① 12ª Aula

Algoritmos de eleição

→ Necessidade de um coordenador em ambiente distribuído

→ A solução geral consiste em eleger o processo com maior ID (ou IP)

→ Todos os processos precisam concordar com o resultado da eleição

② ## Algoritmo do valentão (Bully)

→ Desenvolvido por Garcia-Molina (1982)

→ Quando um processo percebe que o coordenador não está respondendo requisições, ele inicia uma eleição

→ Convocação da eleição

* P envia mensagem de eleição para todos os processos com IDs maiores

(3)

* Se ninguém responde, P vence a eleição e torna-se coordenador
* Se algum processo com ID maior responde, ele desiste

→ Quando um processo recebe mensagem de eleição

* Se o ID recebido é menor que o dele, envia OK "pelo" remetente para indicar que está vivo e assume a coordenação (com "nova" eleição)

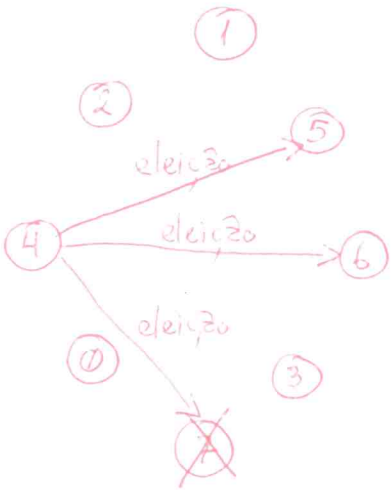
(4)

→ Todos os processos desistem, menos o que tiver maior ID

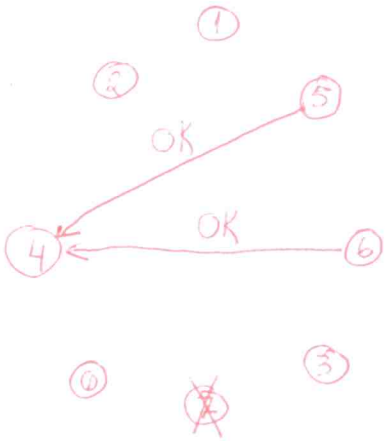
→ Se o processo que estava indisponível voltar é iniciada uma nova eleição

→ Resumo da história: o processo "Chuck Norris" (valentão) sempre vence a eleição

⑤ Exemplo

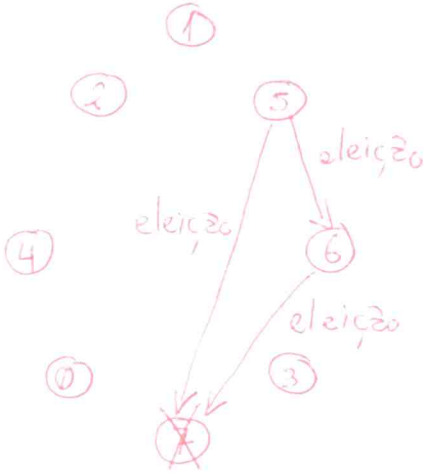


Processo 7 caiu
Processo 4 percebeu e convocou eleição

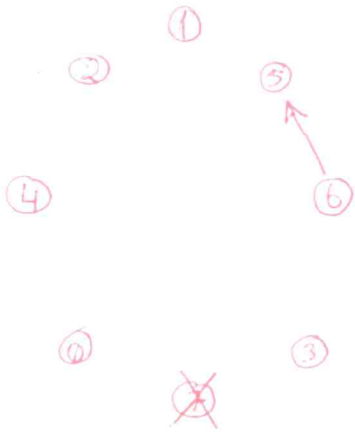


Processos 5 e 6 respondem OK para o processo 4 que desiste

⑥

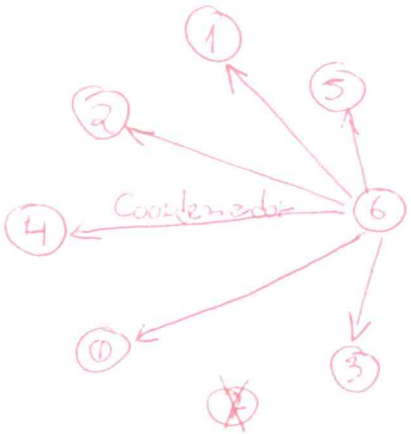


Processos 5 e 6 iniciam/continuam eleição



Processo 6 envia OK para o 5 que desiste

(7)



Processo 6 avisa todos os demais processos que ele é o novo coordenador/líder

(8) Algoritmo em anel

→ Não utiliza token

→ Processos fisicamente ou logicamente ordenados - cada processo conhece seu sucessor

→ Quando um processo percebe que o coordenador caiu ele inicia eleição:

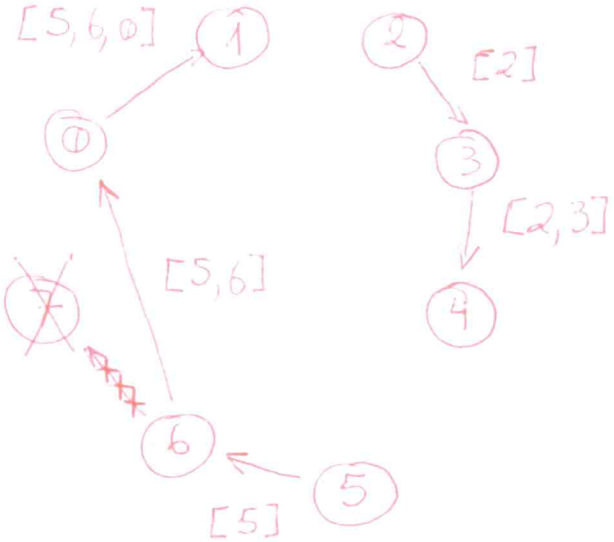
(9)

* O processo envia mensagem de eleição para o sucessor, com seu ID
* Se o sucessor está indisponível, envia para o próximo e assim sucessivamente
* A cada passo, o processo que recebe a mensagem adiciona seu ID e repassa para o sucessor

(10)

- A mensagem deve retornar para quem iniciou a eleição
  * Quem enviou reconhece a mensagem por conter seu ID
  * Decide quem deve ser o coordenador pelo maior ID
  * Avisa com uma nova rodada quem é o novo coordenador

(11) Exemplo



Processo (coordenador) 7 caiu, 2 e 5 percebem e iniciam eleição "simultaneamente", no final o 6 é o novo coordenador

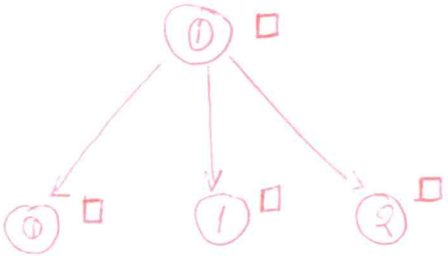
(12) Programação com MPI

Comunicação coletiva (continuação)

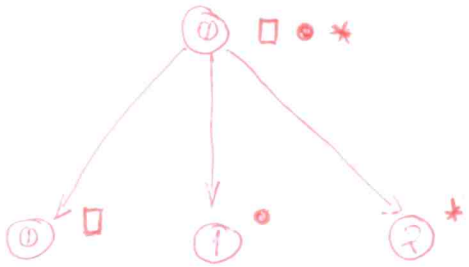
MPI_Scatter()

- Distribui mensagens distintas de um único processo para cada processo do grupo
- MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

(13)



MPI_Bcast()

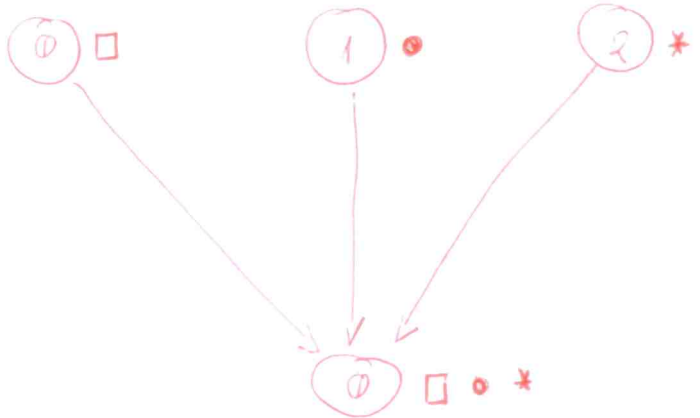


MPI_Scatter()

(14) MPI_Gather()

- Reuni/junta/coleta mensagens distintas de cada processo do grupo para um processo destino
- Esta rotina é a operação inverso de MPI_Scatter()
- MPI_Gather (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

(15)



MPI_Gather()

Cuidado: Não confundir com MPI_Reduce()

(16) Exemplo

```
#include <stdio.h>
#include <stdlib.h>
#incluse <mpi.h>

int numnodes, myid, mpi_err;

#define mpi_root 0

void init_it (int *argc, char ***argv){
    mpi_err = MPI_Init(argc, argv);
    mpi_err = MPI_Comm_size(MPI_COMM_WORLD, &numnodes);
    mpi_err = MPI_Comm_rank(MPI_COMM_WORLD, &myid);
}
```

(17)

```
int main (int argc, char *argv[]){
    int *myray, *send_ray, *back_ray;
    int count;
    int size, mysize, i, K, j, total;

    init_it(&argc, &argv);

    // cada processo terá count elementos do root
    count= 4;
    myray= (int *) malloc (count * sizeof (int));

    // cria dados para ser enviado no root
    if (myid == mpi_root){
        size= count * numnodes;
```

(18)

```
        send_ray = (int *) malloc (size * sizeof (int));
        back_ray = (int *) malloc (numnodes * sizeof(int));
        for (i= 0; i< size; i++)
            send_ray[i]= i;
    }
    // envia diferentes dados para cada processo
    mpi_err = MPI_Scatter ( send_ray, count, MPI_INT,
    myray, count, MPI_INT, mpi_root, MPI_COMM_
    WORLD);
```

(19)

```
// cada processo efetua uma soma local
total = 0;
for( i = 0; i < count; i++)
    total = total + my_ray[i];
printf ("myid = % total = %d \n", myid, total);
// envia as somas locais de volta para o root
mpi_err = MPI_Gather (& total, 1, MPI_INT,
back_ray, 1, MPI_INT, mpi_root, MPI_COMM_
WORLD);
```

(20)

```
// o root imprime a soma global
if (myid == mpi_root){
   total = 0;
   for( i = 0; i < num_nodes ; i++)
      total = total + back_ray[i];
   printf("Resultado dos processos somado = %d \n",total);
}
mpi_err = MPI_Finalize();
}
```

Mais exemplos: http://geco.mines.edu/workshop/aug2011/examples/mpi/index.html