



VRing: Um Algoritmo Distribuído em Anel

Sistemas Distribuídos

Prof. Elias P. Duarte Jr.

Universidade Federal do Paraná (UFPR)

Departamento de Informática

www.inf.ufpr.br/elias/sisdis

Sumário

- O Algoritmo VRing (Virtual Ring - Adaptive DSD)
- Uma Breve Revisão de Diagnóstico
- Modelo de Sistema e Premissas
- Descrição do Algoritmo
- Especificação do Algoritmo
- Prova de Correção
- VRing: *Diagnosability*, Latência e Número de Testes

O Algoritmo VRing

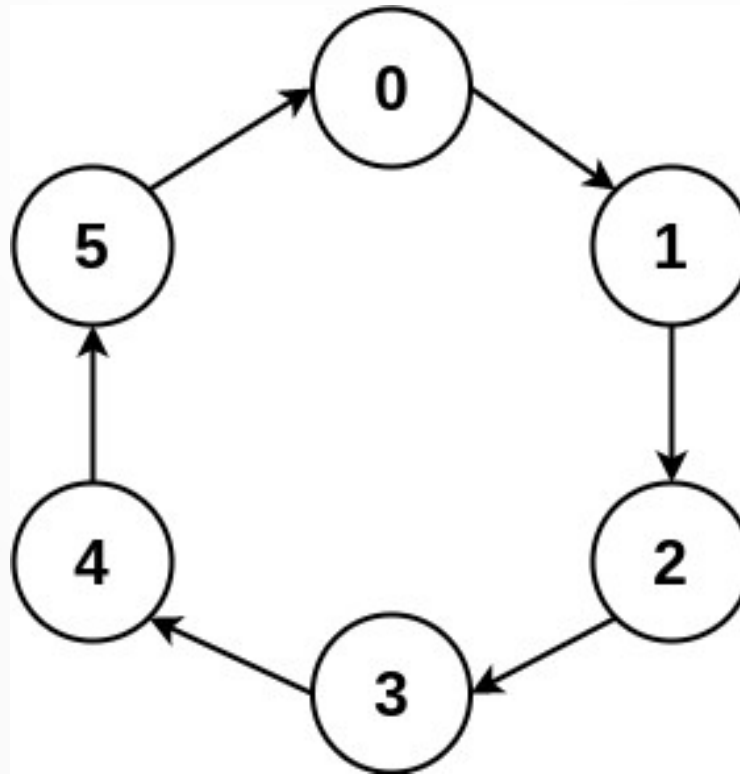
- O primeiro algoritmo distribuído da nossa disciplina!
- Este é um anel que resiste a falhas! Um processo forma enlace com o processo sem-falha *seguinte*
- Os processos formam um anel virtual (ou anel lógico): *Virtual Ring*
 - Processos têm identificadores sequenciais, de 0 até (N-1)
 - Operação módulo : 0 segue (N-1)

VRing Original: Adaptive-DSD

- *Adaptive Distributed System-Level Diagnosis*
- O algoritmo foi originalmente proposto no contexto de diagnóstico em nível de sistema
- Processos executam testes sobre os demais processos
- Objetivo final: determinar os estados de todos os processos → quais processos do sistema estão falhos?
- Modelo de falhas: *crash*

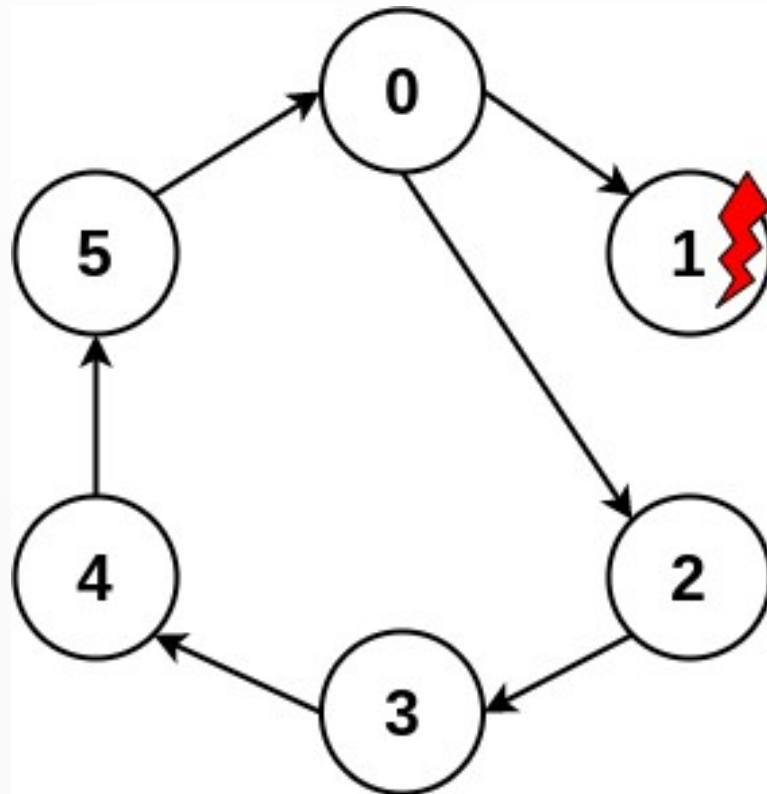
○ Algoritmo *Virtual Ring: VRing*

- Cada processo testa sequencialmente



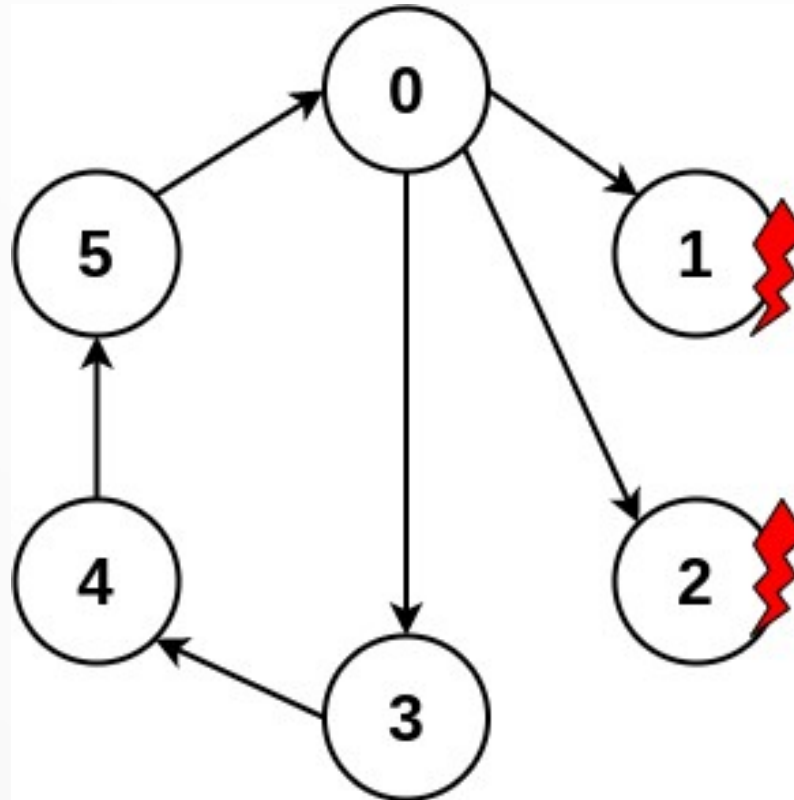
VRing: Testando um Nodo Falho, O Testador Segue até Testar um Correto

- Testes são executados sequencialmente até encontrar um nodo correto: anel que não se rompe



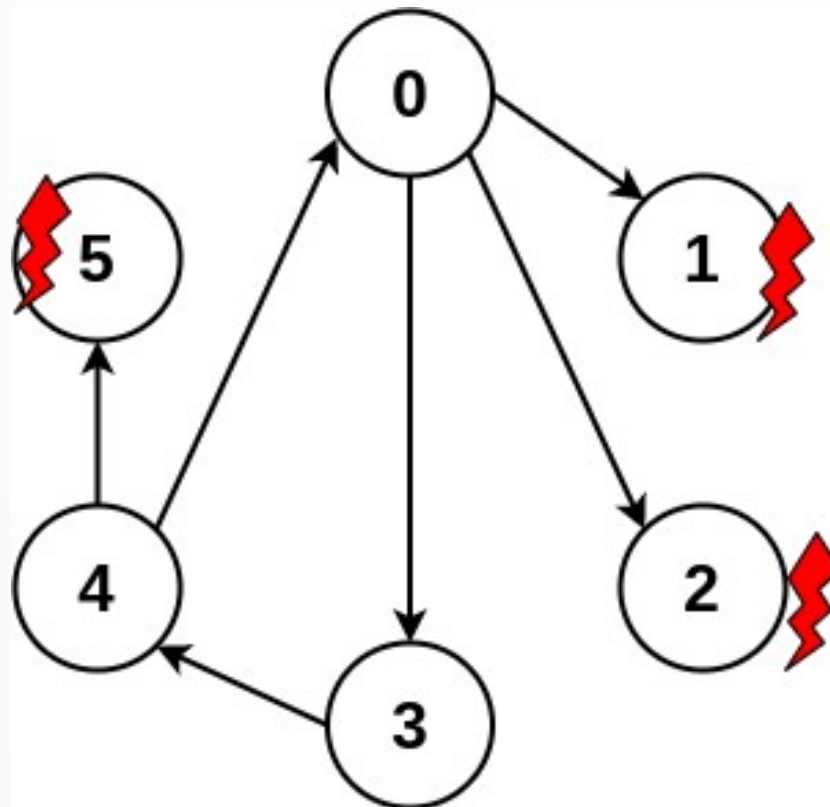
VRing: Testando um Nodo Falho, O Testador Segue até Testar um Correto

- Testes são executados sequencialmente até encontrar um nodo correto: anel que não se rompe



VRing: Testando um Nodo Falho, O Testador Segue até Testar um Correto

- Testes são executados sequencialmente até encontrar um nodo correto: anel que não se rompe



Diagnóstico: Recapitulando

- Considere um sistema distribuído S , que consiste de N processos
$$S = \{0, 1, 2, \dots, (N-1)\}$$
 - Cada processo tem um identificador sequencial, os identificadores começam em 0
- Considere que cada processo pode estar em um de dois estados possíveis: **falho** ou **correto**
- O objetivo do Diagnóstico em Nível de Sistema é determinar os estados dos N processos

Monitoramento Baseado em **Testes**

- Para descobrir os estados dos processos, o Diagnóstico utiliza **testes**, que são executados entre os processos
- O teste consiste da execução de uma bateria de procedimentos que permitem determinar o estado do processo testado
- Como implementar um teste?
 - Totalmente dependente da tecnologia do sistema

VRing: O Grafo de Testes

- O grafo de testes $G = (V, T)$ tem como vértices os processos do sistema no conjunto V
- O conjunto de arcos (arestas direcionadas) T , consiste dos testes executados
- Assim, existe um arco (a, b) em T se a e b são vértices pertencentes a V e a testa b
- No algoritmo VRing: no anel consideramos apenas arcos correspondentes a testes de processos corretos
- O diagnóstico é estático: o próximo evento só ocorre após o evento anterior ter sido diagnosticado
 - Evento: ou um processo correto falha ou vice-versa

Os Processos Corretos

- Se um processo correto executa um teste, então:
 - Executa o teste corretamente
 - Reporta corretamente o resultado do teste executado

Testes 100% Corretos

- Observe que para um processo correto ser sempre capaz de determinar com 100% de precisão o estado do processo testado...
- O sistema deve ser síncrono
 - Devem haver limites de tempo conhecido para a transmissão de mensagem e execução de tarefas
 - Se os limites não existem: uma resposta de teste que atrasa pode levar à conclusão incorreta de que o processo está falho
- Assim vamos assumir o modelo síncrono

Outras Premissas do VRing

- Os canais de comunicação são perfeitos
- E jamais falham
- Apesar de que os processos formam um anel virtual, o sistema é *fully-connected*
 - Qualquer processo é capaz de testar qualquer outro processo

Diagnóstico Adaptativo e Distribuído

- O VRing é adaptativo, no sentido que os próximos testes a serem executados são definidos com base nos resultados dos testes anteriores
- O VRing é distribuído, no sentido que os processos executam testes, obtém resultados de testes e completam o diagnóstico
- O diagnóstico é executado em “intervalos de teste” (*testing intervals*) e o progresso ocorre em “rodadas de testes” (*testing rounds*)
- Dois conceitos essenciais - mas fáceis de confundir um com o outro! Cuidado!!! Fique atento(a)!!!

Intervalos Vs. Rodadas de Testes

- Os processos executam testes periodicamente, em intervalos de testes, que são intervalos de tempo constantes, pré-definidos
 - marcados no relógio local de cada processo
 - por exemplo: 30 segundos ou 10 milissegundos
- Uma rodada de testes é definida como o intervalo de tempo em que todo processo sem-falha testou pelo menos um outro processo sem-falha
 - ou testou todos os demais processos falhos

Pensando sobre Rodadas de Testes

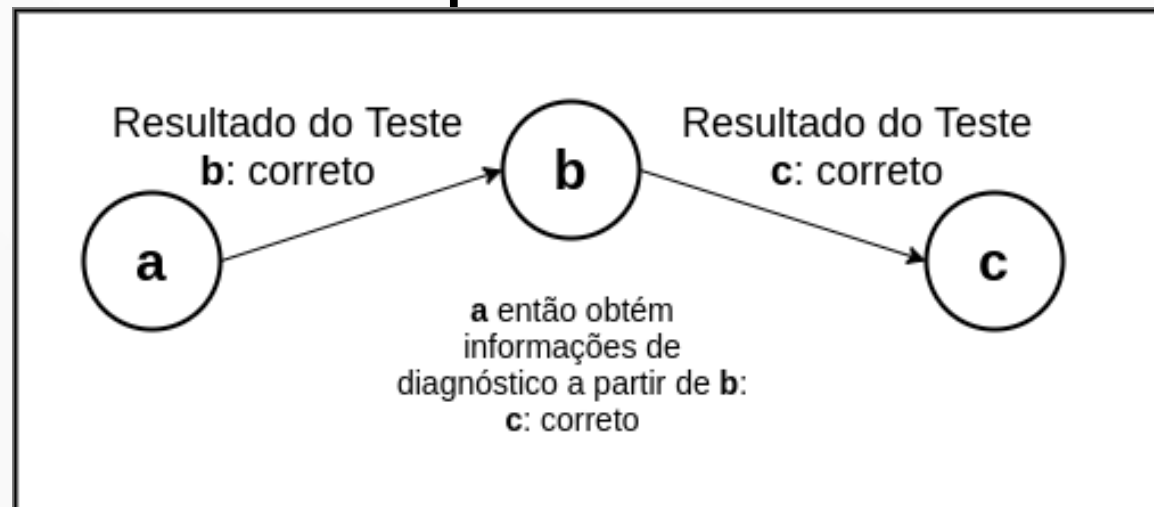
- Em um sistema real é difícil identificar as rodadas de testes
- Não há relógio global: cada processo tem um relógio local, mas não são sincronizados
- Um processo pode rapidamente testar outro processo correto...
- ... enquanto outro pode ter que testar dezenas de falhos antes...
- Quem determina a rodada é o processo mais lento

Latência de Diagnóstico

- A latência do diagnóstico é o número de rodadas de testes que devem ser executadas para completar o diagnóstico
- Situação de falhas: conjunto de estados de todos os processos
- Podemos medir a latência para o diagnóstico de todo o sistema em uma determinada situação de falhas
- Ou podemos medir a latência para o diagnóstico de um evento

Fluxo de Informações de Diagnóstico

- As rodadas de testes são perfeitas para acompanhar o progresso do algoritmo
- Quando um processo correto testa outro processo correto...
- ... pode obter as informações de diagnóstico que o processo correto possui

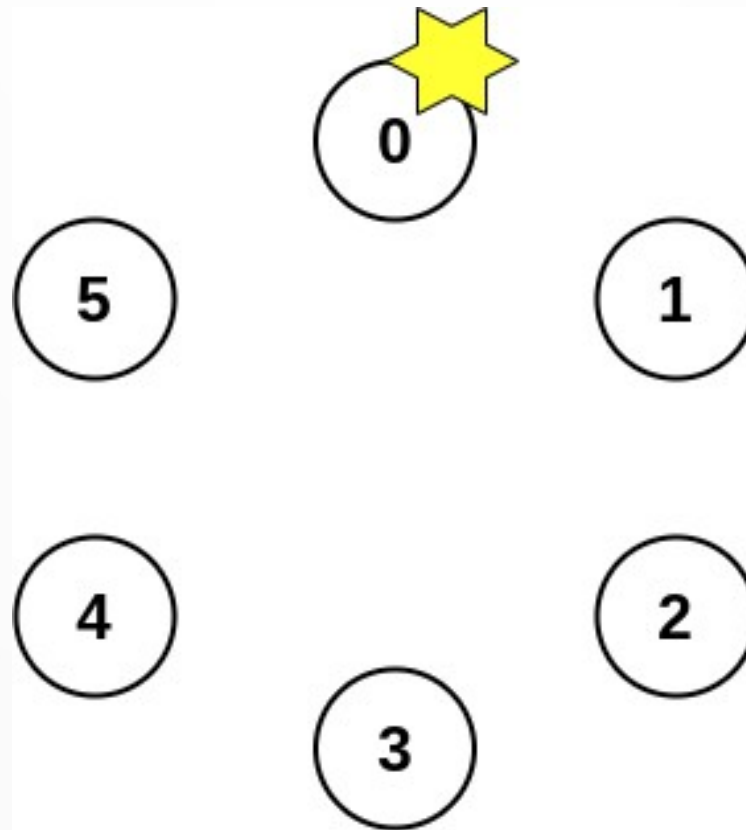


Fluxo de Informações de Diagnóstico

- Após 1 rodada de testes, podemos dizer que todos os processos corretos obtiveram informações de outros processos corretos
- As informações de diagnóstico fluem no sentido oposto dos testes
- Considere que um evento ocorreu no processo 0
 - Evento: ou um processo correto falha ou vice-versa

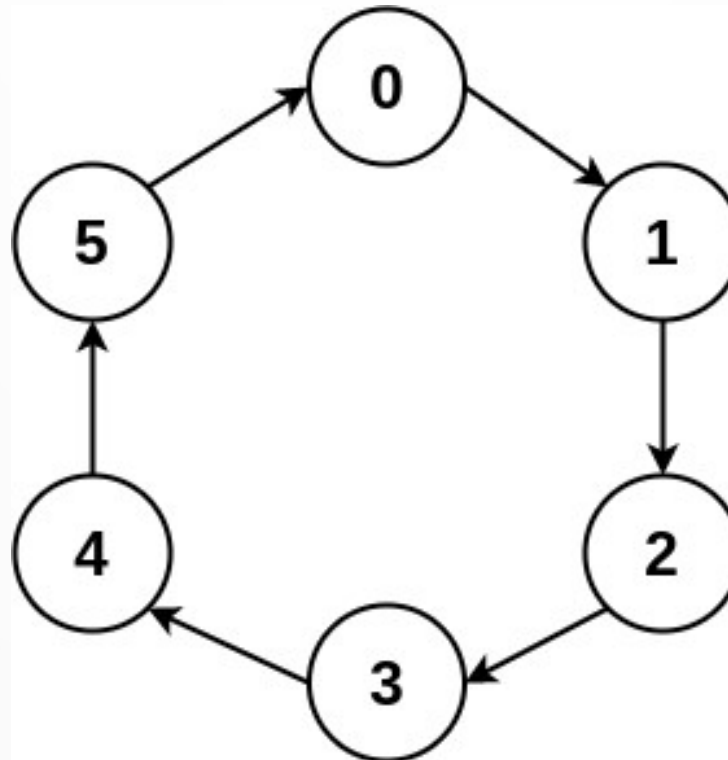
VRing: Fluxo de Informações

- Um evento ocorre no processo 0



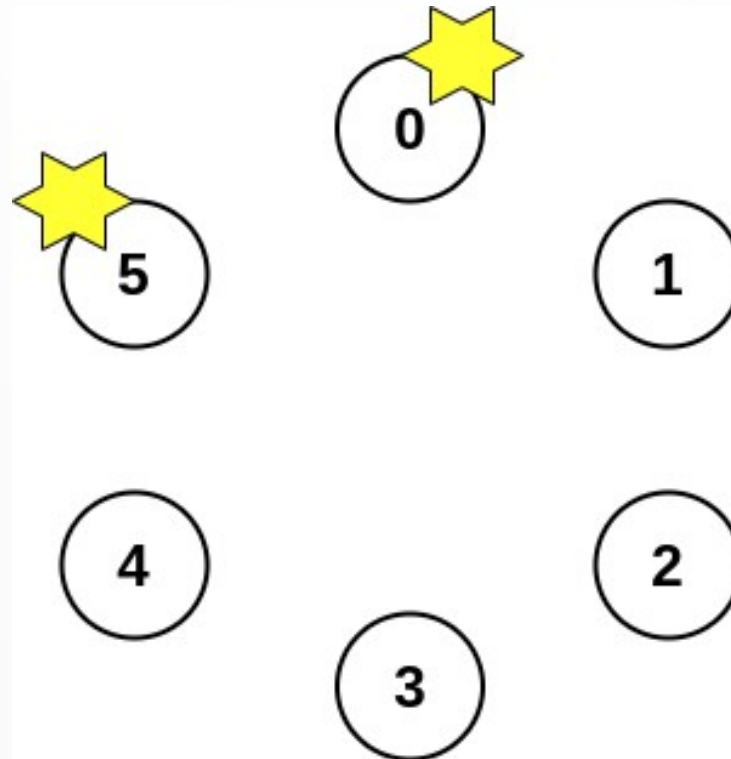
VRing: Fluxo de Informações

- Acontece uma rodada de testes



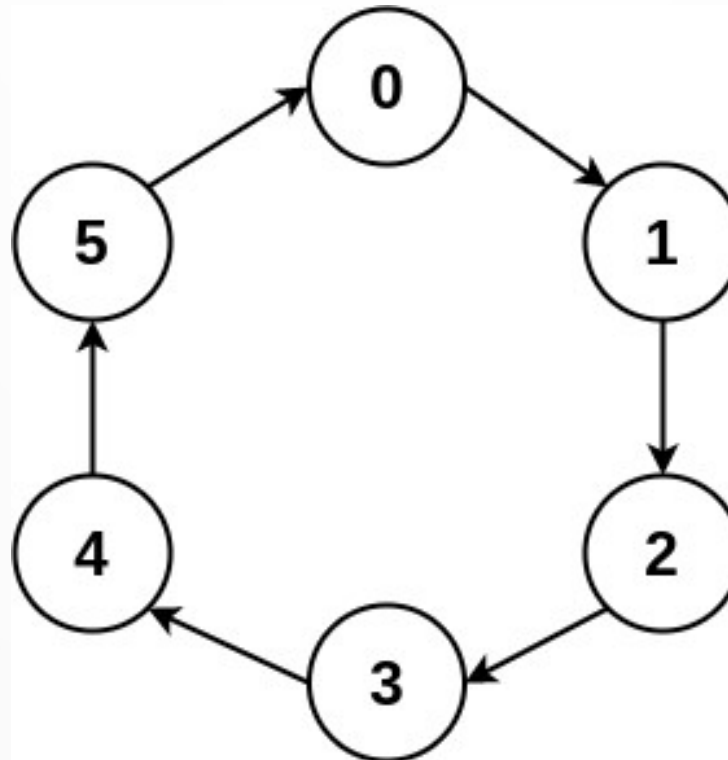
VRing: Fluxo de Informações

- O processo 5 detecta o evento



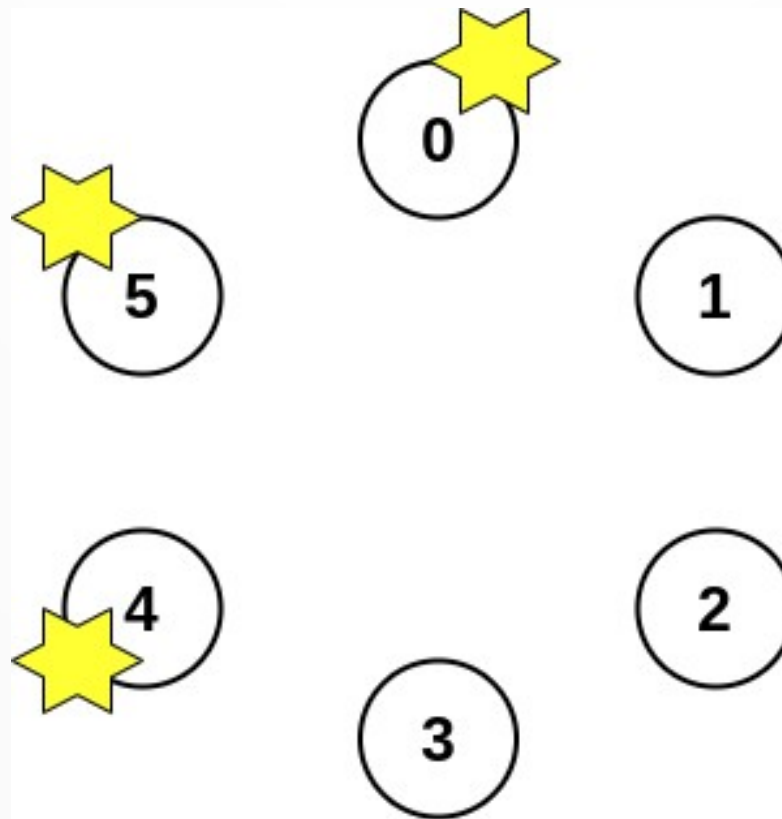
VRing: Fluxo de Informações

- Acontece uma rodada de testes



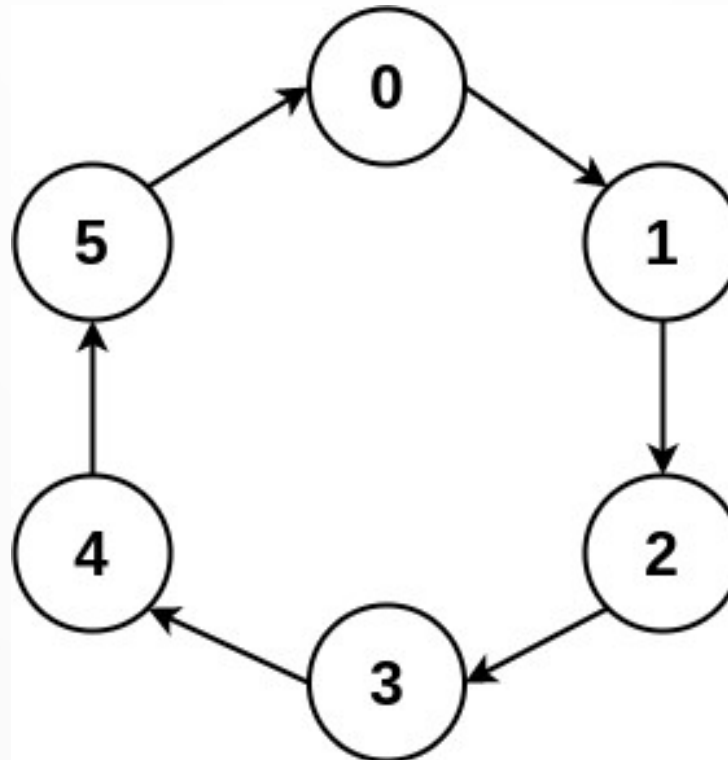
VRing: Fluxo de Informações

- O processo 4 descobre o evento



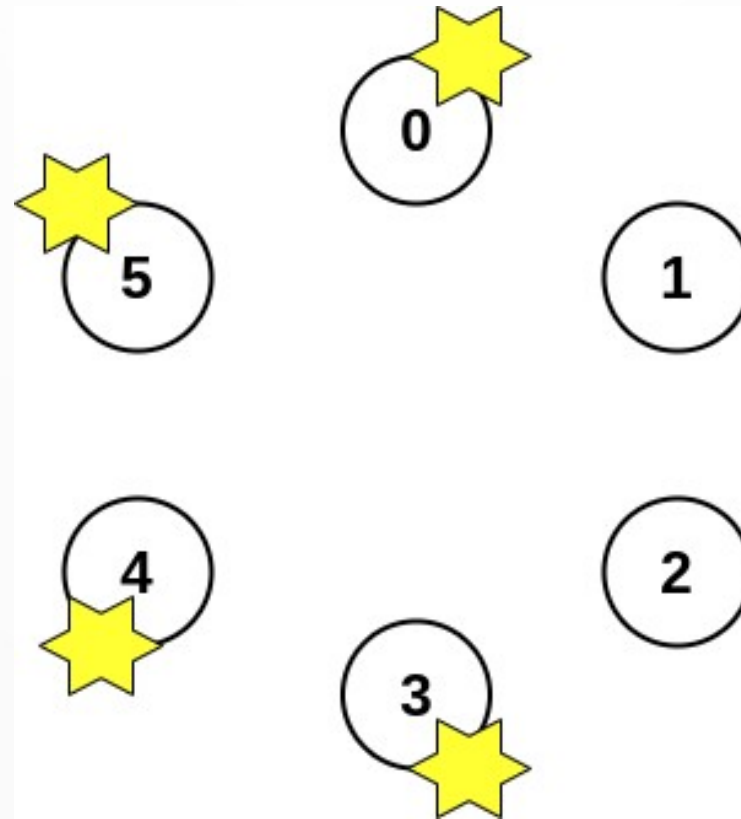
VRing: Fluxo de Informações

- Acontece uma rodada de testes



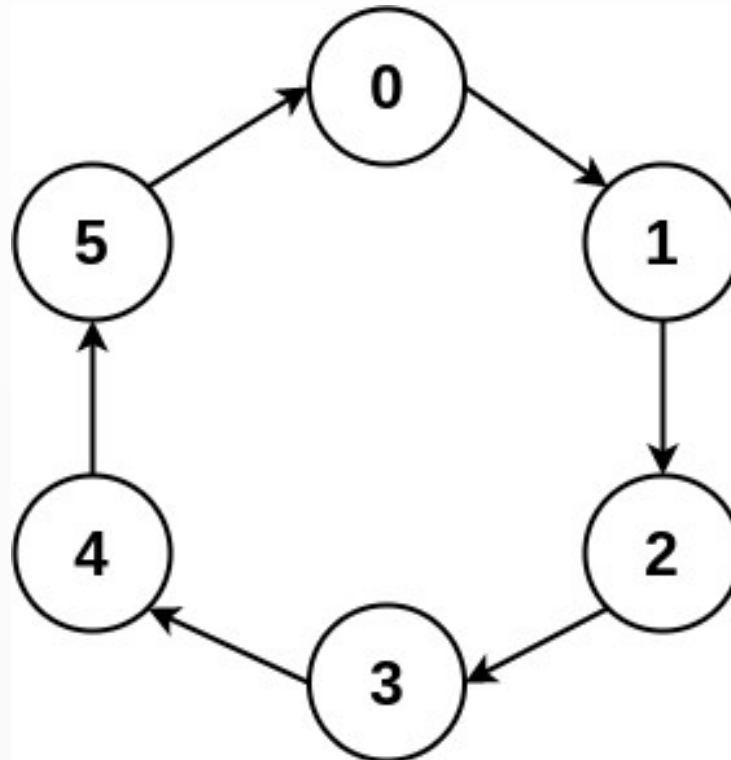
VRing: Fluxo de Informações

- O processo 3 descobre o evento



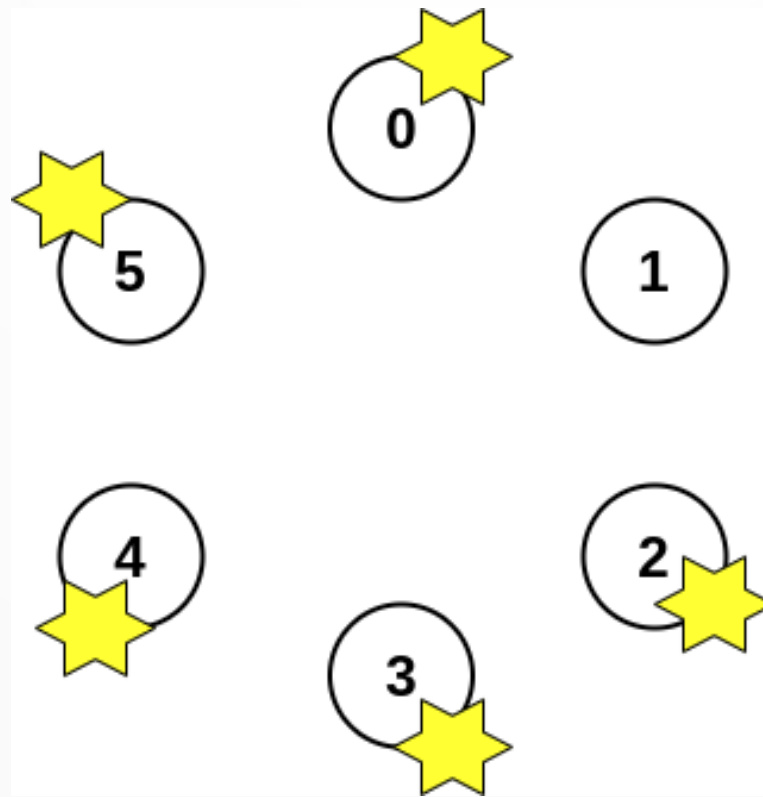
VRing: Fluxo de Informações

- Acontece uma rodada de testes



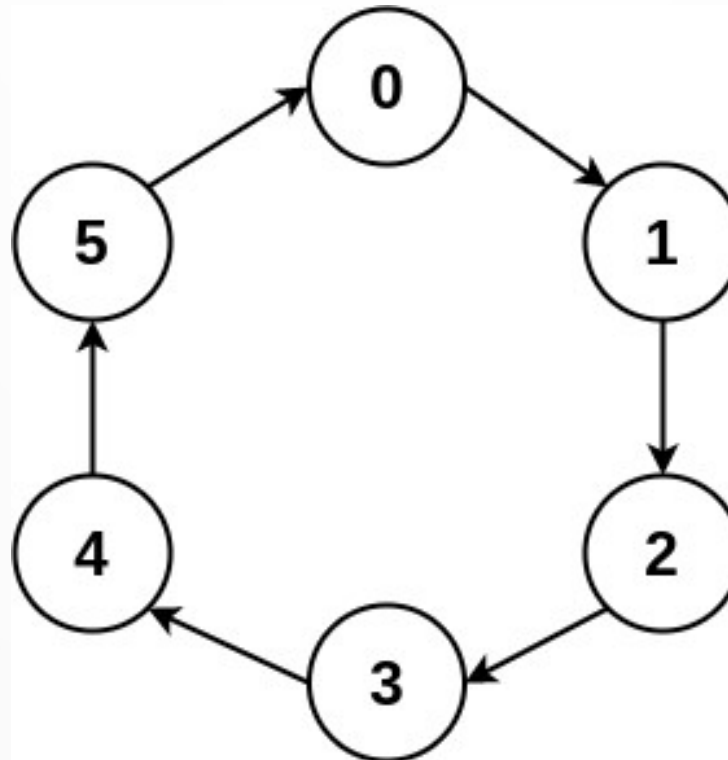
VRing: Fluxo de Informações

- O processo 2 descobre o evento



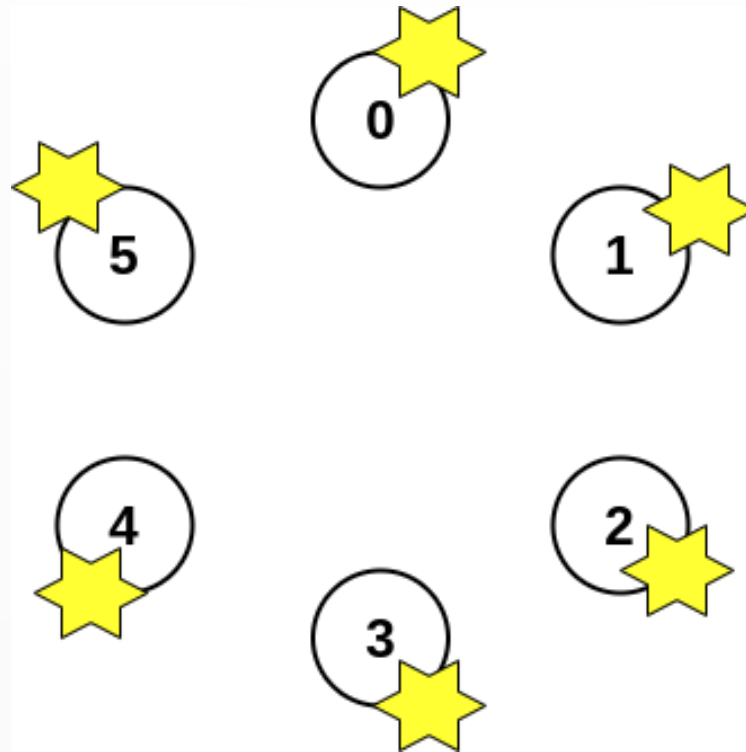
VRing: Fluxo de Informações

- Acontece uma rodada de testes



VRing: Fluxo de Informações

- Diagnóstico completo: todos os processos



O Algoritmo VRing

- A referência principal do VRing:

R. P. Bianchini Jr., R. W. Buskens, "Implementation of On-Line Distributed System-Level Diagnosis Theory," *IEEE Transactions on Computers*, Vol. 41, No. 5, 1992.

O Vetor State[N]

- No nosso algoritmo os processos mantêm informações sobre estado no vetor State[N]
- Vamos manter um contador de eventos (*timestamp*)
- Inicialmente 0 → correspondendo ao estado correto
- Quando um evento é descoberto pelo processo j no processo i: $State_j[i]++$
- 1: falho, 2: correto; 3: falho; ...
 - State[i] é par: i correto
 - State[i] é ímpar: i falho

O Algoritmo VRing

// executado pelo processo i

// a cada intervalo de testes

Início

$j \leftarrow i$;

repita

$j \leftarrow (j+1) \bmod N$;

teste o nodo j ;

se j está correto

então obtenha $State_j[]$;

para todo k não testado neste intervalo

atualize $State_i[k] \leftarrow State_j[k]$;

até (encontrar j correto) ou (testar todos falhos);

Fim.

VRing: Prova de Correção

- Teorema 1: Dado o sistema S em uma determinada situação de falha:
 - em uma rodada de testes do algoritmo VRing
 - no grafo de testes $G=(V,T)$ há um caminho direcionado entre quaisquer 2 processos corretos
- Prova por absurdo
- Escolha um par de processos corretos x & z tal que não exista um caminho direcionado entre eles e...
- ... $(z-x)$ é o mínimo, para todos os pares de processos sem caminho direcionado

Continuando o Teorema 1

- Identifique *o maior* processo y , $y < z$, tal que existe um caminho direcionado de x para y
- Após 1 rodada de testes y deve ter testado um processo correto w
- Como $y < z$, w deve ser maior que z
 - veja que w não pode estar entre $y < w < z$, pois neste caso y não seria o maior
- Entretanto, pela definição do algoritmo, y obrigatoriamente testou z antes de testar w : absurdo!

VRing: Corolário 1

- Corolário 1: Dado o sistema S em uma determinada situação de falhas, em uma rodada de testes do algoritmo VRing o grafo de testes $G=(V,T)$ consiste de um ciclo direcionado conectando todos os processos corretos de S
- Pelo Teorema 1 existe um caminho direcionado entre qualquer par de processos corretos no grafo de testes $G=(V,T)$
- De acordo com o algoritmo VRing, cada processo correto executa 1 único teste em outro processo correto por rodada de testes
- Um ciclo direcionado é a única estrutura que satisfaz estas duas condições

VRing: Teorema 2

- Teorema 2: Considere o sistema S em uma determinada situação de falhas. Após N rodadas de testes do algoritmo VRing, a entrada $State_i[j]$ é idêntica para todos os processos $0 \leq i < N$
- Escolha um processo correto qualquer x
- Pela definição do algoritmo, se x testa z em uma rodada de testes atualiza a entrada $State_x[z]$
- Após duas rodadas de testes, o processo correto y testa o processo correto x e atualiza $State_y[z] \leftarrow State_x[z]$
- Nas próximas rodadas de testes, mais nodos atualizam a entrada correspondente com $State[z] \rightarrow State_x[z]$
- O caminho mais longo no ciclo tem N vértices
- Portanto em no máximo N rodadas todos os processos mantêm o mesmo valor para $State[z] \leftarrow State_x[z]$

Latência do Algoritmo VRing

- Portanto, no pior caso, a latência de diagnóstico do algoritmo VRing é de N rodadas de testes
- Pergunta: qual a latência no melhor caso?

Latência do Algoritmo VRing

- Portanto, no pior caso, a latência de diagnóstico do algoritmo VRing é de N rodadas de testes
- Pergunta: qual a latência no melhor caso?
- 1 rodada de testes!

Diagnosability do VRing

- *t-Diagnosability*: t é o maior número de processos que podem falhar tal que o diagnóstico ainda completa corretamente
- No VRing: $t = N-1$
- Ótimo (matematicamente): se todos menos 1 processo falham, este processo que se mantém correto continua fazendo corretamente o diagnóstico do sistema

Número de Testes Executados

- Qual o maior número de testes executados pelo algoritmo VRing em uma rodada?
- Considere que todos os processos estão corretos

Número de Testes Executados

- Qual o maior número de testes executados pelo algoritmo VRing em uma rodada?
- Considere que todos os processos estão corretos
- Resposta: N testes

Número de Testes Executados

- Qual o maior número de testes executados pelo algoritmo VRing em uma rodada?
- Considere que todos os processos estão corretos
- Resposta: N testes
- Cada processo é testado no máximo 1 vez
- Ótimo (matematicamente): impossível um algoritmo que utilize menos de N testes
 - Cada processo tem que ser testado pelo menos 1 vez por rodada de testes

Conclusão

- O Algoritmo VRing: processos corretos formam um anel virtual que nunca se rompe
- Premissas assumidas pelo algoritmo
- Intervalo de Testes, Rodada de Testes, Latência
- O vetor State[] de *timestamps* (contadores de eventos)
- Especificação do algoritmo VRing em pseudo-código
- Prova de correção, *Diagnosability*, número de testes
- Próxima aula: simulação com SMPL!

Obrigado!
Página da Disciplina
Sistemas Distribuídos:
www.inf.ufpr.br/elias/sisdis