Sistemas Tolerantes a Falhas

Ténicas de TF para Diversidade de Dados

Prof. Jó Ueyama

Introdução

- A diversidade de dados vem complementar as técnicas de diversidade vistas até agora
- A diversidade de dados involve
  - Obter um conjunto de pontos co-relatos
  - Executar o mesmo software neles
  - Utilizar os mecanismos de DM
- Os dados são obtidos através do DRAs
- Duas técnicas chaves
  - Retry Blocks (RtB)
  - N-copy Programming (NCP)

Retry Blocks (RtB)

- É uma das técnicas clássicas de diversidade de dados
- É uma técnica dinâmica
- Serve como um complemento da diversidade de dados para o RcB
- Ela é baseada em duas técnicas:
  - *Backward recovery*
  - AT
- Um WDT é também utilizado para dar o *start* de um algoritomo secundário
  - Caso o original não responda em tempo esperado

Retry Block

- O algoritmo de backup é executado com o dado original, inicialmente
- Se não passar no AT, então gera outro dado através do DRA
- Este novo dado é então executado com o algoritmo original
- Este procedimento acontece até que AT encontre um resultado correto
  - ou o deadline do WDT expire; neste caso o algoritmo de backup é invocado
  - e executa com o dado original (algoritmo de backup)

Funcionamento do RtB

- O RtB consiste de um controlador, AT, DRA, WDT, algoritmo primário e de backup
- O controlador orquestra a execução do RtB que possui a seguinte sintaxe

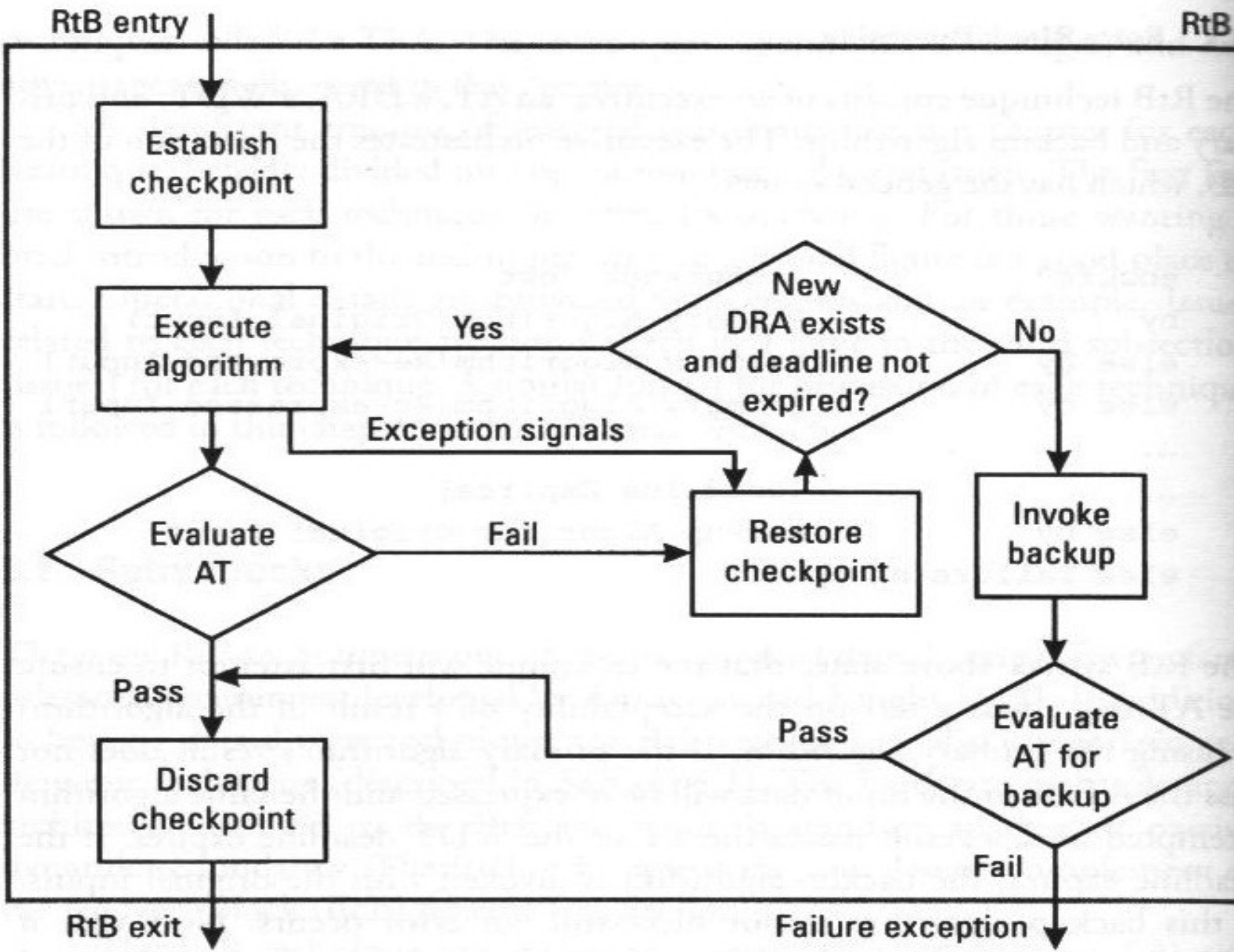
```
ensure              Acceptance Test
by                  Primary Algorithm(Original Input)
else by             Primary Algorithm(Re-expressed Input)
else by             Primary Algorithm(Re-expressed Input)
...
...                 [Deadline Expires]
else by             Backup Algorithm(Original Input)
else failure exception
```

Funcionamento do RtB

- Duas opções quanto ao DRA:
  - Múltiplos DRAs
  - Um único DRA com uma entrada randômica que gera dados a partir disso
- Passou do deadline? Então o algoritmo de backup com o dado original é executado

```
ensure                  Acceptance Test
by                      Primary Algorithm(Original Input)
else by                 Primary Algorithm(Re-expressed Input)
else by                 Primary Algorithm(Re-expressed Input)
...
...                     [Deadline Expires]
else by                 Backup Algorithm(Original Input)
else failure exception
```

Funcionamento do RtB



Cenários do Funcionamento do RtB

- Funcionamento livre de falhas
- Exceção levantada pelo algoritmo primário
- Execução do primário em tempo, mas falha no AT; sucesso na execução do input do DRA (re-expr)
- Todas as re-expressões realizadas sem sucesso; sucesso na execução do algoritmo de backup
- Todas as re-expressões realizadas sem sucesso; backup é executado, mas o AT falha
  - Este cenário será explicado com detalhes no próximo slide

Cenário de Falhas com o RtB

- O controlador guarda todos os checkpoints, invoca o P (algoritmo primário) e estabele o WP
  - WP é o tempo máximo de espera
- O resultado de P é submetido ao AT que falha
- Os dados do checkpoint são restaurados
- Existe um DRA? Se sim, ele é executado tendo como o argumento o dado inicial
- Os dados são submetidos ao AT que falha
- E assim sucessivamente até que exista algum DRA disponível

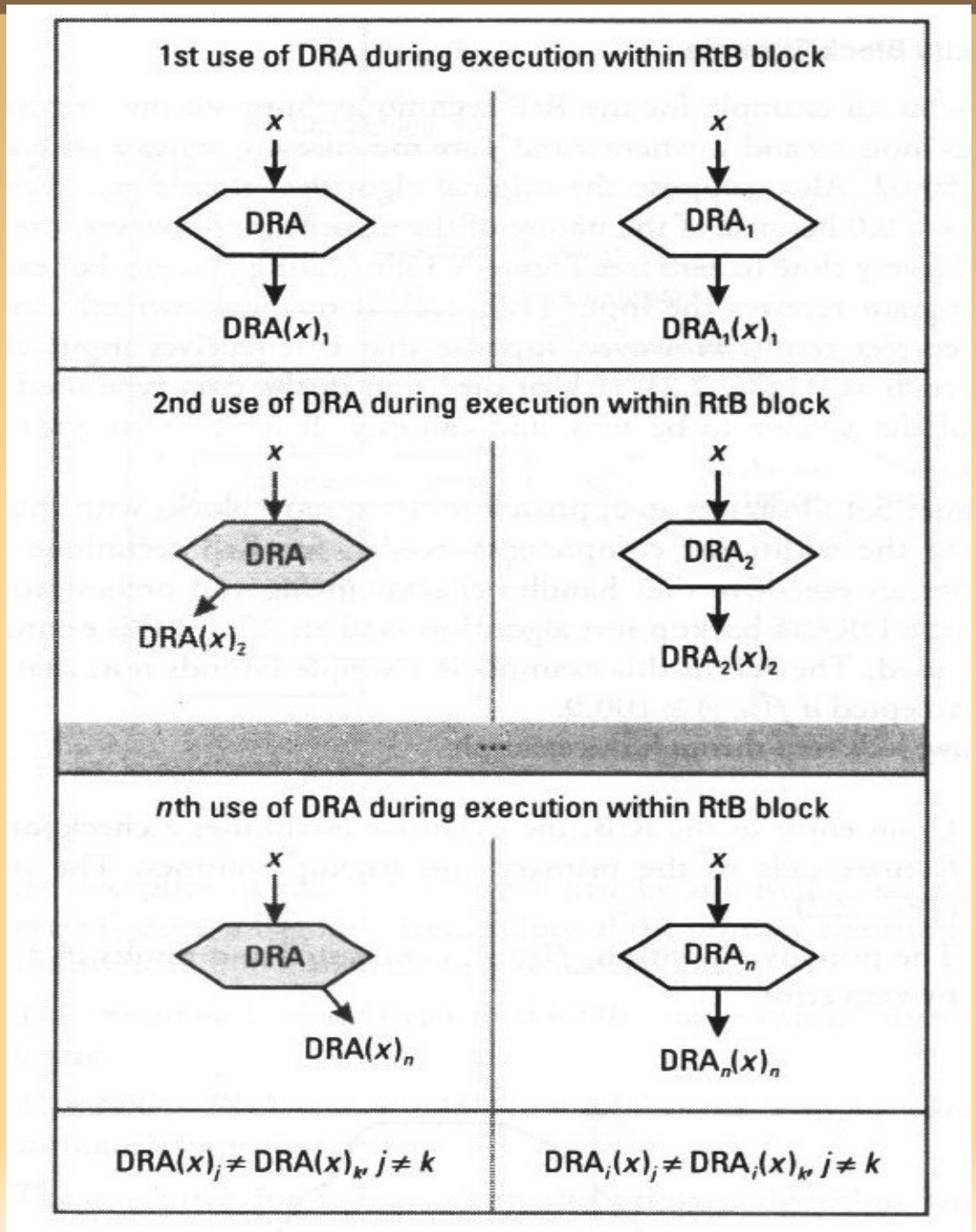
Cenário de Falhas com o RtB

- Se não existe nenhum DRA disponível, então o algoritmo de backup é invocado
- O algoritmo de backup utiliza o dado original e é executado
- O resultado do backup é avaliado pelo ATB (AT diferente) que falha
- Importante: em nenhum dos cenários acima, o processamento ultrapassou o deadline WP
- O checkpoint e o WDT (WP) são removidos
- Uma exceção é levantada

RtB Estendido

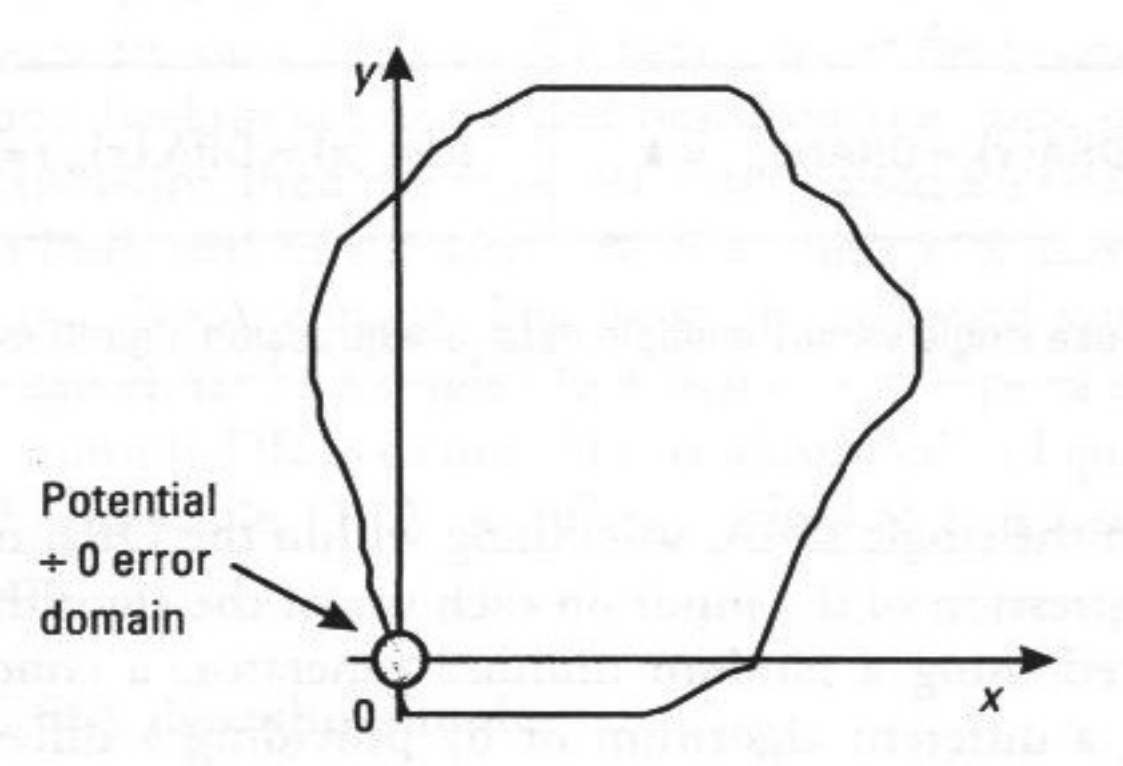
- Uma das extensões é inserir um contador do # de vezes que um primário pode executar com o DRA
- Vantagem? Ela pode substituir o WDT
- Duas opções para o DRA
  - Múltiplos DRAs
  - Um único DRA
- No caso de um único DRA, pode haver:
  - um valor randômico que é utilizado para gerar dados diferentes
  - um parâmetro diferente além do próprio input x
  - uma implementação de switch no próprio algoritmo

Multiuse Single vs. Multiple DRAs

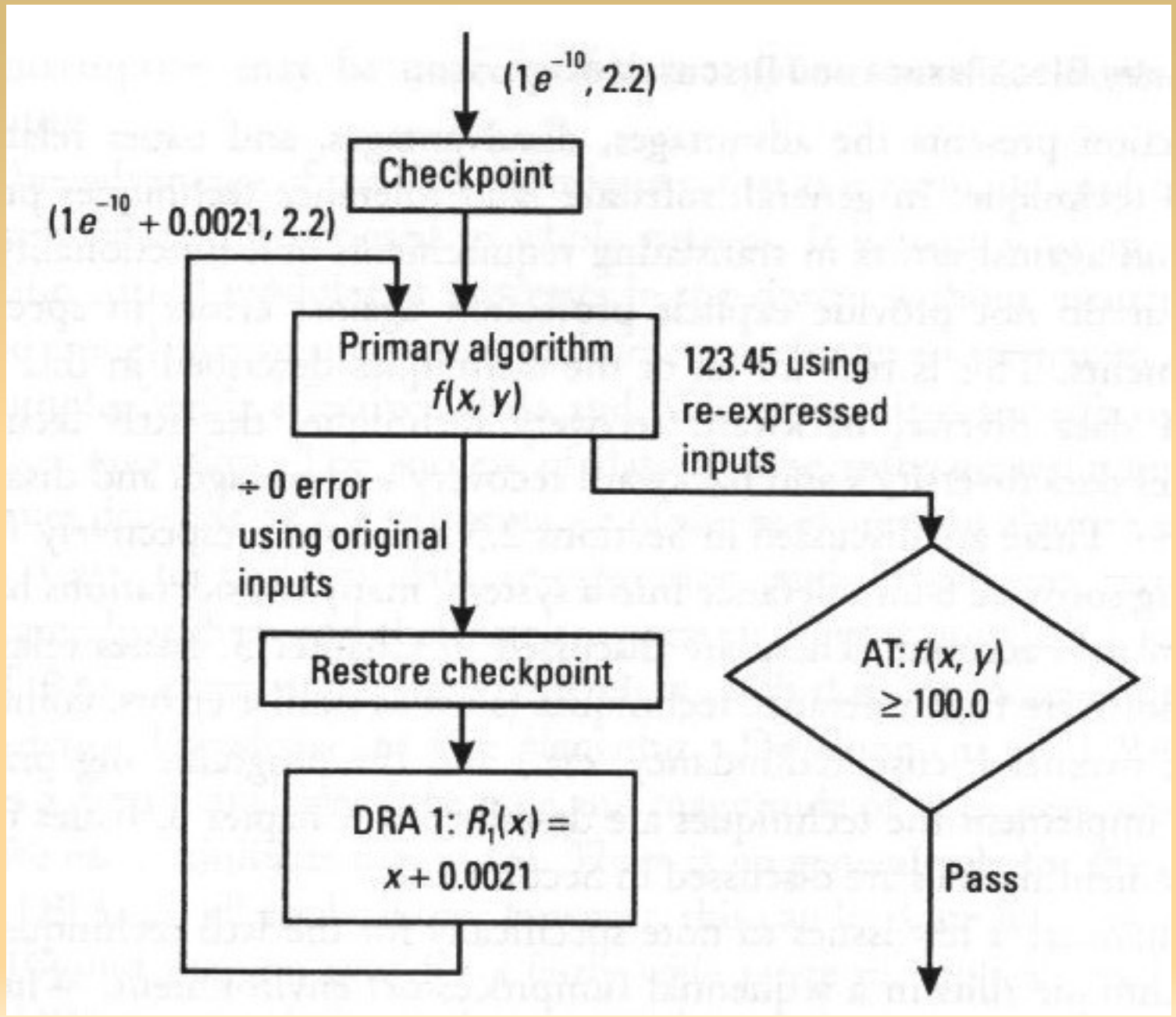


Exemplo de Retry Block

- Um programa usa um input x e y que é coletado dos sensores com uma tolerância de +/- 0.02
- Os valores de x e y não podem ser x=0 e y=0 porque a aplicação possui uma divisão por zero
- O espaço do input é ilustrado abaixo



Exemplo da Implementação do RtB



Discussão do RtB

- É normalmente executado em ambientes monoprocessados
- Inclui overhead do *backward recovery*
- A execução dos DRAs leva a interromper os serviços, o que pode ser inaceitável
  - Por que pode levar a 'parada' do serviço?
- DRA são dependentes da aplicação e normalmente os mais simples são melhores
  - Por que? Porque minimizam erros de projeto e implementação

Discussão do RtB

- As aplicações com sensores podem ser bastante voltados para a diversidade de dados
  - Por que? Porque podem prover pequenas modificações sem alterar a aplicação
  - Os sensores normalmente entregam dados imprecisos e muitas vezes acompanhados de ruídos
- Tanto o RtB como o RcB podem levar ao efeito Dominó
  - efeito cascata nos rollbacks de todos os processos envolvidos 'puxando-os' para o início da execução
- A TF depende do DRA gerar dados fora da região de falhas

Concluindo RtB

- Concluímos o capítulo 4 que trata da diversidade de software
- Finalizamos a parte do Retry Blocks
- Próxima aula abordaremos o N-copy Programming

Sistemas Tolerantes a Falhas

N-copy Programming

Prof. Jó Ueyama

N-copy Programming (NCP)

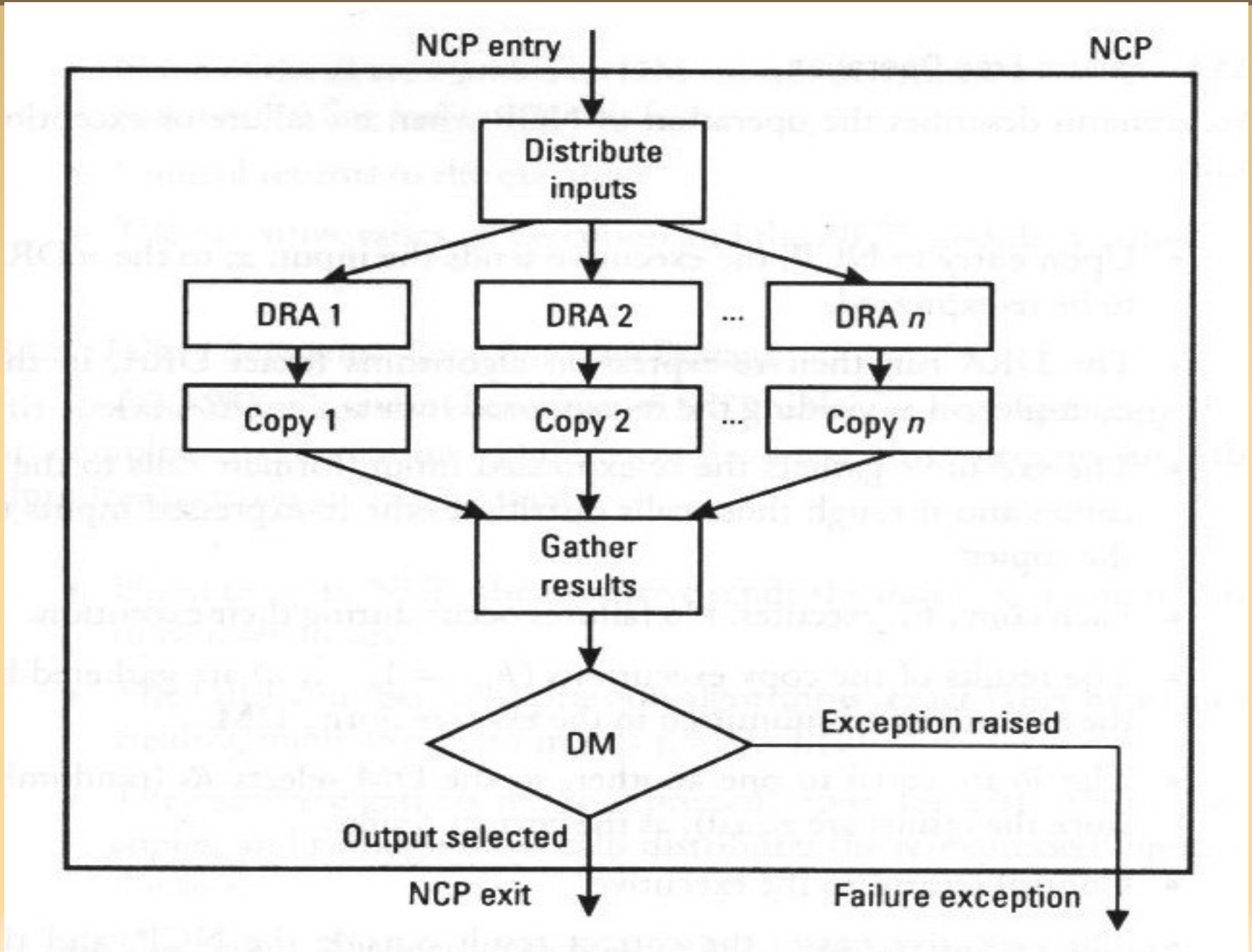
- Juntamente com o RtB, ele é uma outra técnica clássica
- É uma técnica
  - Estática
  - Paralelo ou concorrente
- Serve como uma TF de diversidade de dados para o N-version (i.e. complemento do NVP)
- NCP é baseado no DM e no *forward recovery*
- Utiliza pelo menos duas variantes do mesmo programa
- Os DRAs alimentam as variantes antes

Sintaxe Geral do NCP

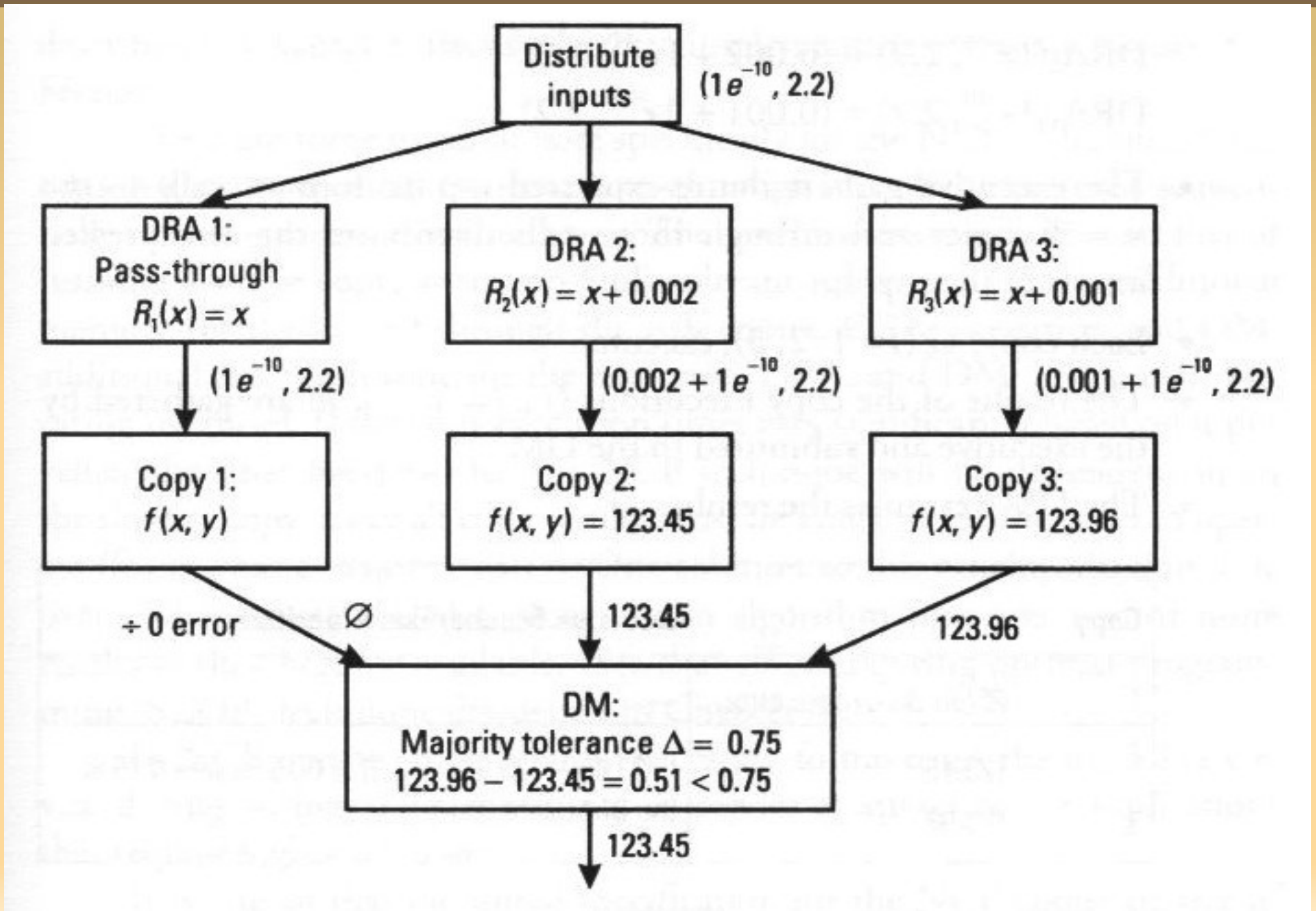
- O NCP consiste de um controlador, 1 a *n* DRAs, *n* variantes do programa e uma DM
- A DM seleciona o 'melhor' resultado caso ele exista
- Ele segue a seguinte sintaxe

```
run DRA 1, DRA 2, ..., DRA n
run Copy 1(result of DRA 1),
    Copy 2(result of DRA 2), ...,
    Copy n(result of DRA n)
if (Decision Mechanism (Result 1, Result 2, ...,
                        Result n))
    return Result
else failure exception
```

Estrutura e Operação do NCP



Exemplo com o NCP



Discussão sobre o NCP

- No exemplo anterior, o DM seleciona um dos resultados pois a diferença entre eles é < 0.75
  - 0.75 é um threshold do exemplo
- A performance depende da variante mais lenta
- Por isso, o DM já pode verificar a diferença com os dois primeiros resultados.
- Os DRAs são bem dependentes de cada aplicação, por isso eles não são genéricos
- Porém, existem DRAs próprios para uma variedade de aplicações (e.g. para sensores)

Concluindo NCP

- Concluímos o NCP
- Visitamos duas técnicas clássicas de diversidade de dados
  - Retry Blocks (dinâmico)
  - N-copy Progamming (estático)
- Fim do capítulo 5

Sistemas Tolerantes a Falhas

Outras Técnicas de TF (Checksum, Paridade e RAID)

Prof. Jó Ueyama

Checksum

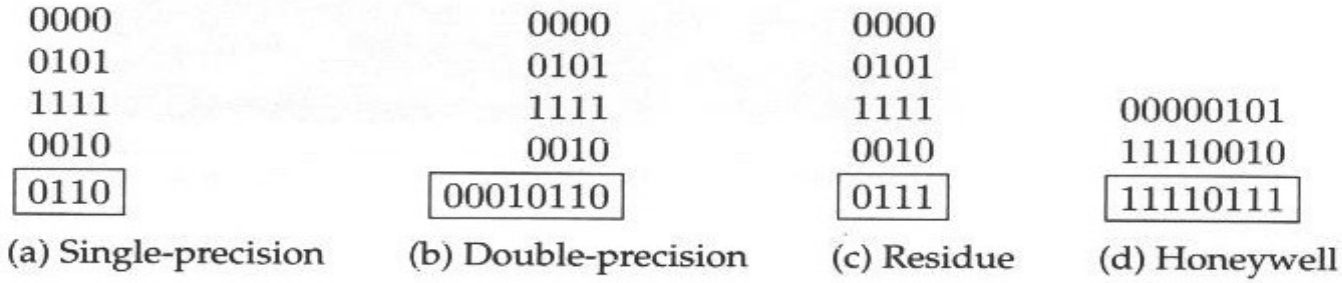


FIGURE 3.6 Variations of checksum coding (boxed quantities are the computed checksums).

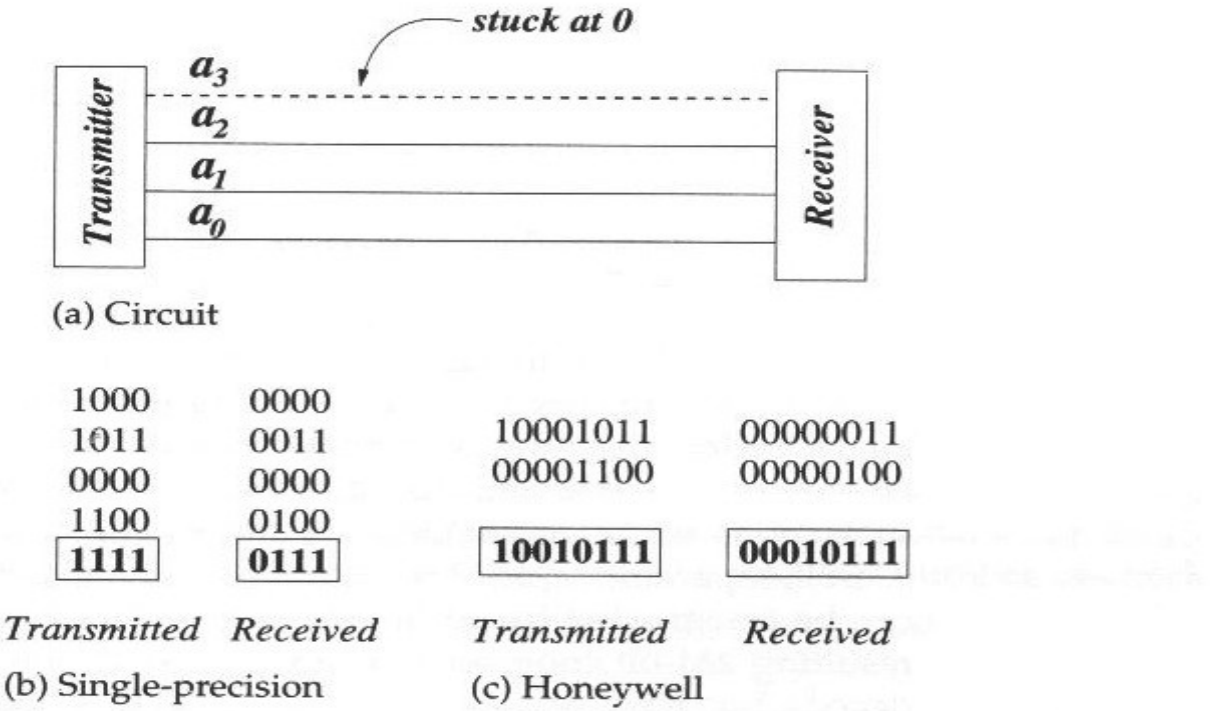


FIGURE 3.7 Honeywell versus single-precision checksum (boxed quantities indicate transmitted/received checksum).

Verificação de Paridade

- **Paridade com bit único:** detecta erro de um único bit.
- **Esquema de paridade par:** d bits de dados + 1 de paridade devem conter número par de bits 1.

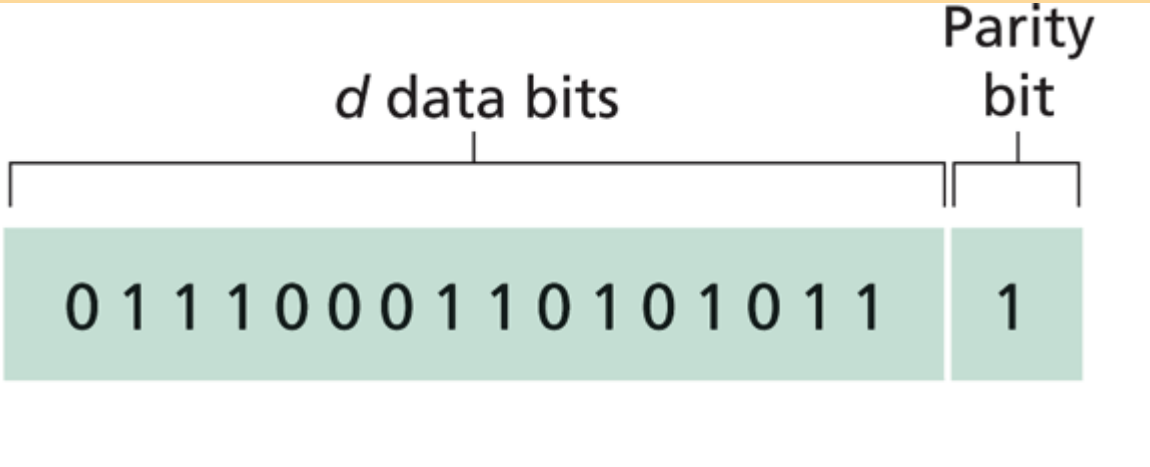


Figure 5.5 ♦ One-bit even parity

Verificação de Paridade

- **Paridade Bidimensional**:
  - permite identificar e corrigir **um** bit errado!
  - também detecta qualquer combinação de dois erros.
  - conhecida como FEC (Forward Error Correction).

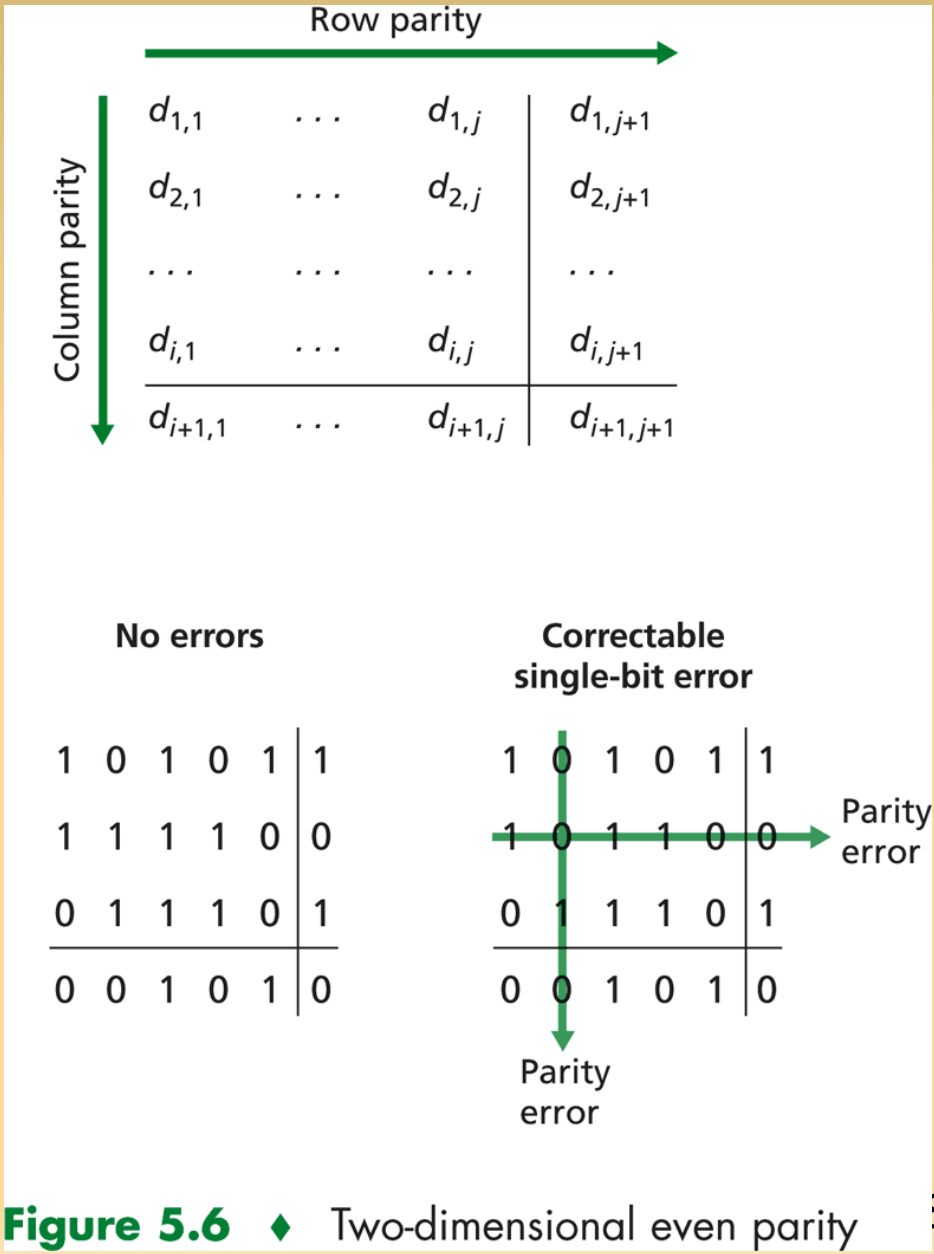


Figure 5.6 ♦ Two-dimensional even parity

RAID

- RAID (*Redundant Array of Independent Disks*)
  - Armazena grandes quantidades de dados;
- RAID combina diversos discos rígidos em uma estrutura lógica:
  - Aumentar a confiabilidade, capacidade e o desempenho dos discos;
  - Recuperação de dados
  - Redundância dos dados;
  - Armazenamento simultâneo em vários discos permite que os dados fiquem protegidos contra falha (não simultânea) dos discos;
  - Performance de acesso, já que a leitura da informação é simultânea nos vários dispositivos;

RAID por Hardware

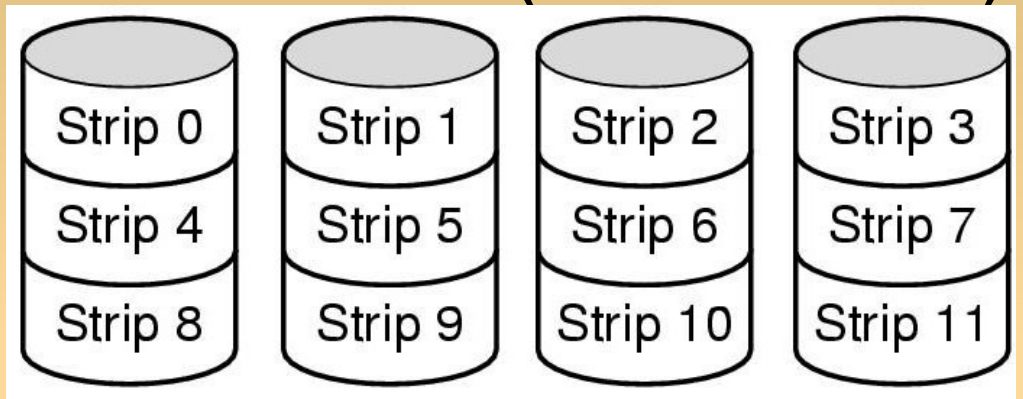
- Pode ser implementado por:
  - Hardware (controladora):
    - Instalação de uma placa RAID no servidor
    - O subsistema RAID é implementado totalmente em hardware;
    - Funciona como se fosse um co-processador RAID
    - Libera o processador para se dedicar exclusivamente a outras tarefas;
    - A segurança dos dados aumenta no caso de problemas devido à checagem da informação na placa RAID antes da gravação;

RAID por Software

- Pode ser implementado por:
  - Software (sistema operacional)
    - Menor desempenho no acesso ao disco;
    - Oferece um menor custo e flexibilidade;
    - Sobrecarrega o processador com leitura/escrita nos discos;
- Para o SO existe um único disco;

Níveis de RAID

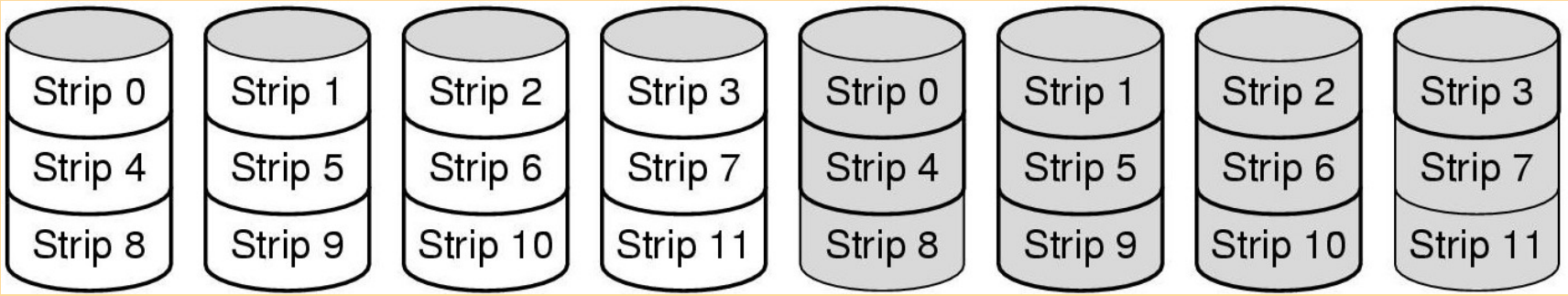
- A forma pela qual os dados são escritos e acessados define os níveis de RAID (até 9 níveis):
- RAID 0:
  - Melhora o desempenho
  - Se o software manda ler um bloco de 4 tiras consecutivas, iniciando em um limite de tira, o controlador do RAID quebrará esse comando em 4 – um para cada disco



  - Paralelismo em I/O em discos separados;
  - Utilizam mesma controladora (controladora RAID);
  - Aplicações multimídia (alta taxa de transferência);
  - Funciona bem para altas taxas de transferências de dados em virtude do paralelismo

Níveis de RAID

- RAID 1:
  - Conhecido como espelhamento (mirroring);
  - Duplica todos os discos
    - 4 principais e 4 de reserva
    - Operações de escrita no disco primário são replicadas em um disco secundário;
    - Leitura pode ser feita de qualquer cópia → distribui a carga
  - Pode ter controladoras diferentes;



Níveis de RAID

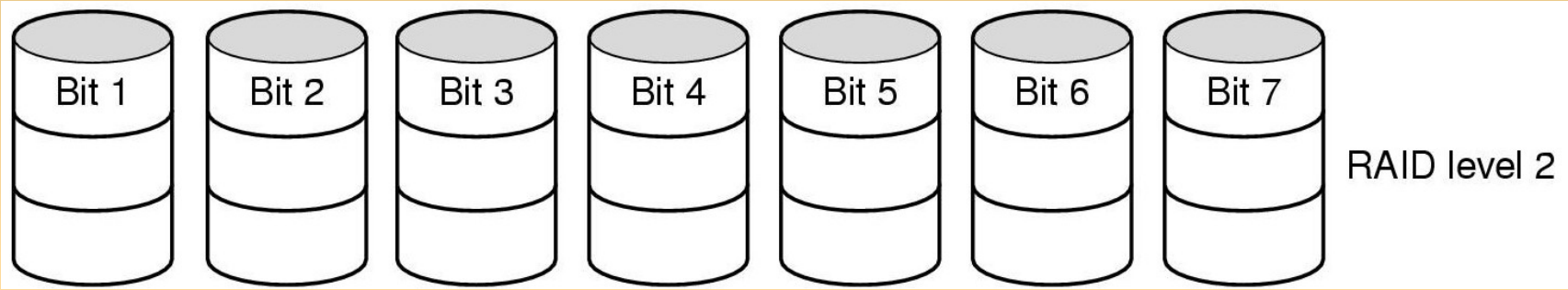
- RAID 1:
  - Excelente tolerância a falhas
  - Se um drive falhar, a cópia é usada
  - Recuperação consiste em instalar um novo disco e copiar do backup para ele
  - Desvantagem: espaço físico em dobro (alto custo);
  - Transações on-line (tolerância a falhas);
- RAID 10:
  - Combinação dos RAID 1 e RAID 0;

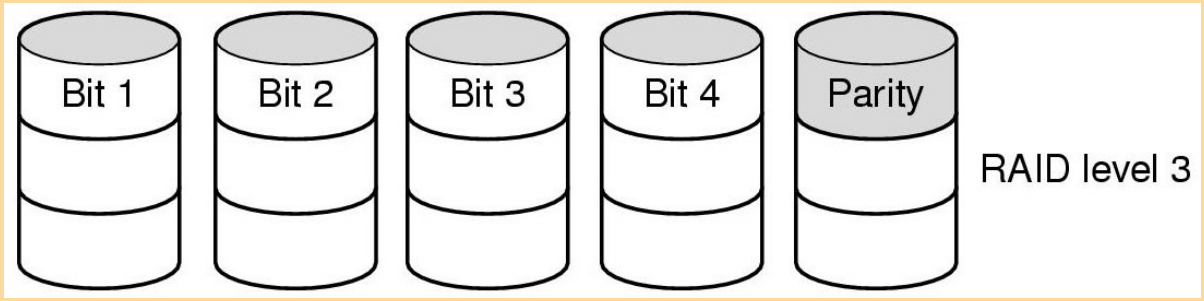
RAID 2, 3 e 4

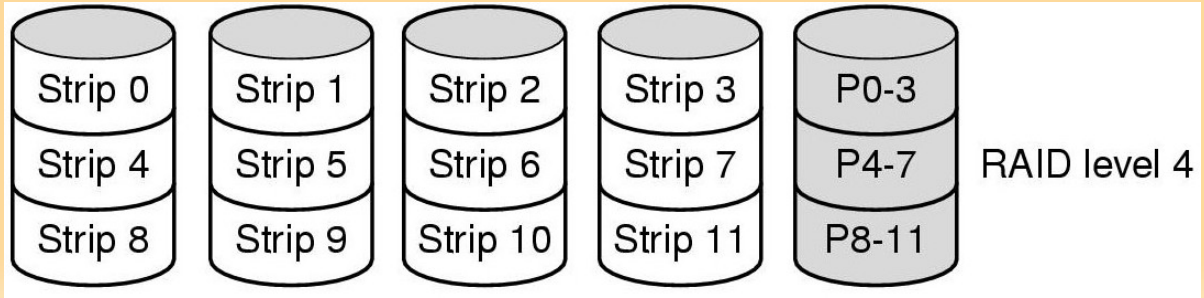
- Trabalham com bytes, bits ou stripes
  - Se um disco 'quebra' apenas alguns bytes são perdidos
- Dados são armazenados em discos diferentes
  - Com bit de paridade (permite reconstruir dados perdidos)
  - Paridade é mantida em um disco apenas;
- Diferença básica: como a paridade é calculada (na transferência):
  - RAID 2 - Hamming ECC (error-correcting codes)– nível de bit;
  - RAID 3 - XOR ECC - nível de byte ou bit (um disco de paridade);
    - Um bit de paridade para cada palavra
  - RAID 4 – XOR ECC - nível de stripe (um bit de paridade para cada stripe)

Niveis de RAID

- RAID 2, 3 e 4

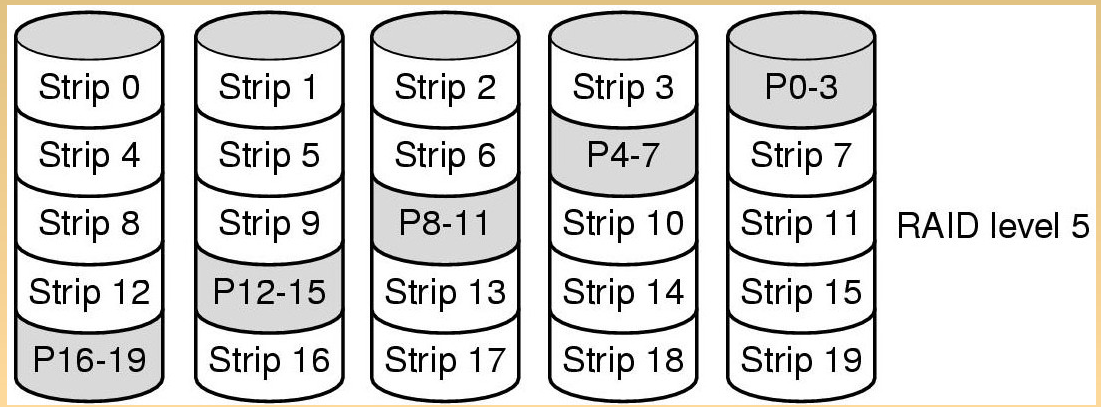






RAID 4 e 5

- RAID 5:
  - Stripes;
  - Paridade XOR ECC distribuída - nível de bloco;
  - Paridade está distribuída nos discos;
- RAID 6:
  - Stripes;
  - RAID 5 com dois discos de paridade;



Concluindo...

- Concluímos a apresentação de outras técnicas de TF
  - Checksum
  - Bit de paridade
  - RAID e os diversos tipos
- Conteúdo retirado de diversos materiais
  - Tanembaum (SO)
  - Kurose (Redes de Computadores)