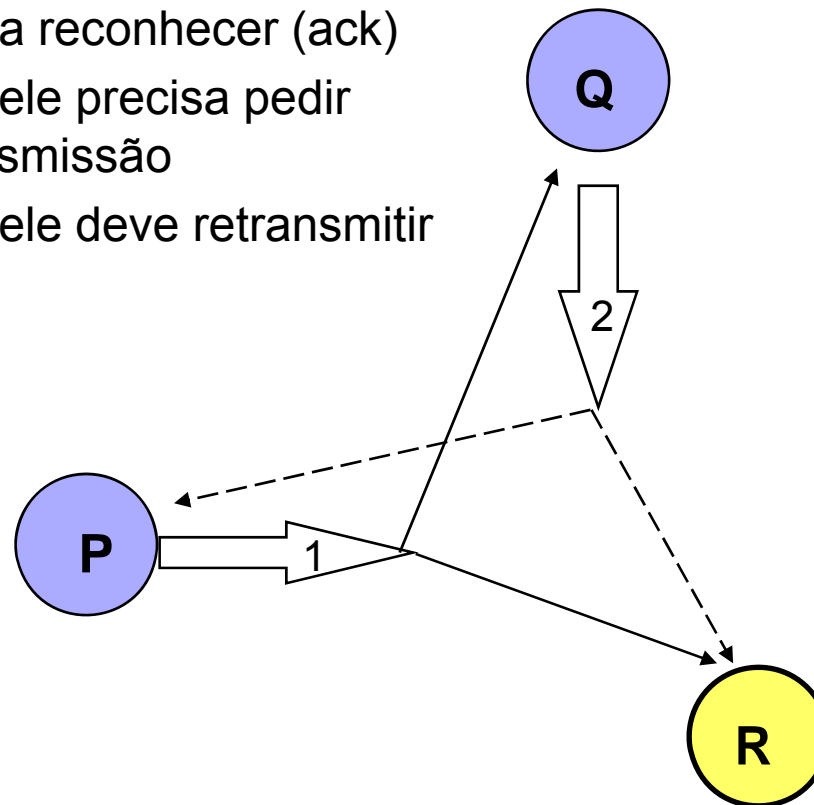
Trans

- ✓ cada mensagem transporta:
 - ✓ identidade do transmissor e número de seqüência unívoco
 - ✓ **acks** e **nacks** na carona de mensagens difundidas



Trans

- ✓ o receptor, a partir de **acks** e **nacks**, determina
 - ✓ que mensagens ele não precisa reconhecer (ack)
 - ✓ quais ele precisa pedir retransmissão
 - ✓ quais ele deve retransmitir

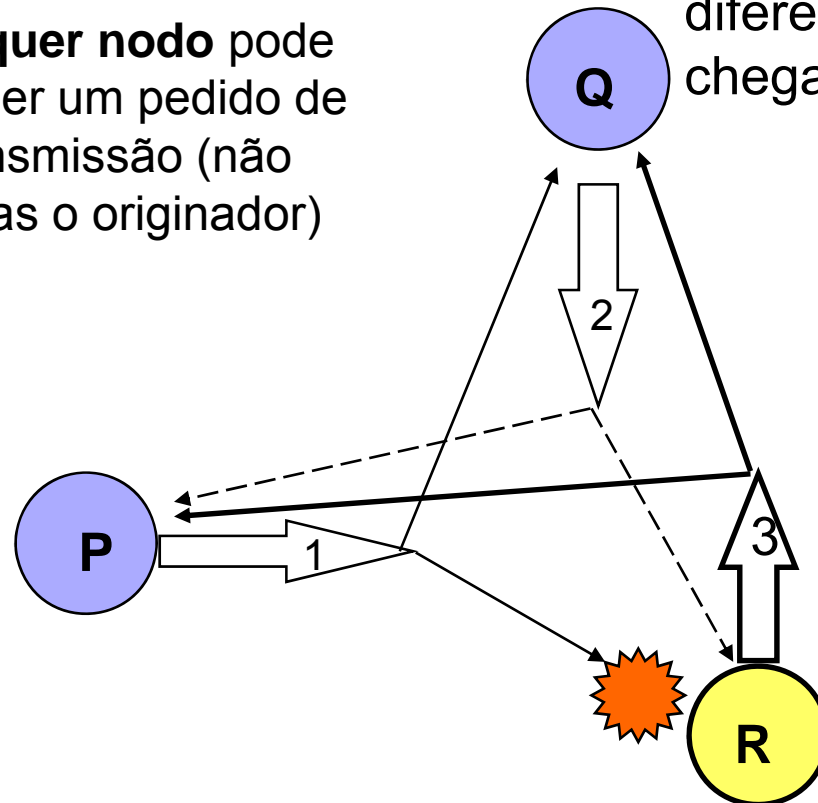


R recebeu m1 e m2
R não envia ack_{m1}
pois Q já enviou

Trans

- ✓ se o receptor **R** determina que não recebeu **m1**
 - ✓ deve pedir retransmissão
 - ✓ **qualquer nodo** pode atender um pedido de retransmissão (não apenas o originador)

sem ordenação: mensagens podem ser recebidas em cada nodo em uma ordem diferente (no exemplo **m1** chegará em R após **m2**)



R **não** recebeu m1
R envia nackm1
pedindo
retransmissão

m3 ack_{m2} nack_{m1}

ExemploTrans

✓ A, B, C, D = **mens** a, b, c, d = **acks**, a, b, c, d = **nacks**

✓ A

✓ A Ba

trans. de B reconhece A

✓ A Ba Cb

trans. de C reconhece B, não precisa rec. A

✓ A Ba Cb Dc

✓ A Ba Cb Dc Ecd

✓ A Ba Cb Dc Ecd Cb

trans. de E viu por Dc que não recebeu C

✓ A Ba Cb Dc Ecd Cb Fec

algum nodo retransmite C(sem novos acks)

—————→ **t**

Bibliografia

- ✓ JALOTE, P. Fault tolerance in distributed systems. Prentice Hall, Englewood Cliffs, New Jersey, 1994
- ✓ GÄRTNER, F. C. Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments. ACM Computing Surveys, Vol. 31, No. 1, March 1999.
- ✓ DEFAGO, X.; SCHIPER, A.; URBAN, P. Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey. ACM Computing Surveys, Vol. 36, No. 4, December 2004, pp. 372–421
- ✓ FREILING, GUERRAOUI, KUZNETSOV, The Failure Detector Abstraction . ACM Computing Surveys. Vol 43 Issue 2, Jan 2011

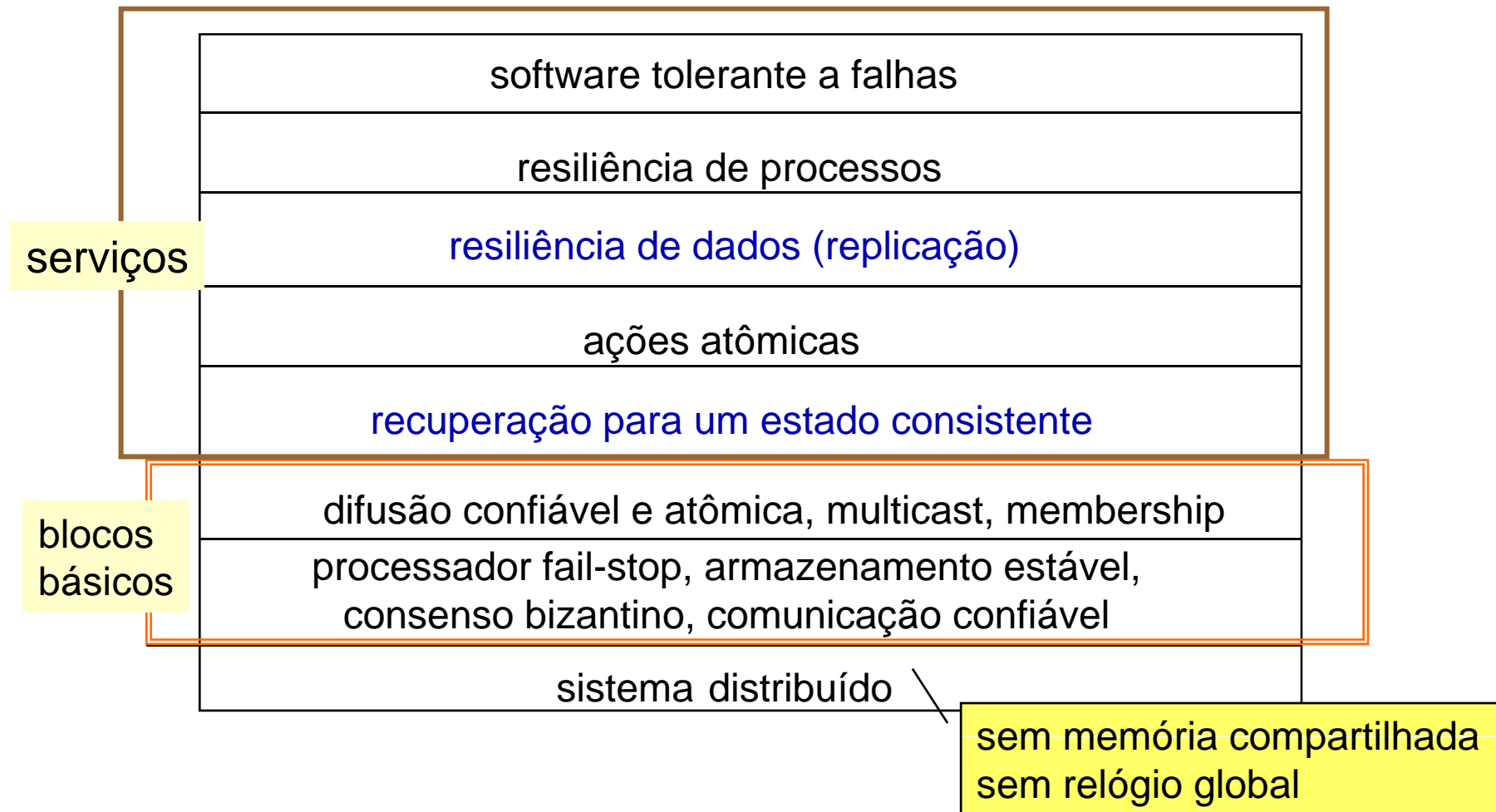
OBS: nas notas de aulas da disciplina, um roteiro básico pode ser encontrado com um resumo dos tópicos discutidos neste item.

TF em sistemas distribuídos: serviços

UFRGS

Taisy Silva Weber

Níveis - [Jalote 94]



Recuperação

- ✓ restaurar para um estado consistente

estado global do sistema inclui estados de diferentes processos executando em nodos diversos

- ✓ problemas:

- ✓ sistema distribuído

sem relógio global, sem memória comum

- ✓ estado consistente em sistema distribuído

- ✓ sistemas distribuídos convencionais

- ✓ usual recuperação por retorno

- ✓ checkpoint ou ponto de recuperação (PR)

- ✓ contém **toda** a informação de **todos** os processos executando no nodo

Recuperação por retorno

- ✓ checkpointing **assíncrono**
 - ✓ eficiente no avanço
 - ✓ demorado no rollback
- ✓ checkpointing **coordenado**
 - ✓ checkpointing coordenado em todos os nodos
 - ✓ o conjunto dos PRs representa um estado consistente para o sistema
- ✓ checkpointing **induzido por comunicação**
 - ✓ info de controle de carona nas mensagens normais
- ✓ rollback-recovery
 - ✓ com log de eventos não determinísticos

não coordenado nos diferentes nodos

ELNOZAHY, E. N. ; et alli. **A Survey of Rollback-Recovery Protocols in Message-Passing Systems**. ACM Computing Surveys, Sept. 2002, pp. 375–408.

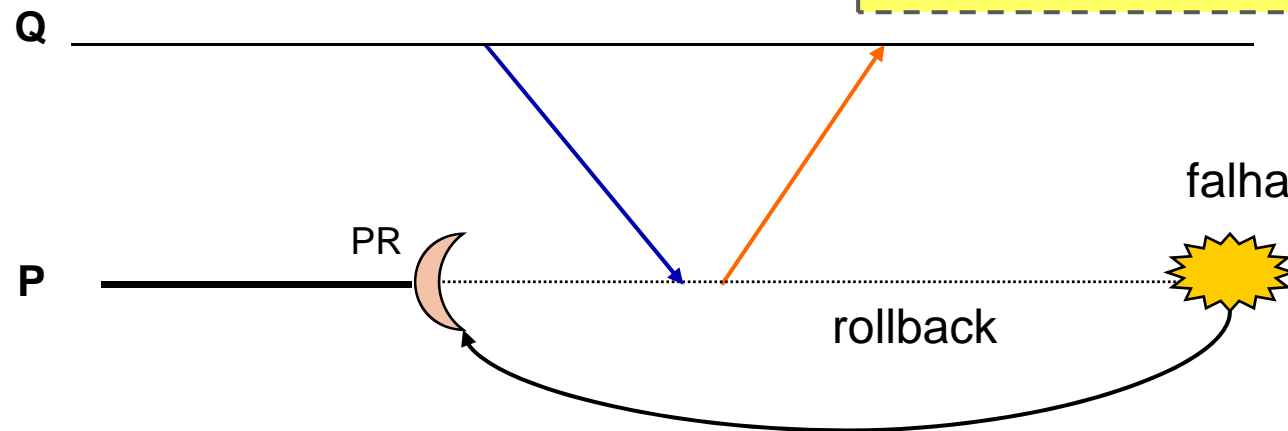
Rollback

sem problemas em um processo isolado

mas em um SD processos trocam mensagens

msg perdida: receptor retornou para um ponto anterior ao recebimento da msg

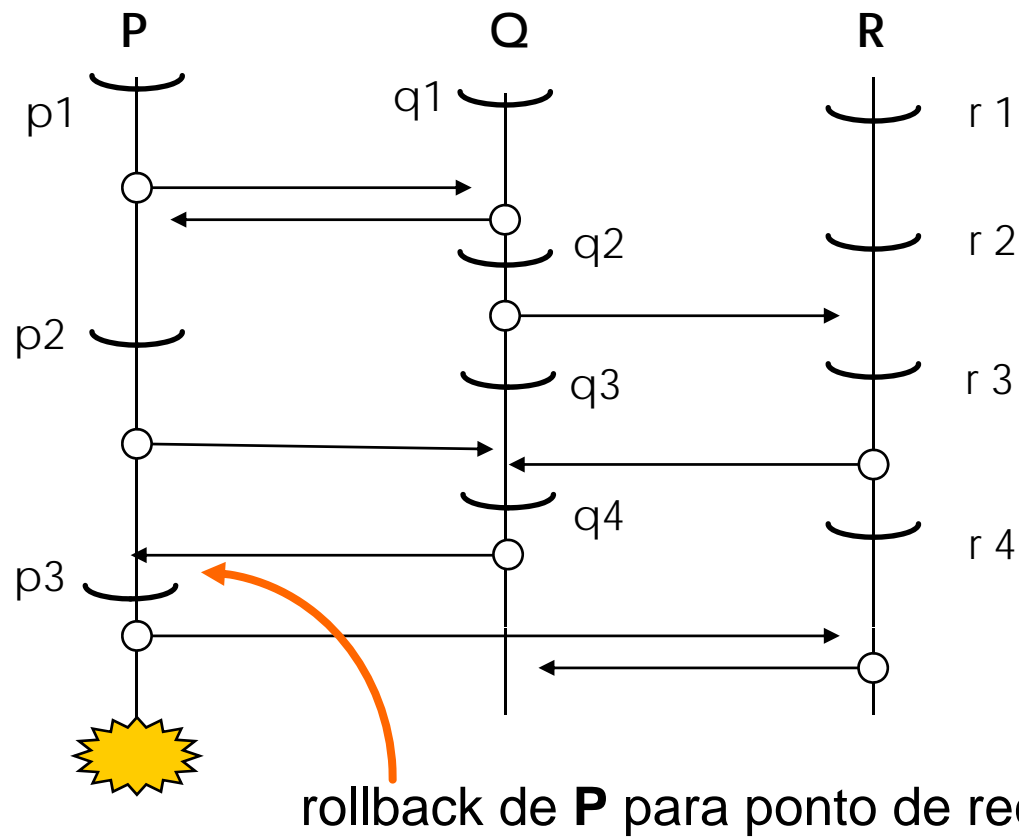
msg órfã: transmissor retornou para um ponto anterior ao envio



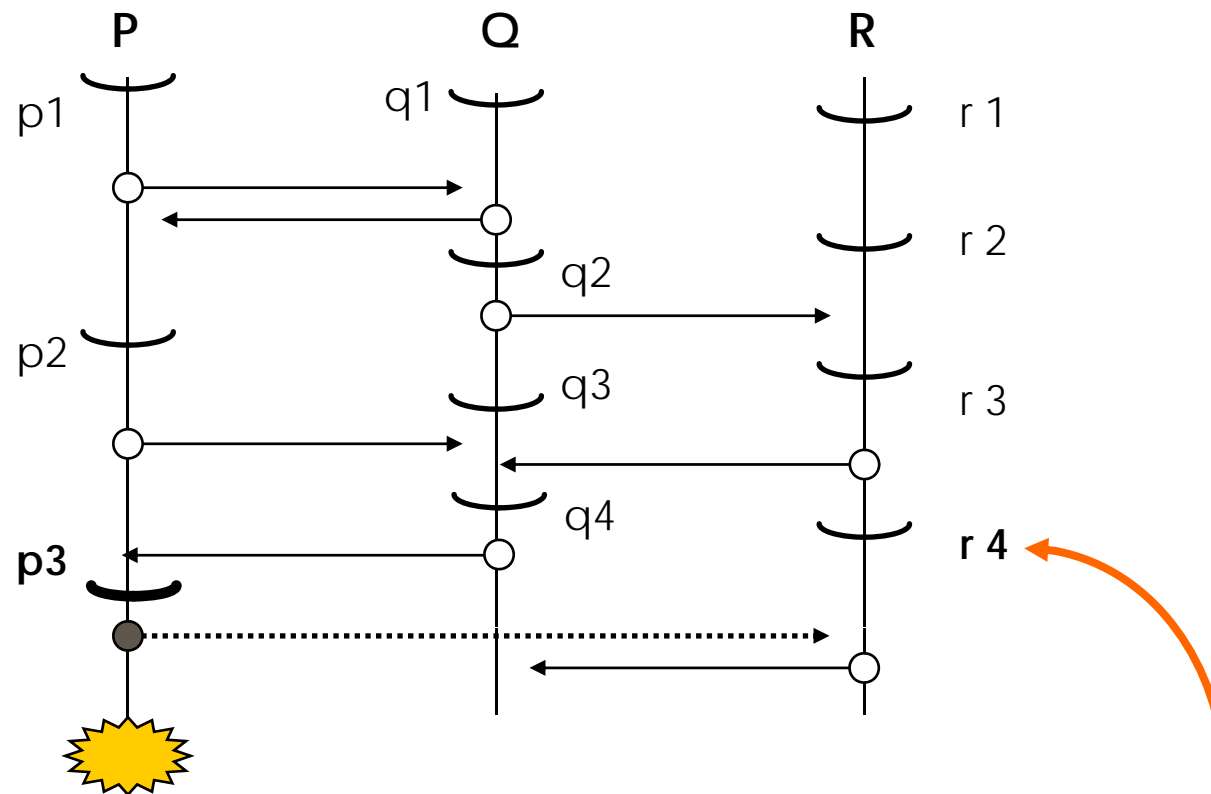
- ✓ mensagens perdidas: msg enviadas e não recebidas
- ✓ mensagens órfãs: msg recebidas que não foram enviadas

Linha de recuperação

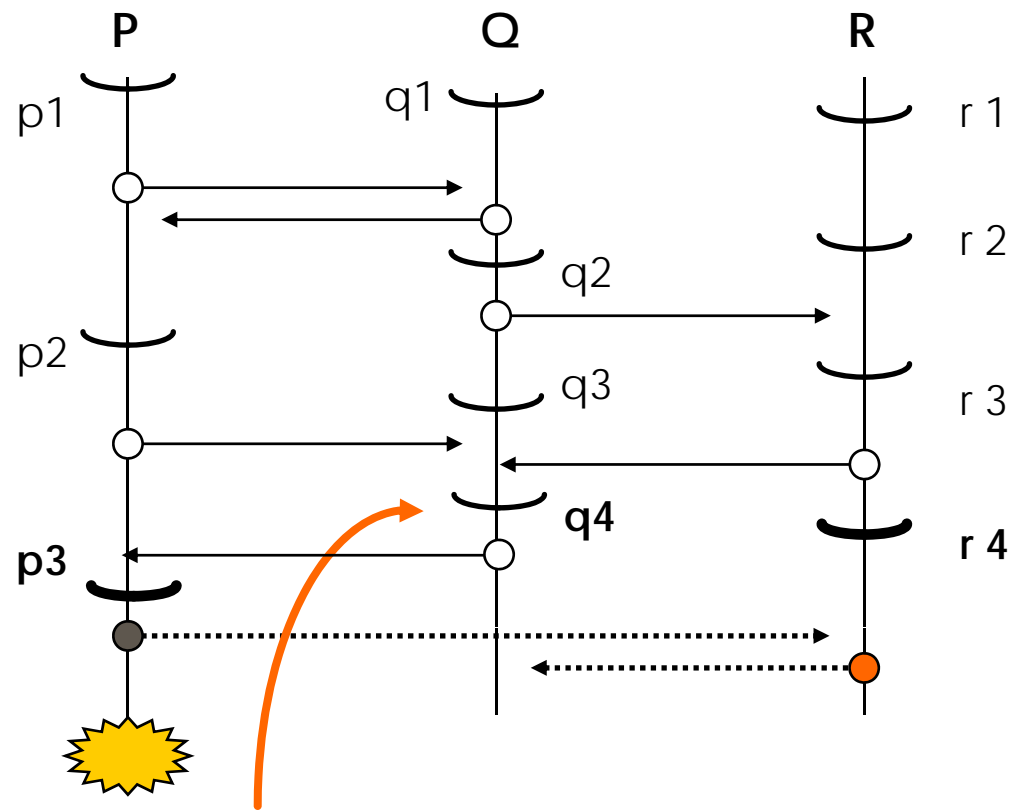
conjunto de *ckpoints*
com apenas **um**
checkpoint por
processo, **sem** msgs
órfãs, **sem** msgs
perdidas



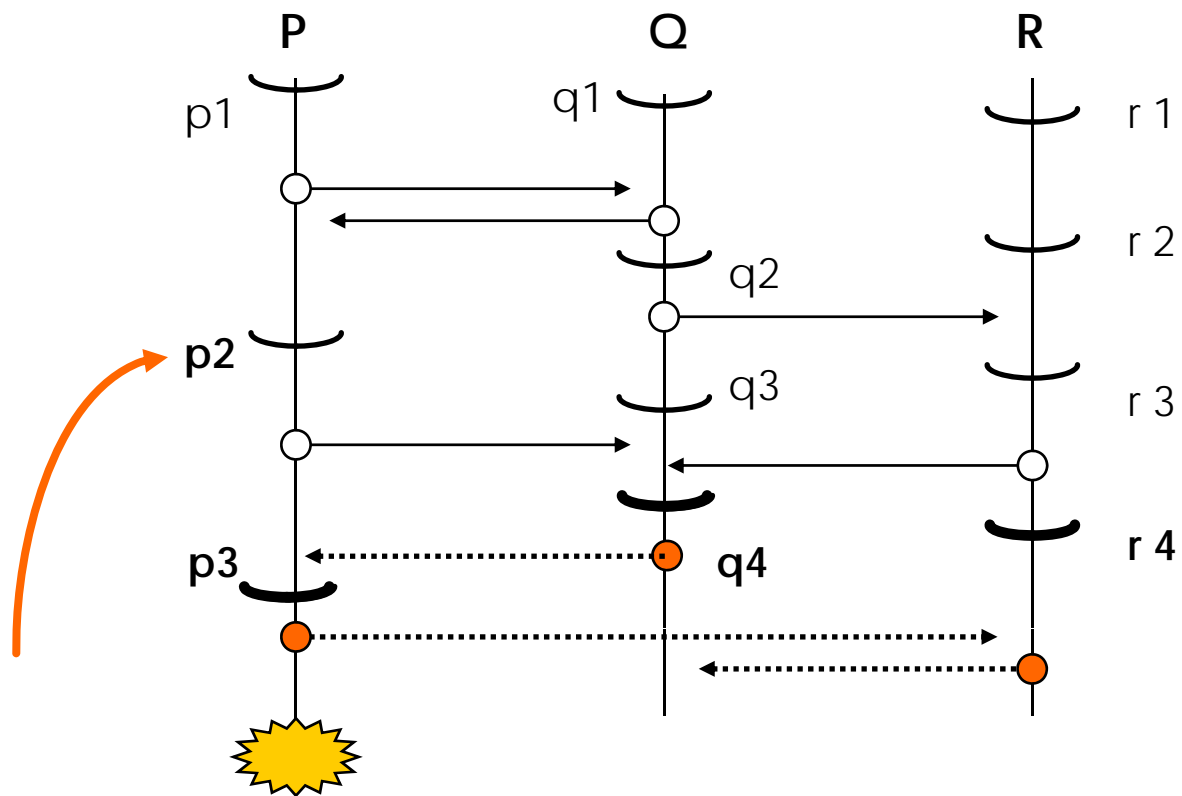
Linha de recuperação



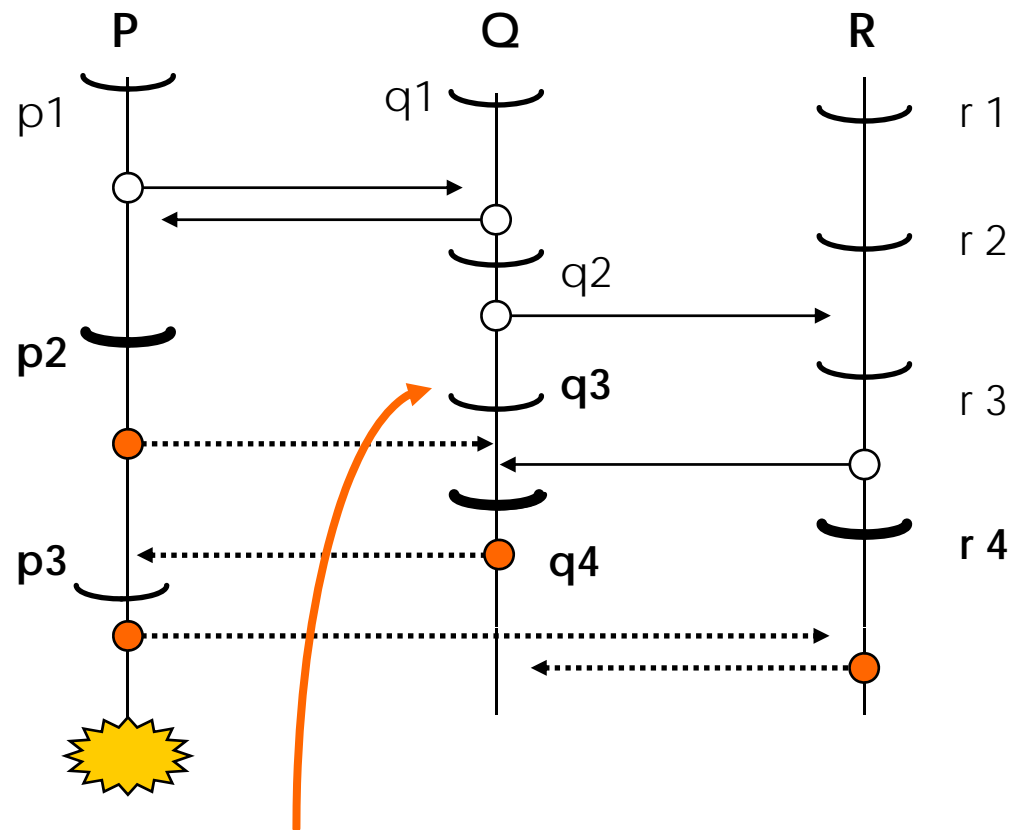
Linha de recuperação



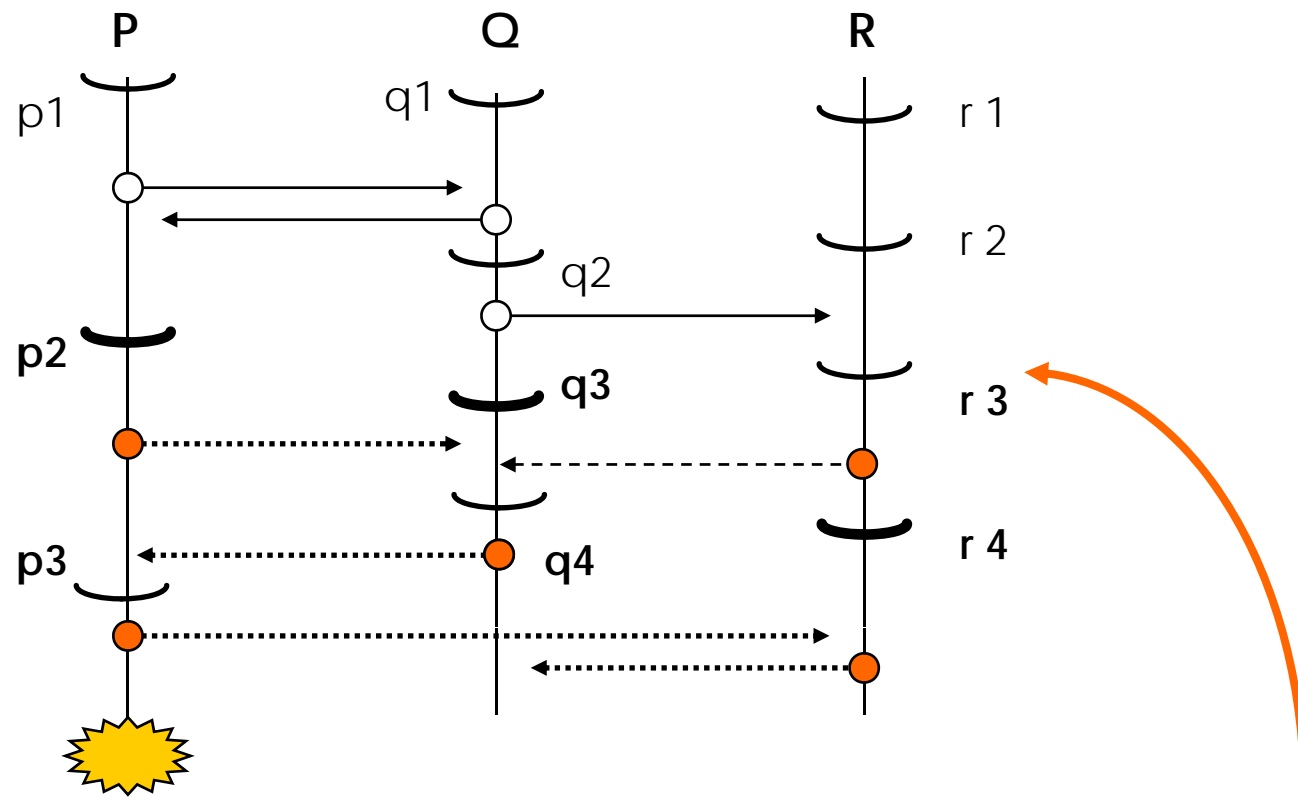
Linha de recuperação



Linha de recuperação

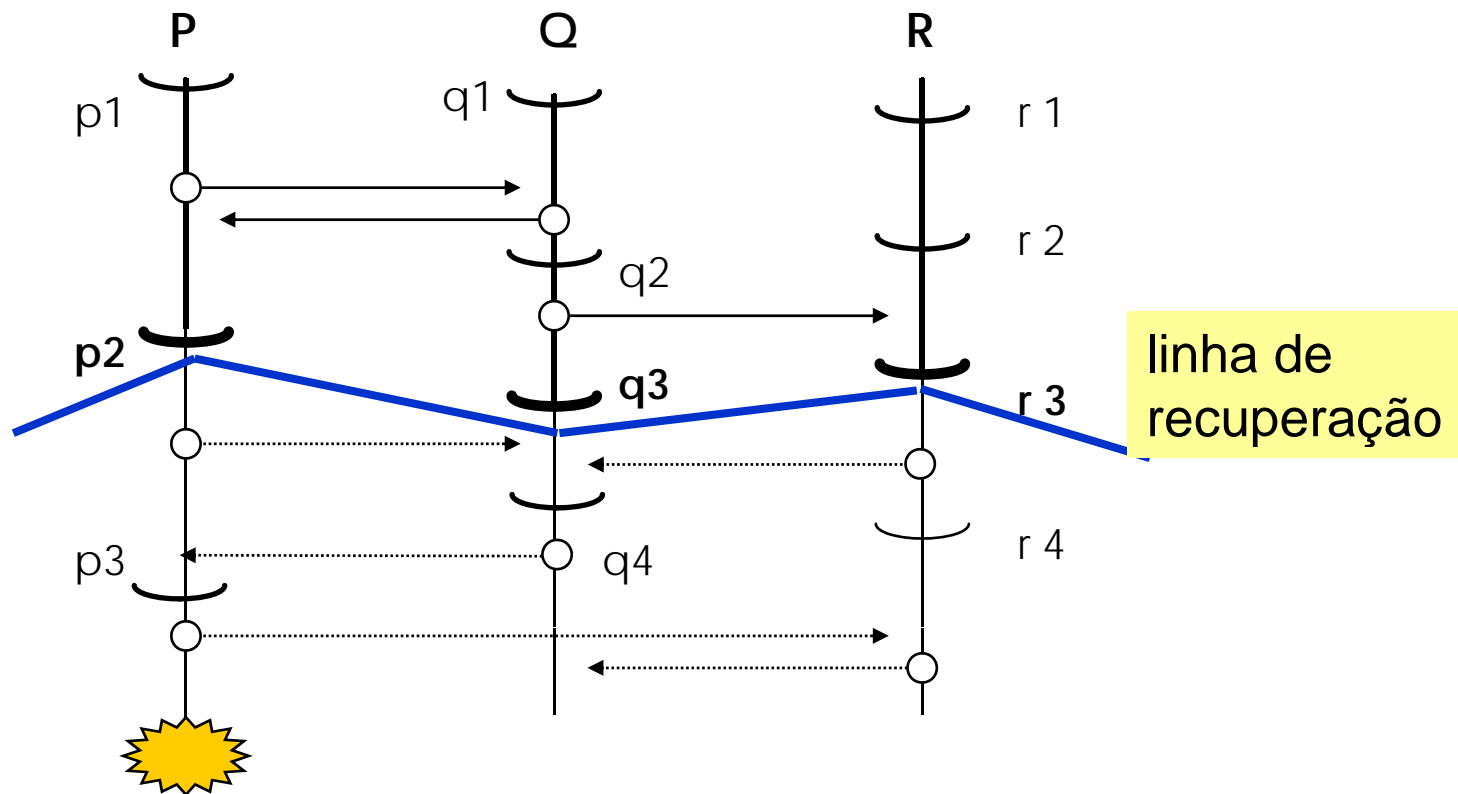


Linha de recuperação



Linha de recuperação

apenas um PR por processo, sem órfãos, sem perdas



Efeito dominó

avalanche de rollbacks que podem ocorrer durante a recuperação

- ✓ perigo real
 - ✓ sistemas com estabelecimento de pontos de recuperação independentes e sem restrições a troca de mensagens
 - ✓ pode provocar volta a estado inicial
 - ✓ típico em ckp não coordenado
- ✓ evitando **efeito dominó**
 - ✓ coordenação de checkpointing
 - ✓ restrição a comunicação



Recuperação baseada em logs

- ✓ premissa
 - ✓ todos os eventos não determinísticos podem ser identificados e *logados*
 - ✓ eventos não determinísticos podem ser modelados como recepção de mensagem
 - ✓ envio de mensagem é um evento determinístico
- ✓ recuperação
 - ✓ usa ckps e logs para voltar precisamente pelo mesmo caminho ao estado anterior à falha
 - ✓ computação não é perdida
 - ✓ mas deve se ter cuidado com respostas (msgs) que alteram o mundo exterior

Replicação de dados

- ✓ dados replicados em vários nodos
 - ✓ a queda de um ou mais nodos não impede **acesso** aos dados
- ✓ novos problemas
 - ✓ consistência
- ✓ serializabilidade
- ✓ execução concorrente nas réplicas deve ser equivalente a execução correta nos dados lógicos
 - ✓ (como se fosse **cópia única**)
- ✓ replicação deve ser **transparente** ao usuário

cópias diferentes de um objeto devem ser mutuamente consistentes entre si

critério de correção

Tipos de falhas

- ✓ falhas nos nodos
 - ✓ cópias no nodo ficam inacessíveis
 - ✓ demais cópias são acessíveis e deve ser garantido o acesso e o critério de serializabilidade
- ✓ **particionamento da rede**
 - ✓ particionamento é difícil de tratar
 - ✓ geralmente são implementadas algumas soluções parciais para casos particulares

modelo físico

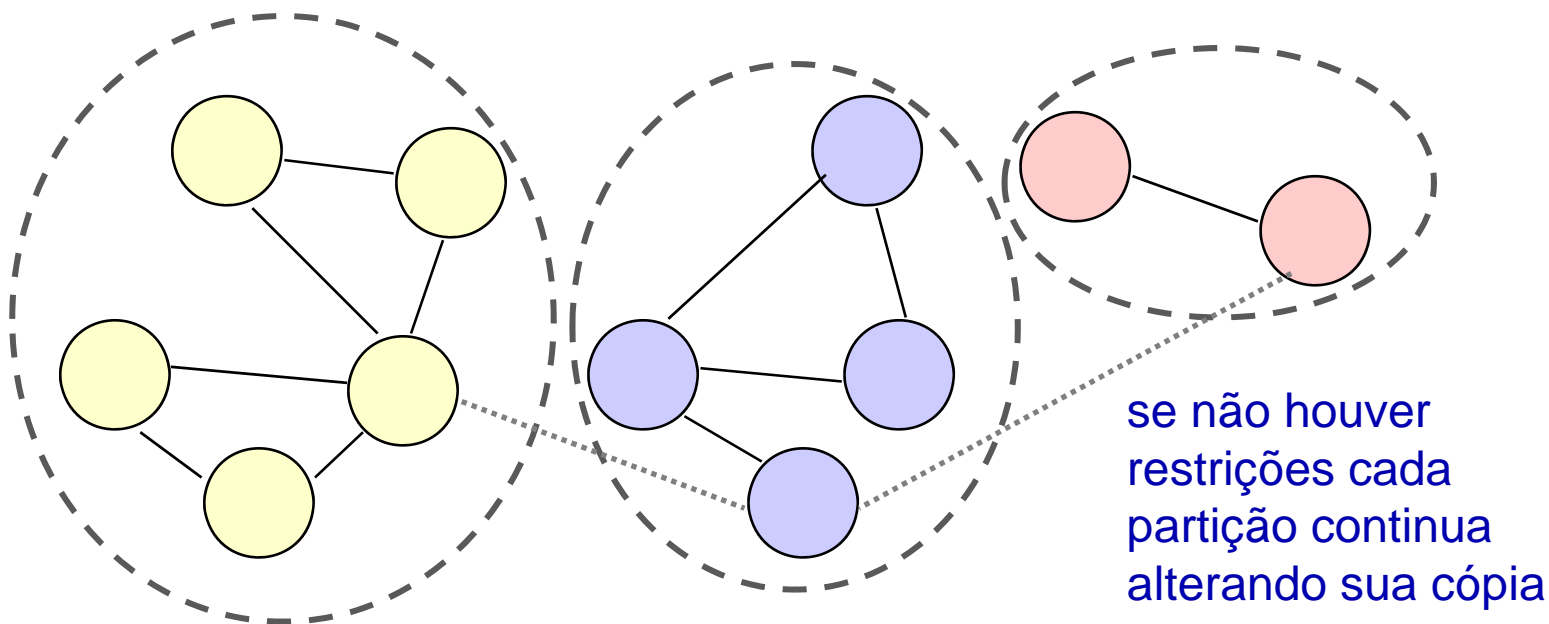
exemplo: cada **partição** inibe alterações na sua réplica caso não contenha a maioria dos nodos

Particionamento

modelo físico

partição com maior
número de nodos

3 partições isoladas que continuam
recebendo requisições de escrita
dos clientes



o maior problema é garantir a serializabilidade
sem comunicação entre as partições

Estratégias

- ✓ protocolo de controle de réplicas

- ✓ otimista

- ✓ sem restrição

esperança de que operações em partições diferentes não vão conflitar

- ✓ réplicas podem divergir e usuários podem ver inconsistência

- ✓ pessimista

- ✓ garantia de *consistência forte*

- ✓ réplicas nunca divergem

- ✓ tipos de abordagem pessimista

- ✓ cópia primária
 - ✓ réplicas ativas
 - ✓ votação

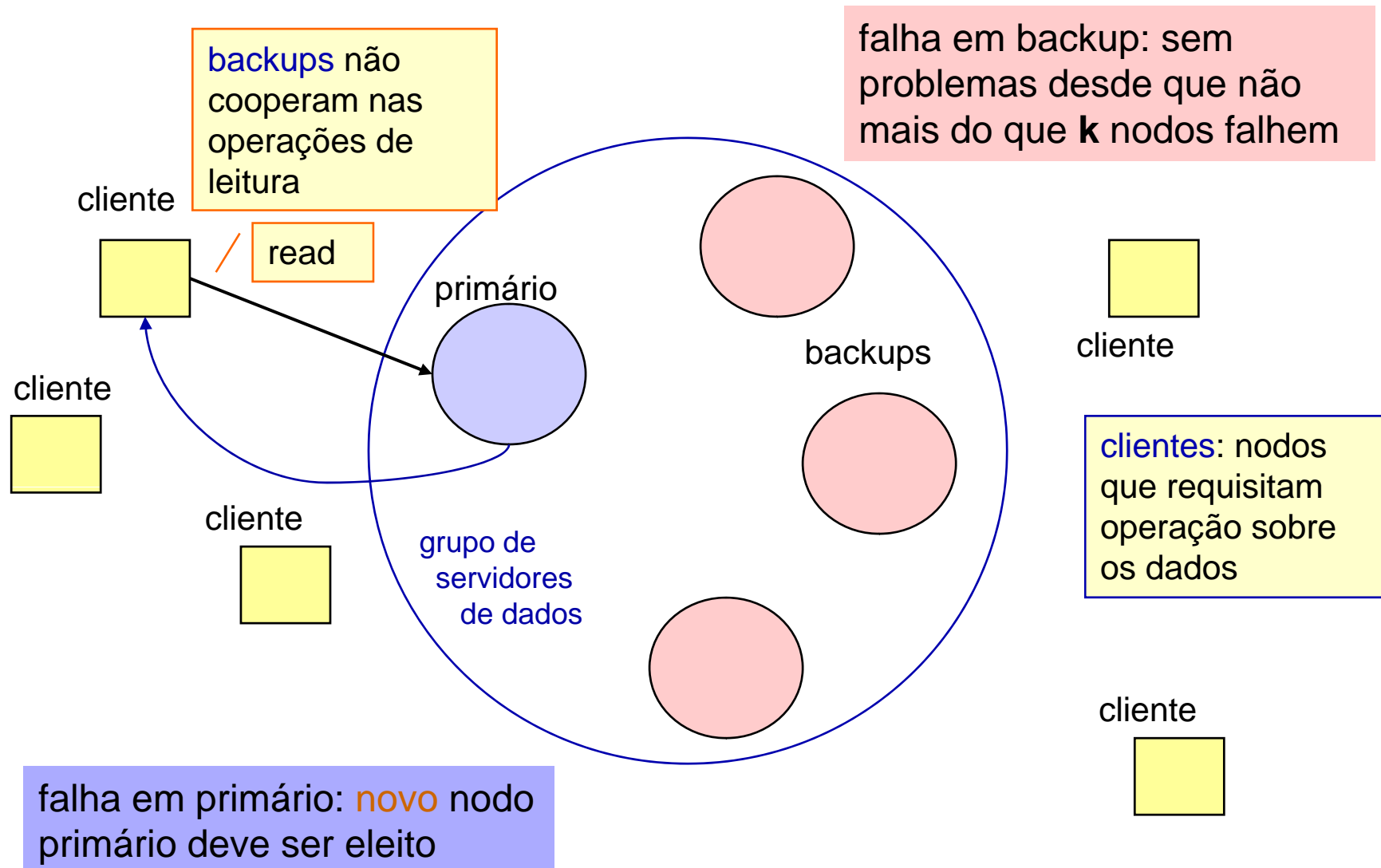
Abordagem otimista

- ✓ estratégia antiga, mas interesse crescente
 - ✓ sistema móveis e Internet
- exemplos: DNS, CVS
- ✓ vantagens
 - ✓ aplicação para sistemas de larga escala
 - ✓ pode usar comunicação epidêmica quando a topologia é desconhecida
 - ✓ mantém disponibilidade (as custas de inconsistências eventuais)
 - ✓ requer pouca sincronização entre réplicas
 - ✓ permite nodos operarem autonomamente
 - ✓ sem necessidade de estar sempre conectado

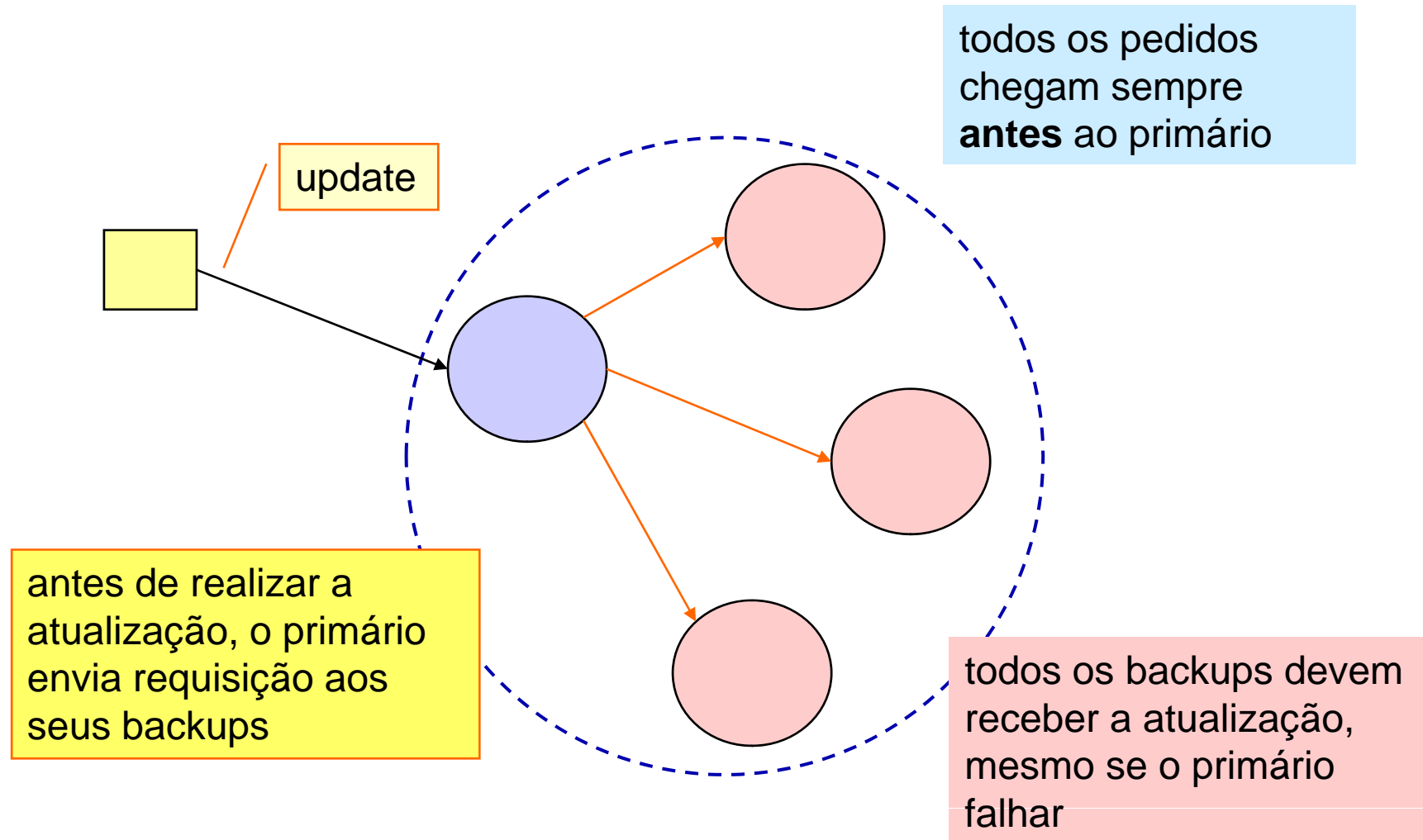
Desafios

- ✓ redes locais
 - ✓ uso popular de primário backup
 - ✓ escala pequena
- ✓ sistemas intensivos quanto a dados
 - ✓ sem necessidade de garantias fortes
 - ✓ redes de entrega de conteúdos
 - ✓ só um nodo altera dados, outros mantêm cópias,
 - ✓ P2P (média taxa de atualização)
 - ✓ Data Grids (raras atualizações)
 - ✓ com garantias fortes (ACID)
 - ✓ bancos de dados distribuídos (não escala)

Cópia primária: abordagem pessimista

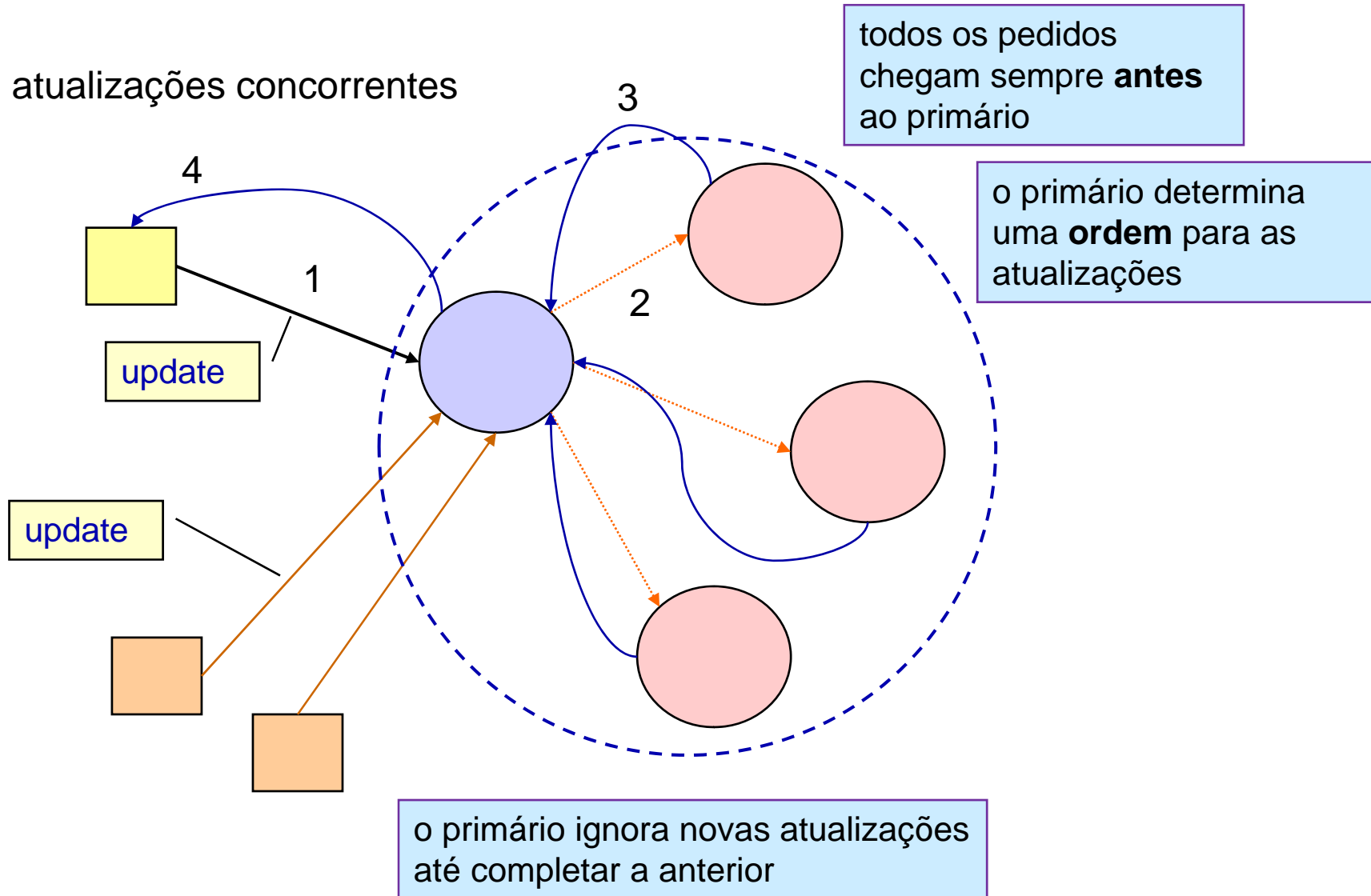


Atualizações



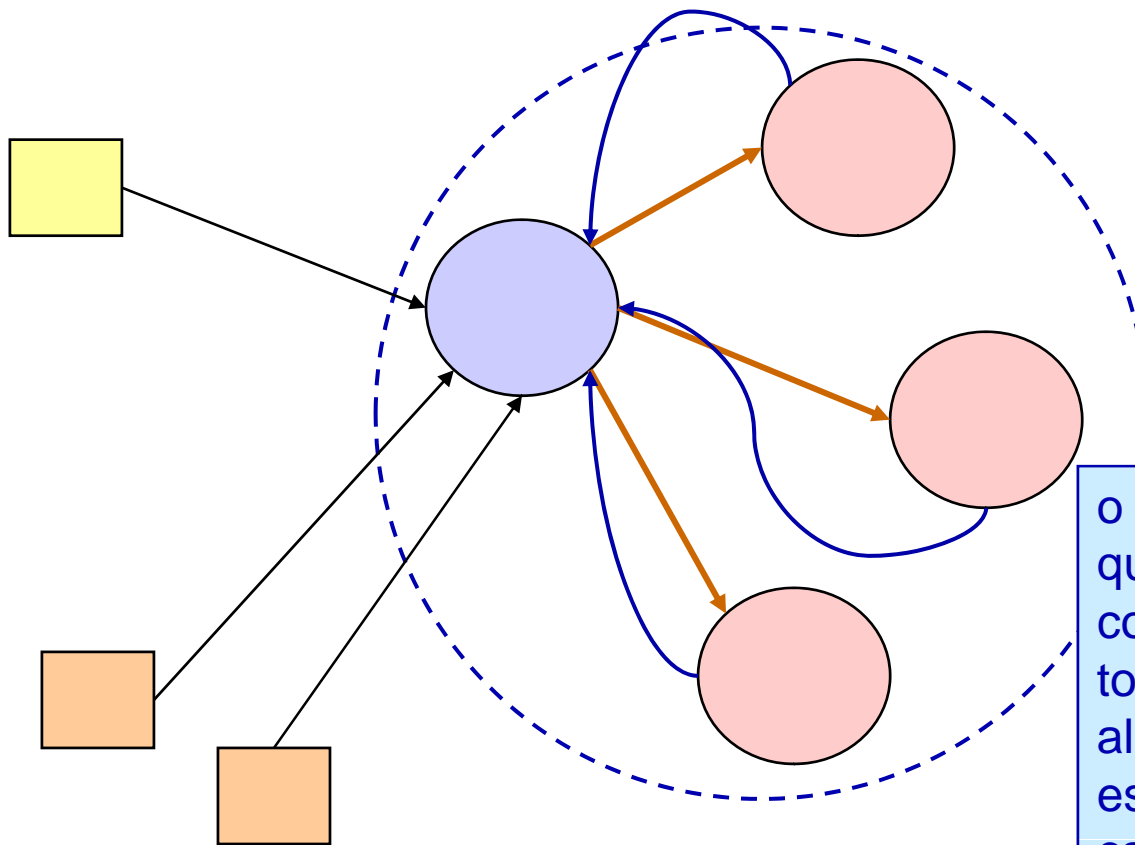
Ordenação de atualizações

atualizações concorrentes



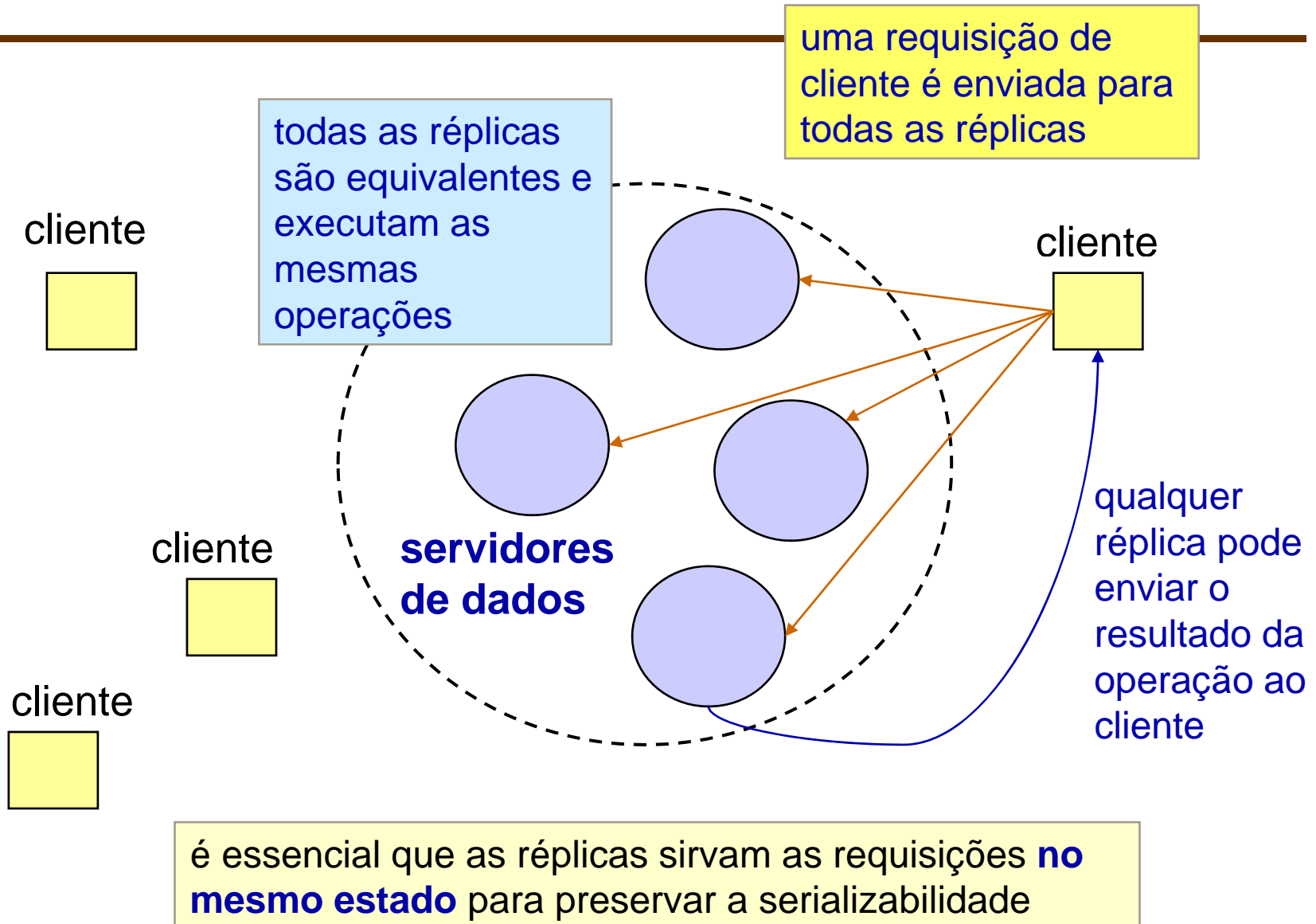
Confiabilidade

a comunicação entre primário e backups pode ser suportada por um **protocolo de multicast confiável**



o protocolo vai garantir que, uma vez que a comunicação iniciou, todos os backups vão alcançar o mesmo estado, mesmo em caso de queda do primário

Réplicas ativas



Serializabilidade com réplicas ativas

- ✓ assumido
 - ✓ se as réplicas estão no mesmo estado e recebem as requisições na **mesma ordem**, então vão produzir os mesmos resultados
- ✓ devem ser satisfeitas propriedades de:

- ✓ consenso

- ✓ ordem

todas as réplicas operacionais devem receber todas as requisições

todas as réplicas operacionais executam as requisições na mesma ordem

multicast atômico garante consenso (confiabilidade) e ordem

Bibliografia

- ✓ JALOTE, P. **Fault tolerance in distributed systems.** Prentice Hall, Englewood Cliffs, New Jersey, 1994
- ✓ ELNOZAHY, E. N. ; et alli. **A Survey of Rollback-Recovery Protocols in Message-Passing Systems.** ACM Computing Surveys, Vol. 34, No. 3, September 2002, pp. 375–408.
- ✓ YASUSHI SAITO, MARC SHAPIRO. **Optimistic Replication.** ACM Computing Surveys, Vol. 37, No. 1, March 2005, pp. 42–81.

Clusters de alta disponibilidade

UFRGS

Taisy Silva Weber

Cluster

- cluster ou agregado
 - computadores com múltiplos processadores
 - termo usado para vasta gama de configurações
 - número variável de nodos de computação convencionais: de 2 nodos a poucos milhares
 - opcionalmente alguns dispositivos de armazenamento compartilhados
 - interconexões de alta velocidade

exemplo de arquitetura tolerante a falhas
exemplo da aplicação de conceitos de sistemas distribuídos

Definição

- **coleção de computadores** que trabalham visando prover um sistema de grande capacidade.
 - deve ser tão **fácil de programar** e de gerenciar como um único computador de grande porte.
- **vantagens**
 - pode **crescer muito** mais do que um único computador (*escalabilidade*)
 - pode **tolerar defeitos em nodos** e continuar a oferecer serviços (*failover*)
 - pode ser construído a partir de **componentes de baixo custo**

Sistemas distribuídos versus cluster

- cluster são sistemas distribuídos
 - sem memória compartilhada
 - sem relógio global
 - comunicação por troca de mensagens
 - mas tem a vantagem da proximidade física
- técnicas de TF em sistemas distribuídos são úteis em clusters
 - comunicação de grupo e membership
 - checkpointing, logging e recuperação
 - tratamento de particionamento

Tipos

- implementação
 - por hardware: mais eficiente, pouco adaptável
 - por software: menor custo
- objetivos
 - alto desempenho
 - balanceamento de carga
 - alta disponibilidade

alguns autores falam de mais um tipo: **disponibilidade contínua**

vários objetivos podem ser **combinados**

Combinações de tipos

- bons esquemas de **balanceamento de carga** podem contribuir para aumentar a disponibilidade
- em cluster de **alto desempenho**:
 - nodos críticos podem compor um núcleo de alta disponibilidade
 - todos os nodos podem contribuir mantendo réplicas de dados ou processos, checkpoints e logs uns dos outros
- **redundância** inerente no cluster facilita implementar tolerância a falhas

HA-Cluster

- alta disponibilidade
 - tempo de inicialização após falha (*failover*) pode variar de poucos minutos até uma hora
 - aplicações em sistemas de missão crítica
 - servidores primário e backups
- disponibilidade contínua
 - tempo de *failover* na ordem de 10 segundos

primário e backup executam mesmos processos (warm backup)

Compartilhamento de disco

- sistemas de **disco compartilhado**:
 - necessitam de um gerenciador de bloqueio
 - evitar conflitos devido a requisições de acesso simultâneo a arquivos
 - um arquivo sendo escrito por um nodo não pode ser aberto para escrita em outro nodo
- sistemas de armazenamento **não compartilhado**:
 - cada nodo é independente
 - toda a interação é por troca de mensagens

Sinal de vida (*heartbeat*)

- mensagem **periódica** enviada de um processo a outro para indicar que continua operacional
 - detecção de falhas: ausência de **heartbeats**

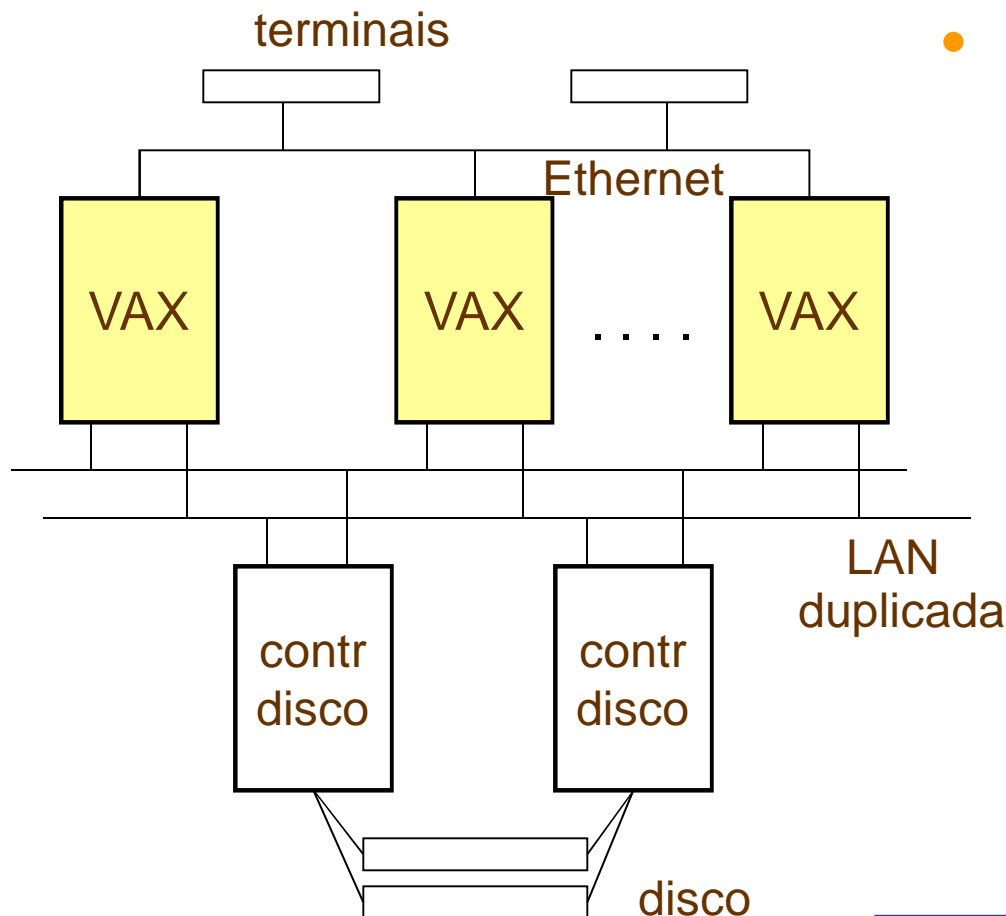
heartbeats são esperados a cada poucos **segundos**

- modelo **fail-stop**

assume que se um nodo pára de enviar sinais, ele efetivamente não envia mensagens, nem altera dados no armazenamento estável

- técnica antiga
 - muito usada antes mesmo dos primeiros clusters (Tandem,...)

Arquitetura VAX Cluster



- VAXcluster da Digital
 - primeiro cluster de sucesso
 - formado por nodos VAX
- se um VAX colapsa
 - todos os processos nele caem
 - serviços precisam ser reiniciados em outro servidor do cluster

não é transparente ao usuário

tempo longo de recuperação

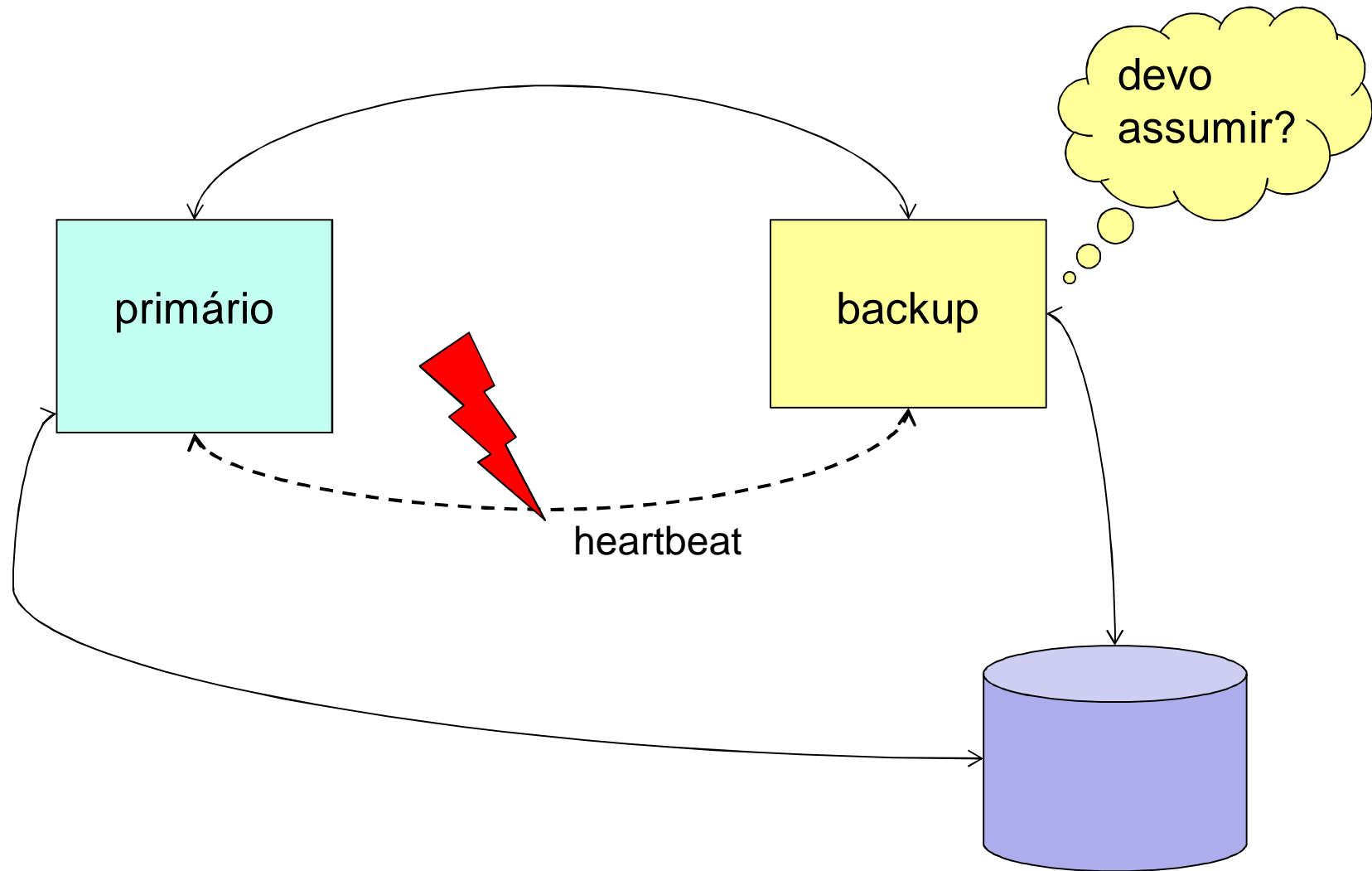
Disponibilidade em HA-clusters

- qual a disponibilidade efetivamente alcançada?
 - promessa de 99,99%
 - o VAXCluster não chegava a isso
- como avaliar?
 - experimentalmente por injeção de falhas
 - analiticamente através de modelos
 - ou durante operação levantando registros de falha (em logs por exemplo) e analisando

Problemas

- split-brain
 - um computador detecta o outro como defeituoso e assume as funções de primário
- modelo fail-stop
 - assumido pelos fabricantes mas raramente implementado
- particionamento

split brain



Bibliografia

- Birman, K. *Building secure and reliable network applications*. Manning Publications Co, Greenwich, 1996
- Vogels, W. *The Design and Architecture of the Microsoft Cluster Service - A Practical Approach to High-Availability and Scalability*, FTCS-IEEE, 1998
- Azagury, Alain et al. *Highly Available Cluster: a Case Study*. FTCS-IEEE, 1994
- Hughes-Fenchel, Gary. *A Flexible Clustered Approach to High Availability*. FTCS-IEEE, 1997
- links de fabricantes