Computação Distribuída

Odorico Machado Mendizabal

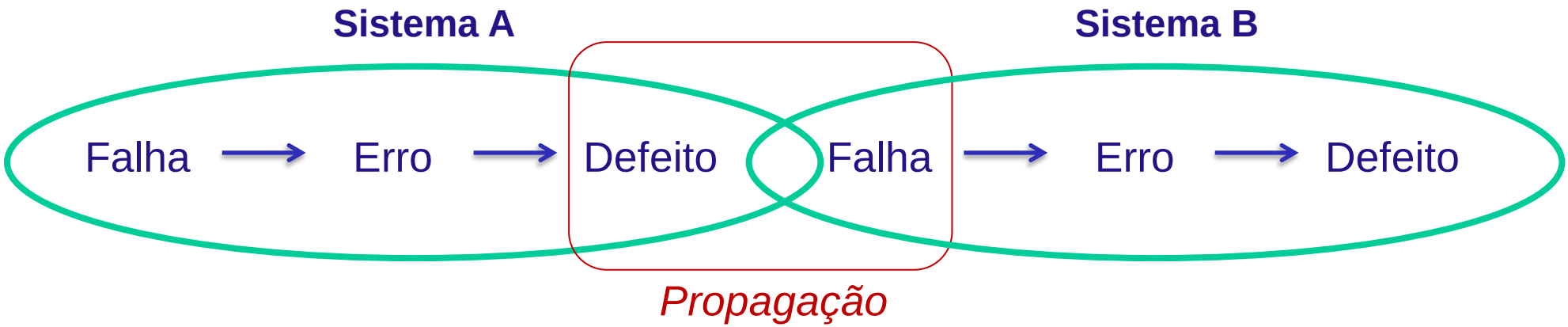
Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

Tolerância a Falhas

Motivação

- Falhas em sistemas simplesmente acontecem
  - Problemas durante o desenvolvimento do sistema
  - Sistemas mais complexos
  - Falhas físicas (HW ou fenômenos ambientais)
  - Falhas ocasionadas por atuação dos usuários

- Consequências para o sistema:
  - Desvio do comportamento esperado (correto)



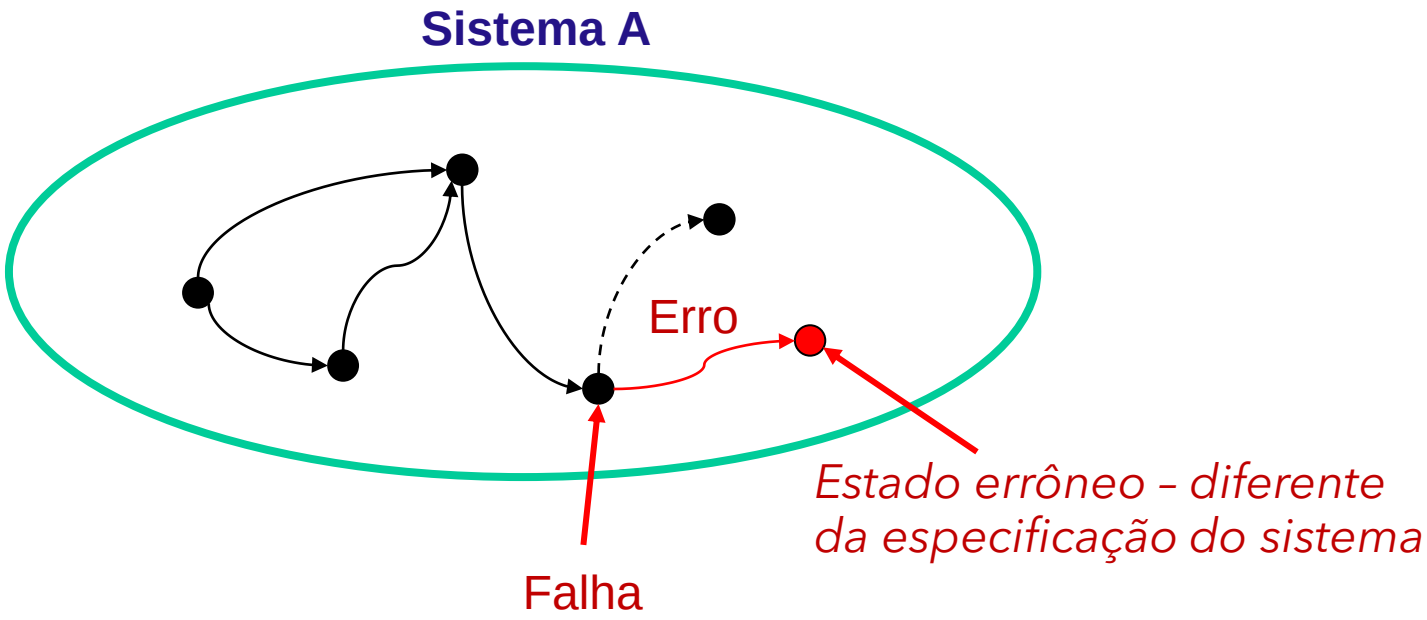
Tipos de Falhas em Sistemas Distribuídos

Classe de falha	*Afeta*	*Descrição*
*Fail-stop (*Parada*)*	Processo	Processo para e permanece parado. Outros processos podem detectar este estado.
Crash (Colapso)	Processo	Processo para e permanece parado. Outros processos podem não conseguir detectar este estado.
Omission (Omissão)	Canal	Uma mensagem inserida no buffer de saída nunca chega ao *buffer* de chegada no destino.
Send-omission	Processo	Um processo completa um envio, mas mensagem não é enviada ao *buffer* de saída
Receive-omission	Processo	Uma mensagem chega ao *buffer* de entrada no destino, mas processo não a recebe
Arbitrary (Byzantine)	Processo ou Canal	Processo/canal exibe comportamento arbritátio: ele pode enviar/transmitir mensagens arbritárias em tempos arbritátios, omitir commit; um processo pode parar ou executar um passo incorreto.

Tolerância a Falhas – Intuição

- Detecção
  - Identificação da presença de erros no sistema
  - Percepção sobre o desvio do comportamento esperado

- Mascaramento
  - Esconde um estado do sistema com erros (e possivelmente falhas), evitando que um defeito se manifeste



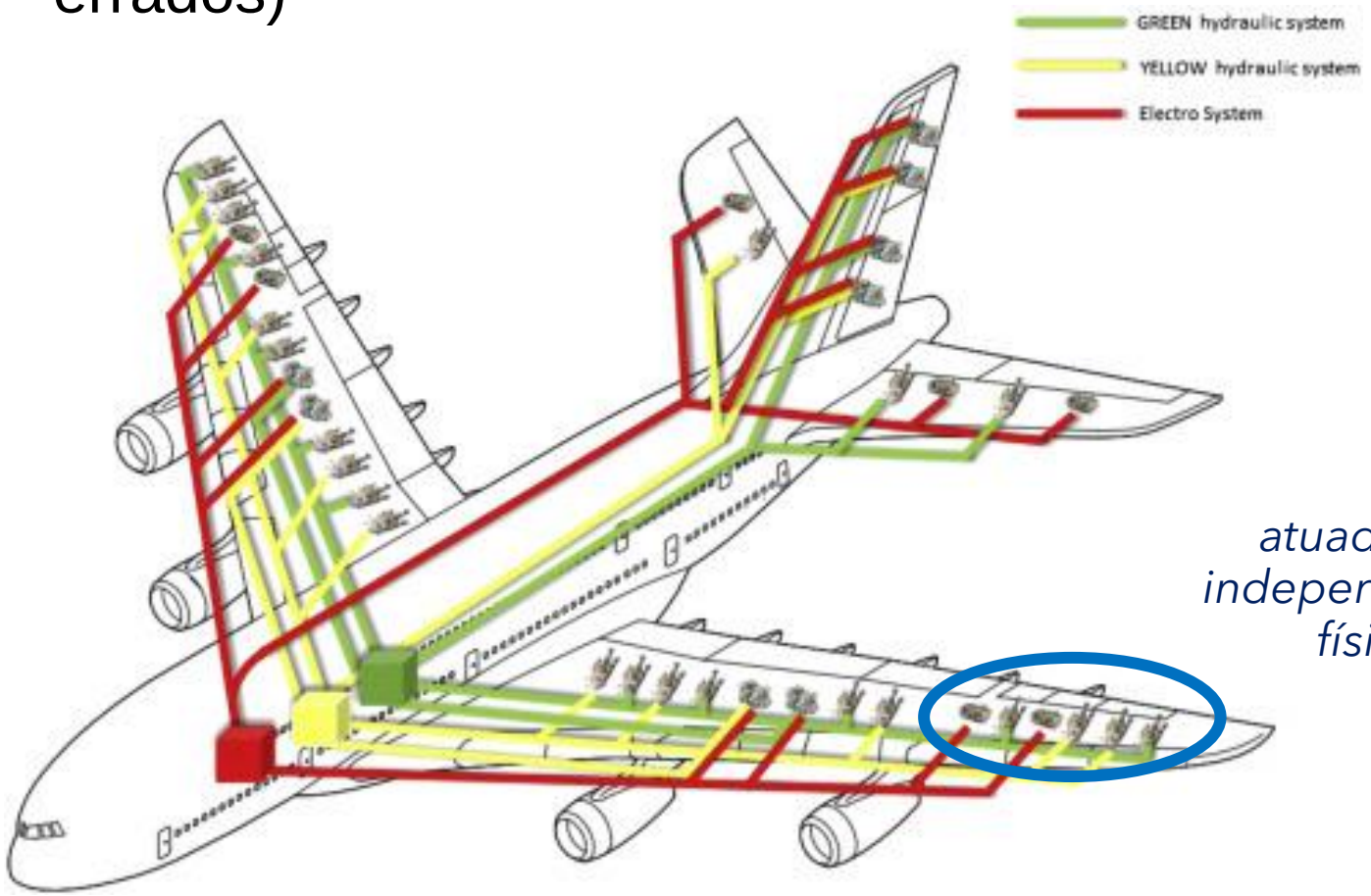
Mascaramento de Falhas

A ocorrência de falhas pode ser mascarada utilizando técnicas de redundância

- Redundância de informação
  - Ex.: O uso de bits extras para permitir recuperação da informação (Código de *Hamming*)
- Redundância de tempo
  - Ex.: Se necessário, uma ação realizada previamente pode ser repetida. Comum para tentativa de execução de transações abortadas
- Redundância Física
  - Uso de técnicas de replicação (Ex. RAID, servidores replicados)

Redundância Física

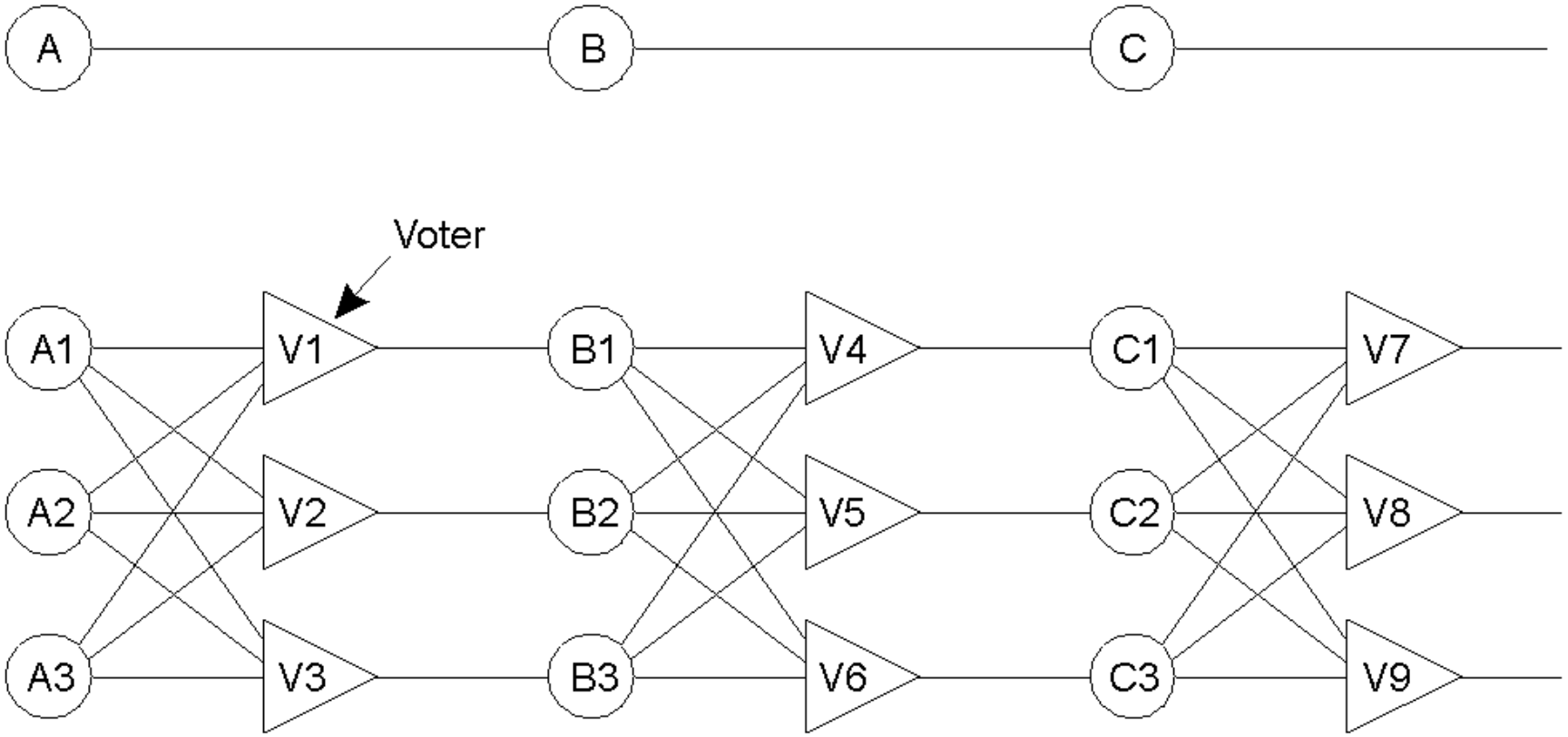
Sistemas replicados são adotados para garantir o comportamento correto na maioria das réplicas, sendo adotado o valor de saída observado pela maioria das réplicas (mascarando valores de saída errados)



Redundância de sensores, atuadores com implementações independents e com propriedades físicas diferentes (controlador hidráulico, elétrico, etc.)

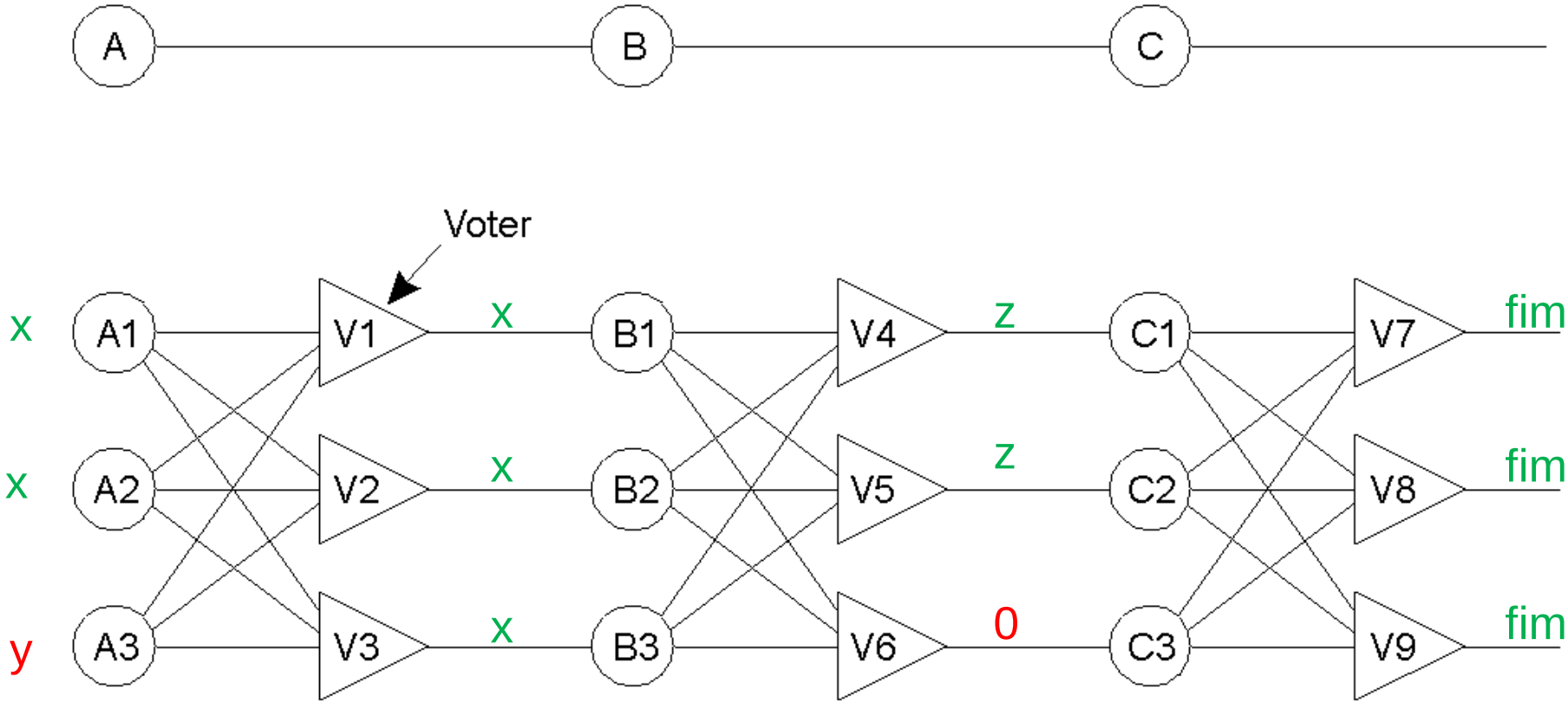
TMR – Redundância Tripla Modular

Para garantir que as saídas de A, B e C estão corretas, estes módulos são triplicados e um votador é decide o valor da saída com base na maioria dos valores iguais



TMR – Redundância Tripla Modular

Para garantir que as saídas de A, B e C estão corretas, estes módulos são triplicados e um votador é decide o valor da saída com base na maioria dos valores iguais



Observe que o votador também é um componente do sistema sujeito à falhas. Por isso ele também é replicado

Diversidade de Projeto

A replicação de um componente defeituoso (software ou hardware) também replica as suas falhas

- Para reduzir as chances de reproduzir uma mesma falha em diferentes réplicas, pode-se utilizar técnicas de diversidade de projeto
  - Componente é desenvolvido de maneira diferente
    - Por equipes diferentes
    - Usando linguagens/plataformas diferentes
    - Em locais diferentes

Mascaramento de Falhas e Replicação

- Ao usar *n* réplicas, a falha de *f* processos pode ser mascarada, desde que o número de processos corretos suficiente para atingir o consenso sobre o valor correto

Quantos processos são necessários no TMR para suportar :

f = 1 réplicas faltosas?

f = 2 réplicas faltosas?

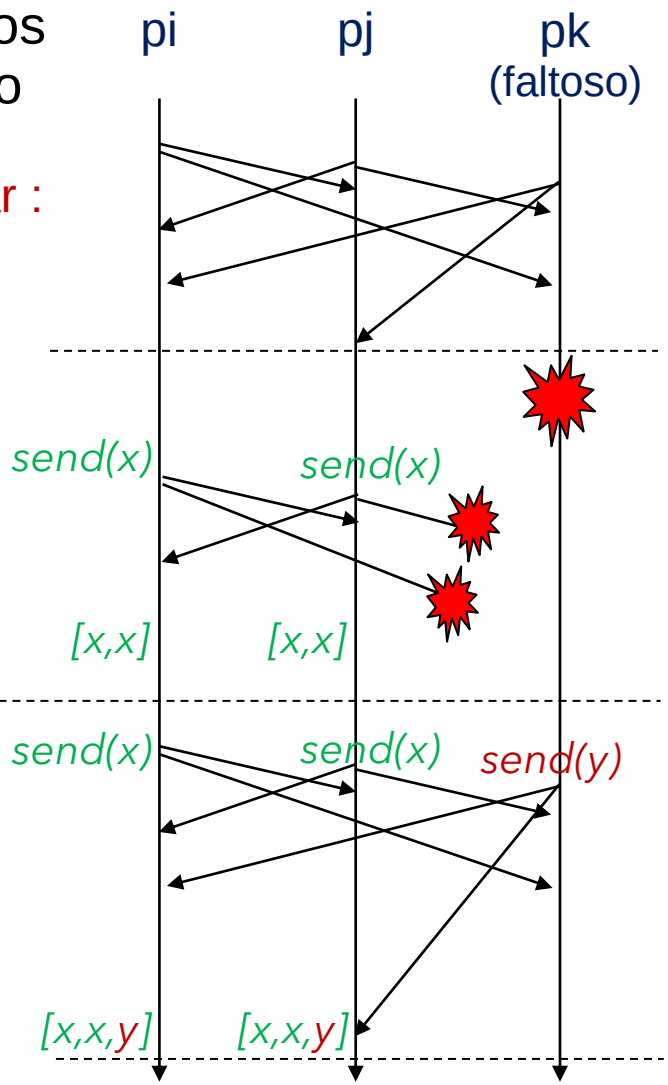
Falhas silenciosas: f processos param sem propagar informações erradas (ex. falha por colapso, parada)

`n = f + 1`

(n réplicas para tolerar f réplicas faltosas)

Caso os f processos apresentem comportamento arbitrário (ex. falhas bizantinas)

`n = 2f + 1`



Tratamento de Falhas

- Diagnóstico
  - Identifica a causa dos erros (localização, tempo e/ou tipo)
- Isolamento
  - Alteração física ou lógica no sistema, de forma que o componente falho não participe em ações futuras. O componente defeituoso não propagará falhas para o sistema
- Recuperação
  - Restaura o componente defeituoso e permite que este seja reinicializado de forma segura (*safety*)

Leituras recomendadas

L. Lamport, R. Shostak, M. Pease.
The Byzantine Generals Problem
Journal of ACM TOPLAS, 1982

F. Schneider.
What Good are Models and What Models are Good?
Distributed Systems, 1993

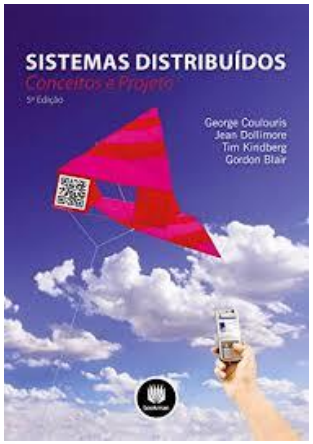
Schneider, F. B.
Implementing fault-tolerant services using the state machine approach: A tutorial
ACM Computing Surveys (CSUR), 1990

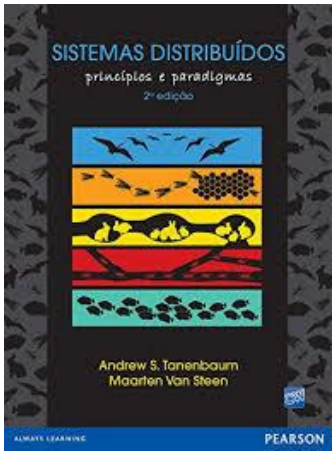
Castro, M., Liskov, B.
Practical byzantine fault tolerance
OSDI, 1999

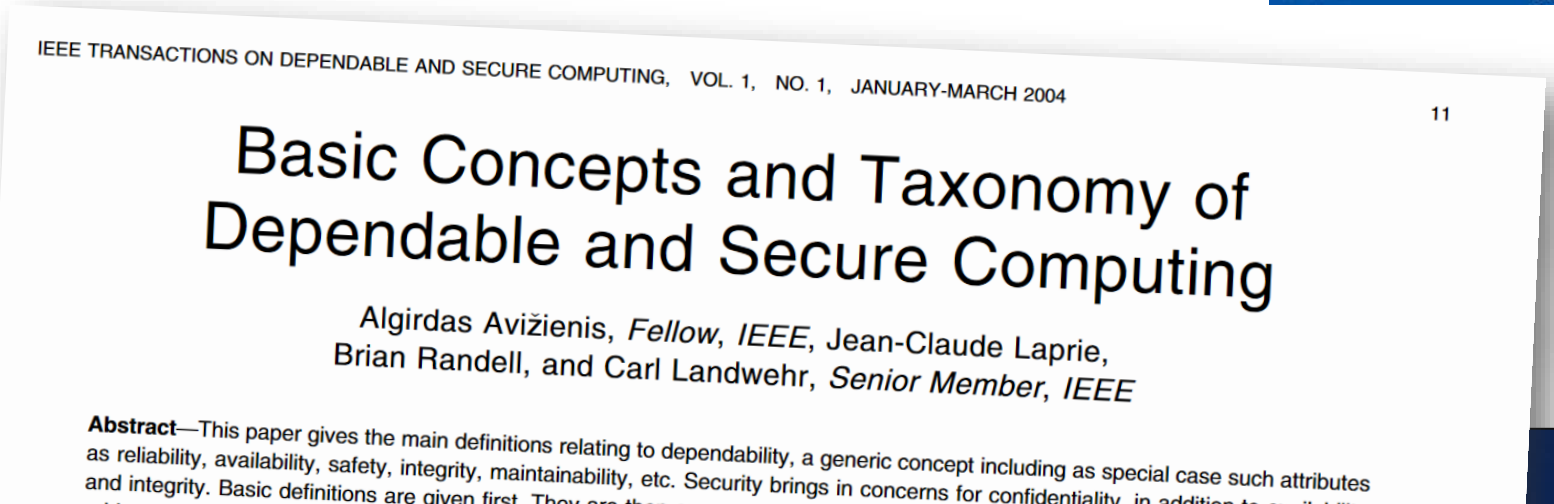
Referências

Parte destes slides são baseadas em material de aula dos livros:

- *Coulouris, George; Dollimore, Jean; Kindberg, Tim; Blair, Gordon. Sistemas Distribuídos: Conceitos e Projetos. Bookman; 5ª edição. 2013.*
- *Tanenbaum, Andrew S.; Van Steen, Maarten. Sistemas Distribuídos: Princípios e Paradigmas. 2007. Pearson Universidades; 2ª edição.*






IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO. 1, JANUARY-MARCH 2004 11

Basic Concepts and Taxonomy of Dependable and Secure Computing

Algirdas Avižienis, *Fellow, IEEE*, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, *Senior Member, IEEE*

Abstract—This paper gives the main definitions relating to dependability, a generic concept including as special case such attributes as reliability, availability, safety, integrity, maintainability, etc. Security brings in concerns for confidentiality, in addition to