Sistemas Tolerantes a Falhas

Programação *N Self Cheking* (NSCP), *Consensus Recovery Block* (CRB) e Callback

Prof. Jó Ueyama

N Self Checking

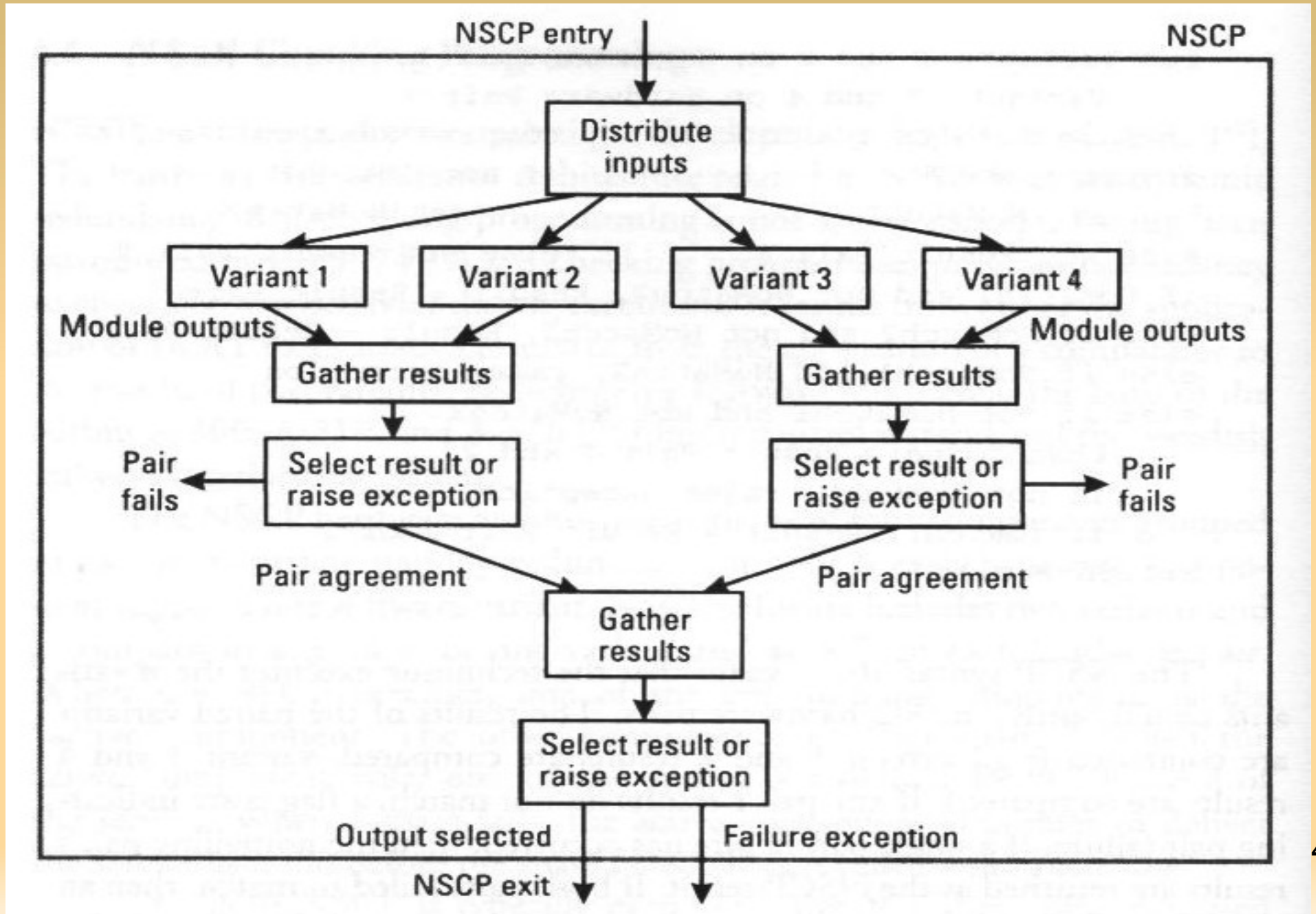
- É um tipo de implementação para a diversidade
- Em ambientes multiprocessados
- Os resultados podem ser avaliados por um:
  - Um AT para cada variante
  - Comparador entre os resultados das variantes
- Uma implementação típica seria
  - Quatro variantes cada uma delas em um hardware
  - Um comparador para cada par de resultados
- Dois tipos de variantes
  - *Active*
  - *Hot spares*

N Self Checking

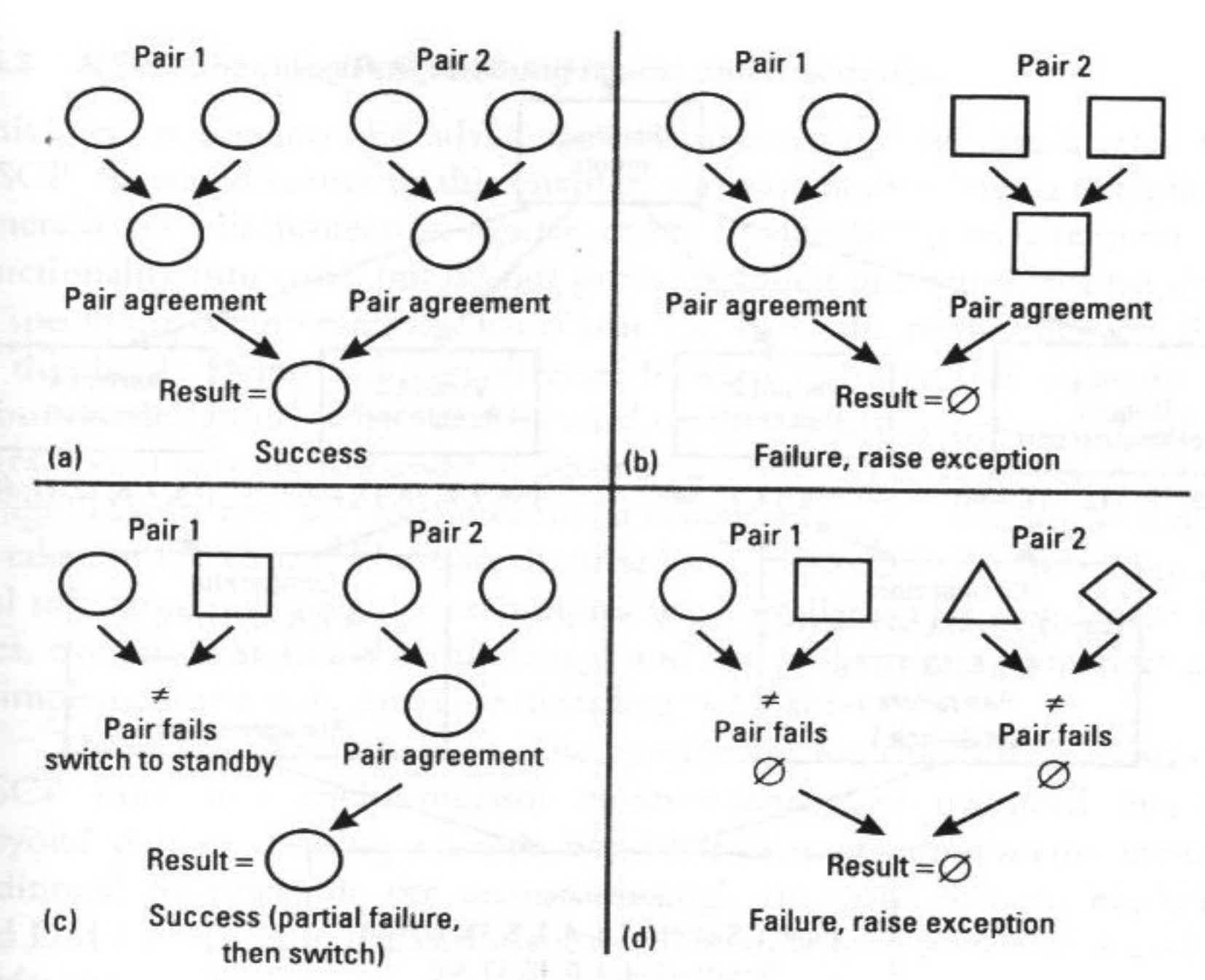
- Pares executados concorrentemente (estático)
- Um exemplo para *n* = 4

```
run Variants 1 and 2 on Hardware Pair 1,
    Variants 3 and 4 on Hardware Pair 2
compare Results 1 and 2          compare Results 3 and 4
if not (match)                   if not (match)
    set NoMatch1                     set NoMatch2
else set Result Pair 1           else set Result Pair 2
if NoMatch1 and not NoMatch2, Result = Result Pair 2
else if NoMatch2 and not NoMatch1, Result = Result Pair 1
else if NoMatch1 and NoMatch2, raise exception
else if not NoMatch1 and not NoMatch2
    then compare Result Pair 1 and 2
    if not (match), raise exception
    if (match), Result = Result Pair 1 or 2
return Result
```

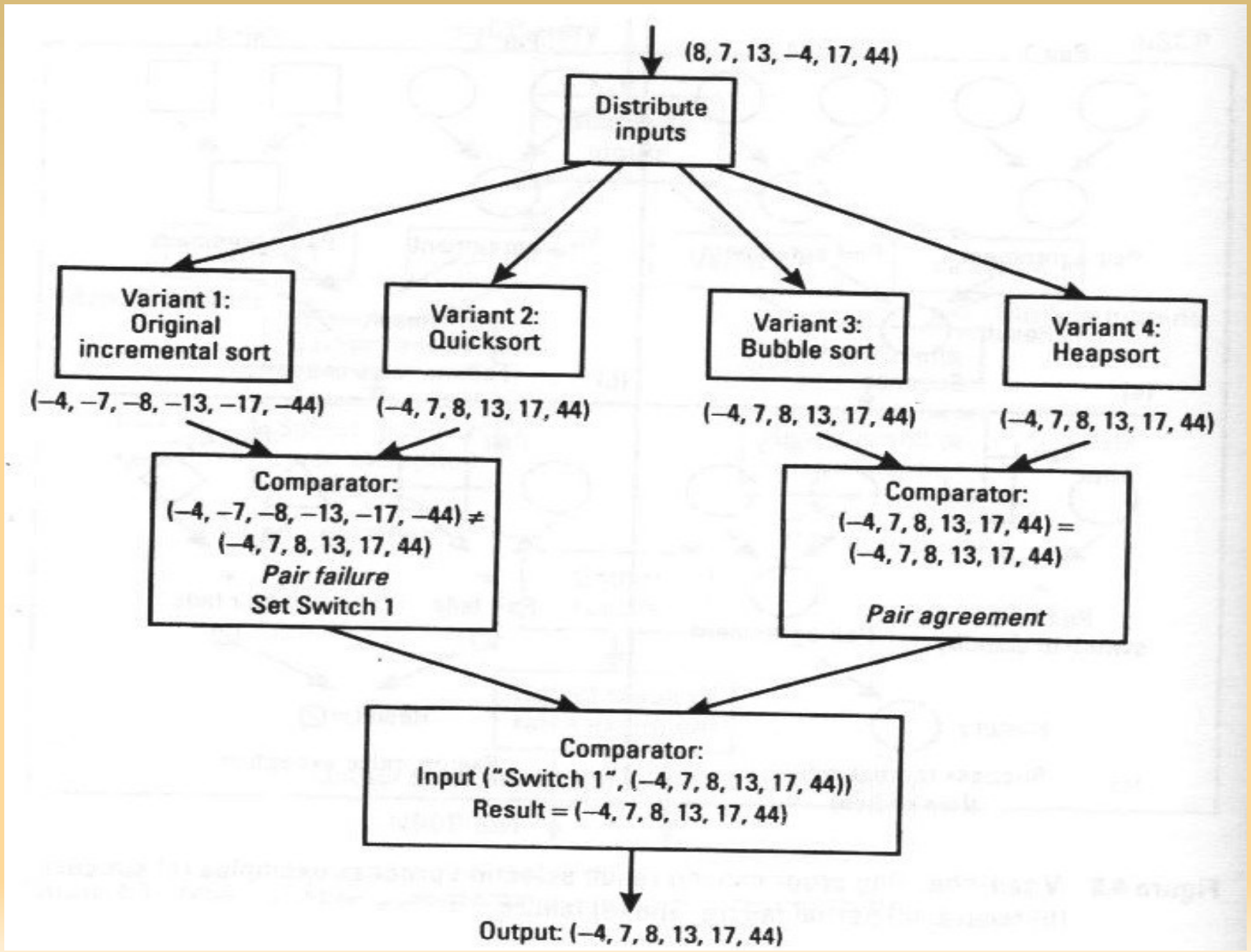
NSCP-Estrutura e Operação



N Self Checking



N Self Checking - Exemplo



N Self Checking - Discussão

- Um tipo de diversidade e implementa o *forward recovery*
- E o overhead?
- Memória e processamento extra para os
  - 'primário' *active*
  - E as *n* variantes
  - DM (*voter* e um AT)
  - Controlador
- Sem falar do tempo de sincronização. Por que?
- *Result switching:* troca de resultados quando o 'primário' falha

N Self Checking - Discussão

- E a diferença com o NVP?
- No NVP, a cooperação é para prover resultados
  - Através do voto pela melhor decisão pelo DM
- O NSCP vai além disso e preocupa para dar um resultado aceitável
- Técnicas usadas com o NSCP
  - Assertions
  - Ações atómicas
  - Componentes idealizadas

N Self Checking - Discussão

- E os tamanhos dos módulos?
- Pequenos? Grandes?
- Vantagens e desvantagens associados
- Onde aplicar o NSCP? E em que nível?
- O controlador pode permanecer em um hardware separado dependendo do custo
- Se o custo das 4 variantes for muito alto, NSCP permite criar apenas 3 variantes
  - As 3 permanecem em hardwares separados
  - E a última é uma réplica de uma das variantes

Consensus Recovery Block (CRB)

- É uma combinação do RcB com o NVP
- CRB reduz a importância do *voter* e pode lidar com o MCR (Multiple Correct Results)
  - MCR não é apropriado para o NVP, por que?
- Assim como no RcB, as variantes são 'rankeadas'
  - Em ordem de serviço e confiabilidade. Por que?
- As variantes são executadas em paralelo, como no NVP
- E os resultados são votados
  - Voto da maioria ou do consenso

Consensus Recovery Block (CRB)

- O mecanismo do voto não usa a escolha da maioria?
  - Então a variante tida como a primária tem o seu resultado submetido ao AT
  - Se falhar no AT, então o resultado da próxima variante do ranking é selecionada
  - Assim sucessivamente, até passar no teste
- Primeiro, a parte do NVP é executada e depois o mecanismo RcB entra em ação
  - Caso a parte do NVP não encontre um resultado

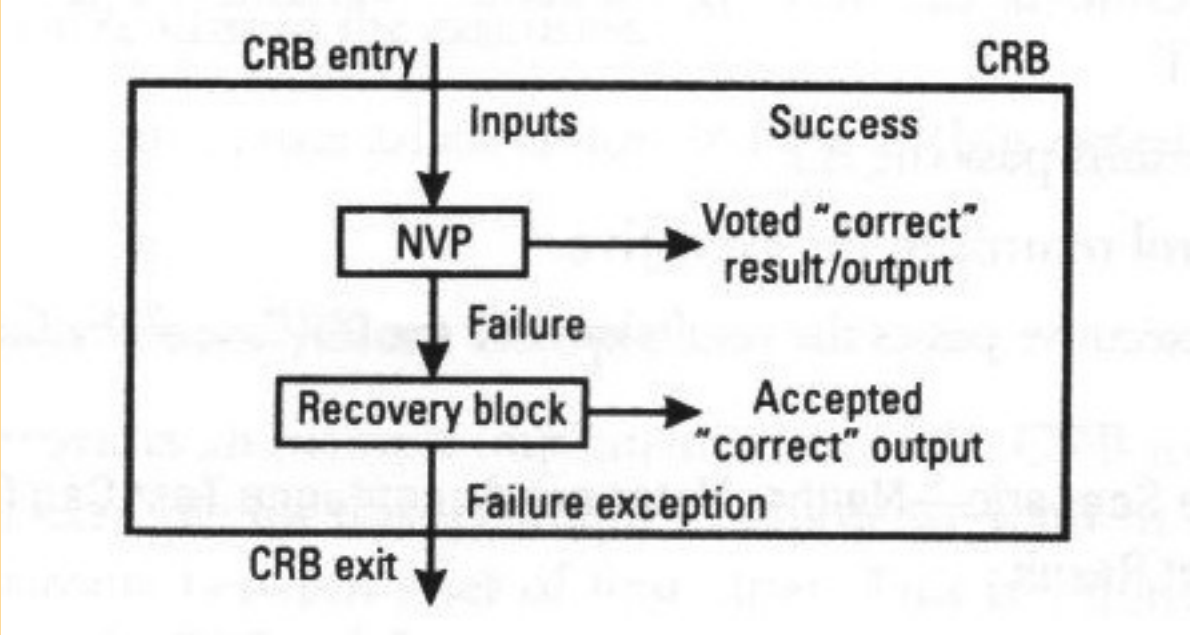
Consensus Recovery Block (CRB)

- O controlador do CRB orquestra a operação do CRB que possui a estrutura abaixo:
  - Primeiro a parte do NVP e depois a do RcB são executadas
  - O resultado pode ou não ser utilizado na parte do RcB

```
run Ranked Variant 1, Ranked Variant 2, ..., Ranked Variant n
if (Decision Mechanism (Result 1, Result 2, ..., Result n))
     return Result
else
     ensure                  Acceptance Test
     by                      Ranked Variant 1 [Result]
     else by                 Ranked Variant 2 [Result]
     ...
     else by                 Ranked Variant n [Result]
     else raise failure exception
return Result
```

Consensus Recovery Block (CRB)

- Estrutura e a operação do CRB
- *Failure-Free Operation*
- *Partial Failure Scenario (voter Fails, but AT does)*
- *Failure Scenario (voter and AT fail)*



Failure-Free Operation - CRB

- Os *inputs* passam pelo teste do formato e são submetidos às variantes
- Nenhum erro ocorre com a execução das variantes
- O controlador coleta cada resultado e os submete ao DM
- O DM então seleciona o resultado da maioria
  - Seleciona um único randomicamente, uma vez que todos possuem o mesmo valor
- O resultado selecionado é enviado ao módulo fora do CRB

Partial Failure Scenario - CRB

- Inicialmente, passa-se pelo teste do formato e distribuem-se os *inputs*
- Os resultados são coletados e submetidos ao DM
- *Voter* não pode determinar o resultado correto
- Controlador passa o resultado da variante 'primária' para o AT
- O AT falha;
- O controlador passa o resultado da variante No. 2 no ranking
- Finalmente, o AT acaba podendo selecionar este resultado

Failure Scenario - CRB

- O DM e o AT falham; considera-se *n* = 3
- Inicialmente realizam-se os testes de formato e depois enviam-se os *inputs* às variantes
- Após a execução das variantes, os seus resultados são coletados e submetidos à DM
- Os resultados diferem-se significantemente um do outro e a DM falha
- O controlador submete o resultado da variante 'primária' ao AT; o AT falha
- O controlador submete o resultado da variante No. 2; o AT falha

Failure Scenario - CRB

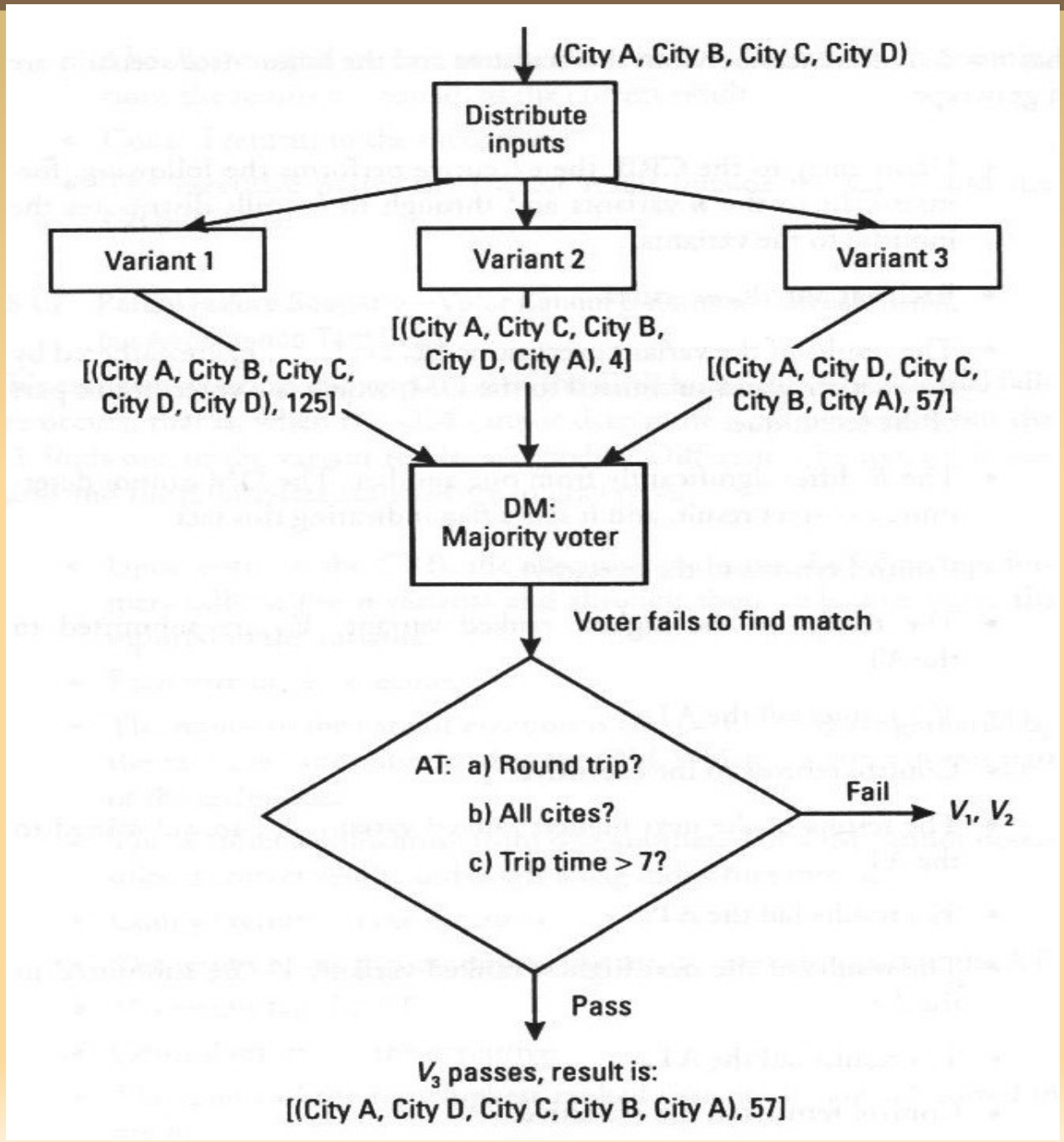
- O controlador submete o resultado da variante No. 3; o AT falha
- O controlador então levanta uma exceção
- Note que o CRB necessita que o controlador sincronize a chegada dos resultados
- O AT permite o MCR. Por que?

Exemplo com o CRB

- Cada variante provê uma rota (as cidades por onde passam) e o tempo gasto no percurso
- *Round trip* e o tempo (*round trip time*)

j	r_{1j}	r_{2j}	r_{3j}	**Result**
1	City A	City A	City A	
2	City B	City C	City D	
3	City C	City B	City C	Multiple correct or incorrect results
4	City D	City D	City B	
5	City D	City A	City A	
Time	125	4	57	

Exemplo com o CRB



CRB - Discussão

- Overhead extra:
  - Ambiente multiprocessado
  - Execução e sincronização das variantes
  - *Voting mechanism*
  - AT
- Técnica híbrida (NVP e RcB)
- Uma desvantagem típica da técnica híbrida: complexidade e maior probabilidade de erros
  - Projeto e implementação
- Note que o AT só é executado após o OK do *voter*

CRB - Discussão

- O CRB é bem parecido com o NVP (ambiente multiprocessado) exceto o AT
- O controlador pode residir em um hardware separado
- Tal hardware separado pode acomodar o AT tb

Interface Callback

- A interface ICallback é uma implementação para prover TF de forma similar aos preconditions e aos postconditions
- Os métodos das operações são utilizadas para assegurar que uma determinada operação pode ser realizada
- Utiliza dois métodos
  - veto() que retorna *true* ou *false* para permitir ou não uma operação
  - receive() recebe a operação e os parâmetros de uma chamada

Interface Callback

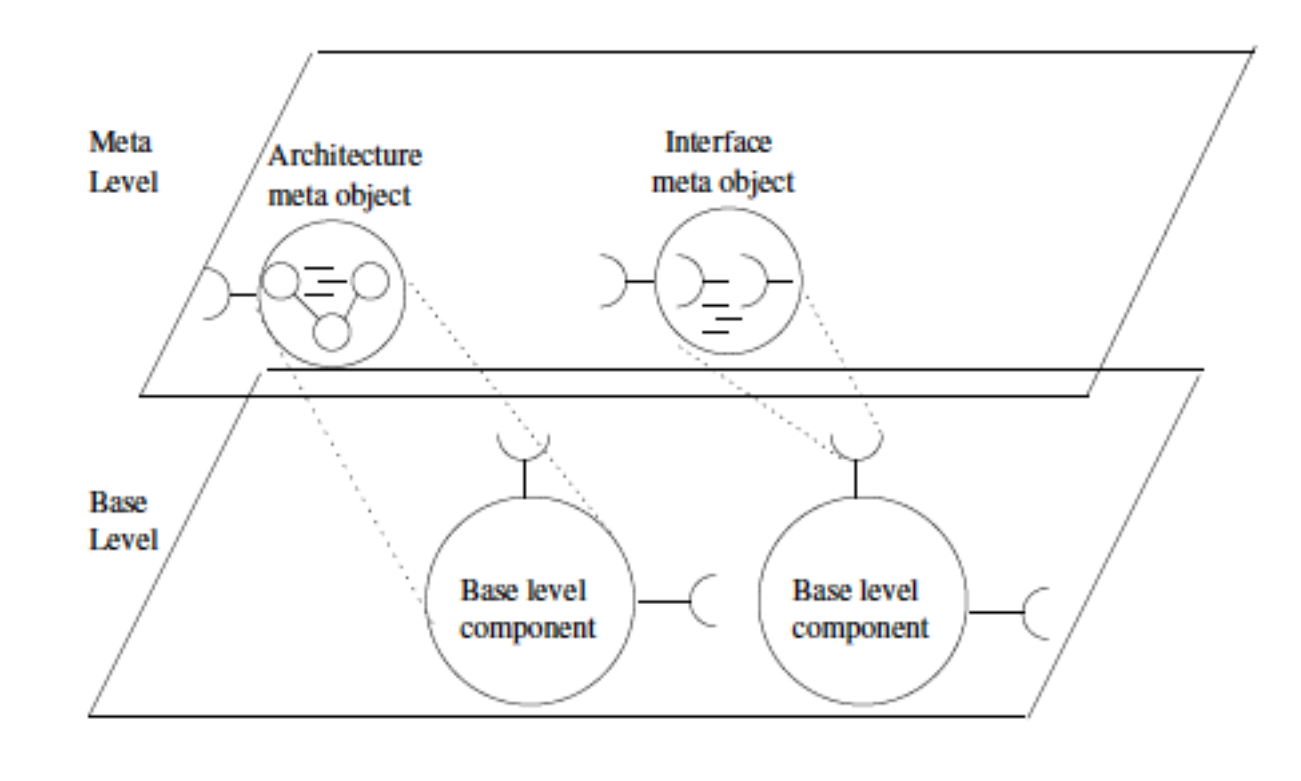
```
interface ICallback{
  boolean veto(in short operation,
               in vetoParameters parameters);
  void receive(in short operation,
               in reportParameters parameters);
};
```

Uso do Callback no OpenCom

```
typedef int status;
typedef long GlobalID;
typedef short extension_type;
interface extensionsCF
{
    status load(in string componentType,
                in GlobalID loader_inst_ID,
                in GlobalID caplet_inst_ID,
                out GlobalID ID);
    status unload(in GlobalID ID);
    status instantiate(in GlobalID ID,
                       out GlobalID compID);
    status destroy(in GlobalID compID);
    status bind(in GlobalID interface_ipnt,
                in GlobalID receptacle_ipnt,
                in GloblaID binder_inst_ID,
                out GlobalID connID);
    status extend(in GUID componentType,
                  in extension_type ext_type
                  out GlobalID compID);
    status setdefaultextension(in extension_type ext_type,
                               in GlobalID compID);
    status notify(in GlobalID compID);
};
```

Uso do Callback no OpenCom

- A meta-arquitetura usando o callback pode certificar-se que uma operação possa ser realizada
- Da mesma forma fazer as atualizações na topologia da arquitetura



Finalizando...

- Abordamos hj:
  - N Self Checking Programming
  - Consensus Recovery Block
  - Callback
- Prx aula: *Acceptance Voting*

Acceptance Voting

Prof. Jó Ueyama

Acceptance Voting (AV)

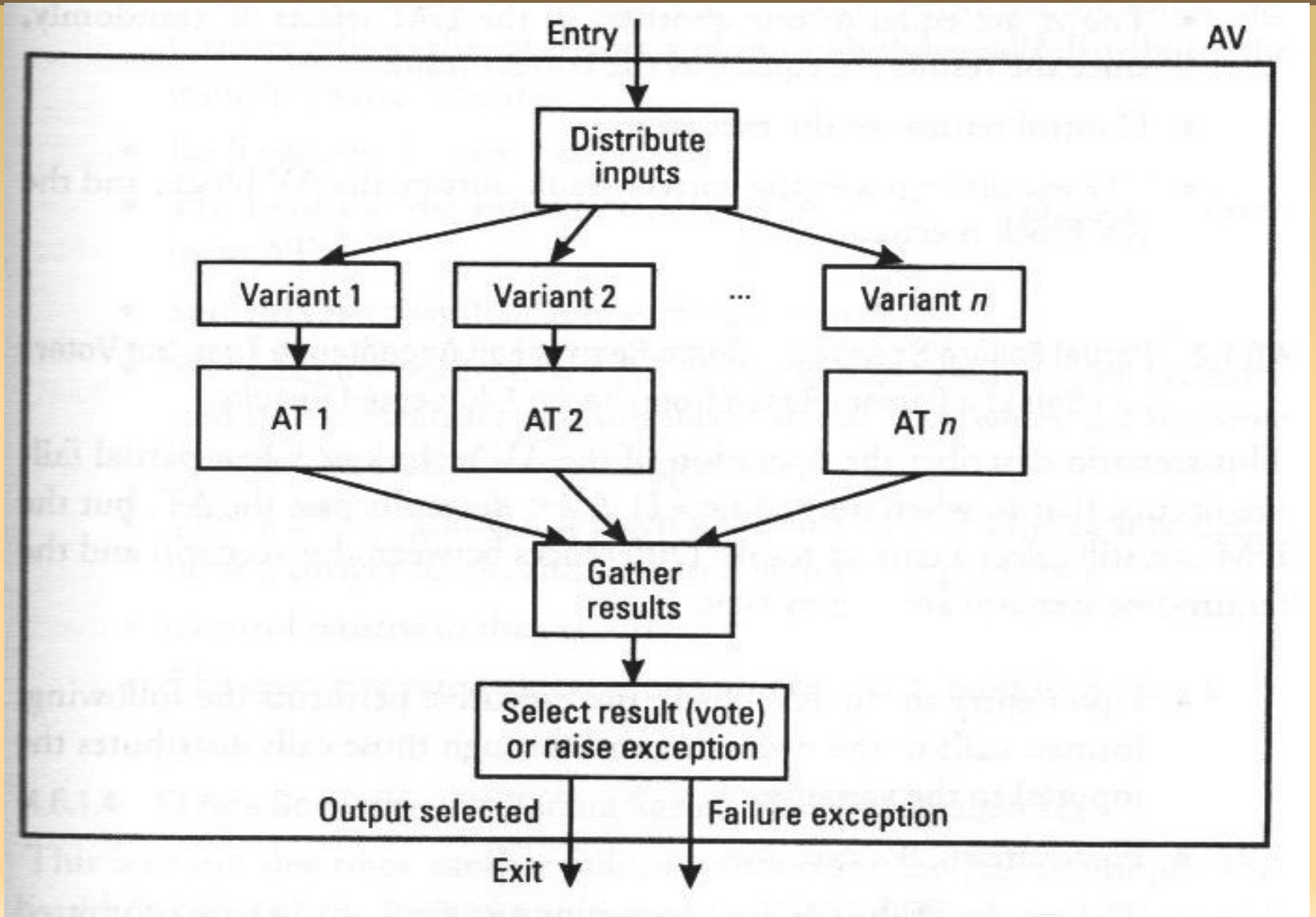
- O mecanismo utiliza as seguintes tecnologias:
  - AT
  - *Voting type DM*
  - *Forward recovery*
- Todas as variantes são executadas em paralelo e os resultados passam por um AT
- Só os que passam por um AT são enviados a um *voter* DM
- O *voter* é dinâmico, pois tem um número variável de resultados (máximo de *n*)

Acceptance Voting (AV)

- O AT possui a operação seguindo o pseudo-código abaixo

```
run Variant 1, Variant 2, ..., Variant n
    ensure   Acceptance Test 1 by Variant 1
    ensure   Acceptance Test 2 by Variant 2
    ...
    ensure   Acceptance Test n by Variant n
[Result i, Result j, ..., Result m pass the AT]
if (Decision Mechanism (Result i, Result j,
                ..., Result m))
    return Result
else
    return failure exception
```

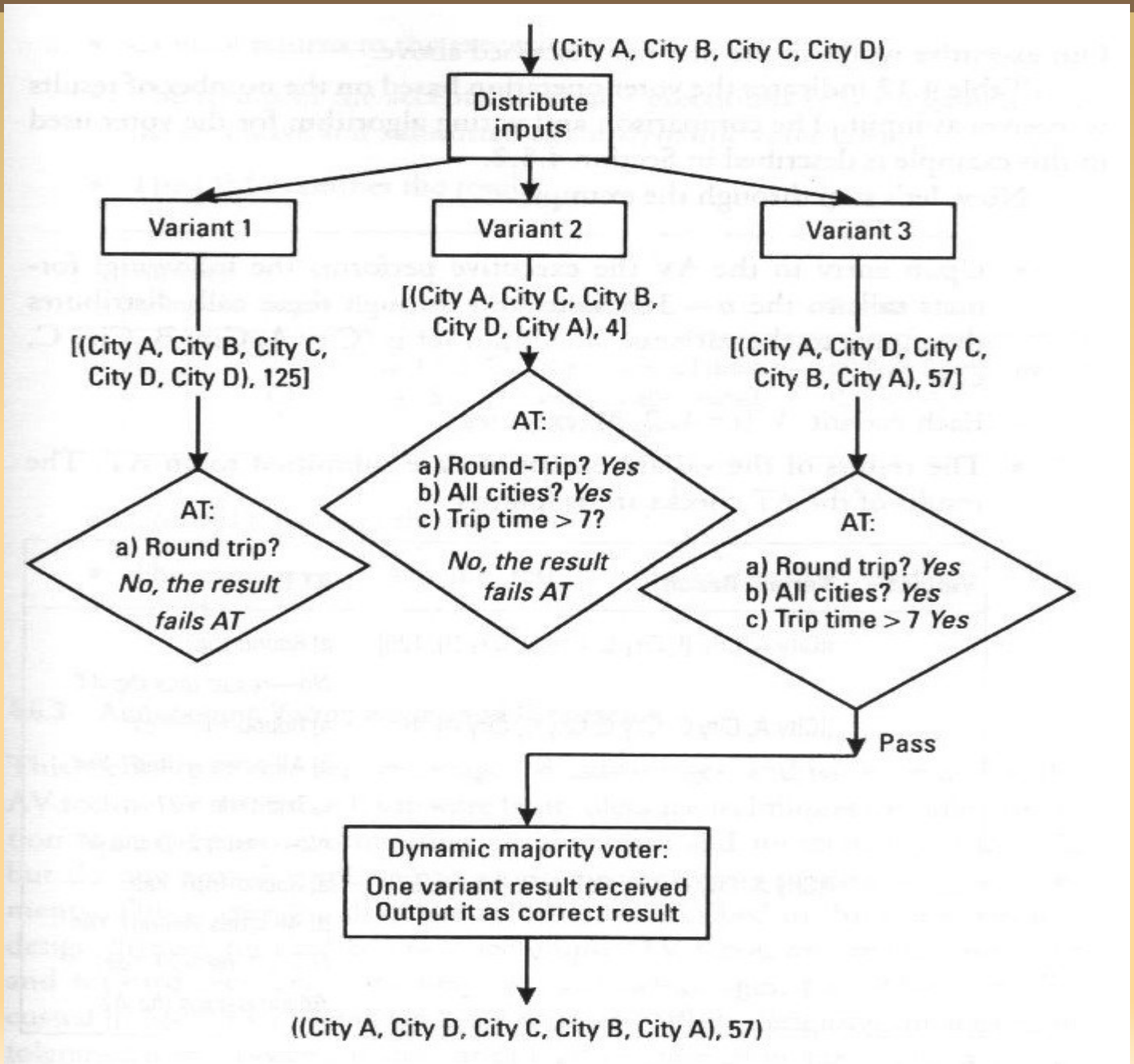
AV Estrutura e Operação



AV - Discussão

- Executado em ambientes multiprocessados
- Pode utilizar o mesmo AT para todas as variantes (caso do exemplo)
- O modelo raramente requer que as variantes interrompam os seus serviços. Por que?
- A confiabilidade do AV é muito dependente do AT
- A técnica é muitas vezes utilizada para testar o RcB
- O AT não precisa ser tão sofisticado quanto os ATs do RcB

Exemplo com o AV



Finalizando o Capítulo 4...

- Diversidade em software tolerante a falhas
  - Recovery Blocks
  - N-Version Programming
  - Distributed Recovery Blocks
  - N-Self Checking Programming
  - Consensus Recovery Block
  - Acceptance Voting
- Lembrando que este assunto foi baseado nos dois livros
- Próximo assunto: Diversidade de dados em software tolerante a falhas