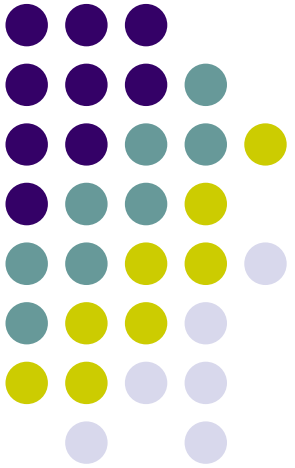
Sincronização em Sistemas Distribuídos

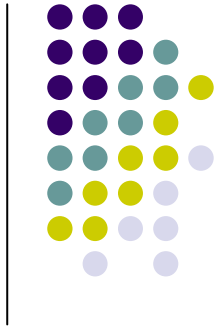
Universidade Federal do ABC

Turma:
Ciência da Computação

Prof. Dr. Francisco **Isidro** Massetto



Sincronização

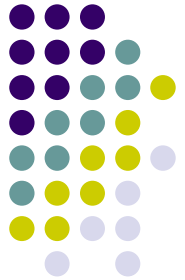


- Como as regiões críticas são implementadas em um SD?
- Como os recursos são alocados aos processos?
- Em SO, os problemas de exclusão mútua e região crítica são solucionados através de semáforos ou monitores.
- Tais métodos utilizam memória compartilhada para implementar a solução, portanto, impossível de ser feito em um SD.
- **Pergunta:** como promover a sincronização em um ambiente distribuído?

Sincronização através do CLOCK

- Os Sistemas Distribuídos utilizam algoritmos distribuídos na implementação de serviços e aplicações.
- Geralmente não é desejável ter todas as informações sobre o sistema em um único lugar.
- Os algoritmos distribuídos apresentam as seguintes propriedades:
  - As informações relevantes são espalhadas pelas múltiplas máquinas;
  - Os processos tomam as decisões baseadas somente em informações locais;
  - Um ponto de falha que paralise todo o sistema deve ser evitado;
  - Não existe relógio comum ou um tempo global.

Clock Lógico

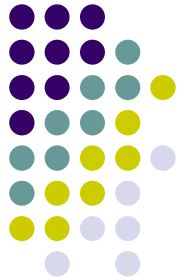


- Em Lamport (1978) - "**Time, Clocks, and the Ordering of Events in a Distributed System**" é provada que a sincronização dos relógios é possível e apresenta o algoritmo para se conseguir isto.
- Posteriormente, em outro artigo ele defende que a sincronização dos clocks não precisa ser absoluta, pelos seguintes motivos:
  - Se dois processos não interagem, não é necessário que seus clocks sejam sincronizados.
  - Usualmente o que importa não é que todos os processos concordem com o exato tempo em que os eventos aconteceram, mas que concordem na ordem em que os eventos ocorreram.

Clock Lógico

- **Clock Lógico** - consistência interna é o que importa e não quanto eles estão próximos do tempo real.
- **Clock Físico** - os clocks não podem diferir do tempo real mais que um determinado valor.

Clock Lógico – Algoritmo de Lamport

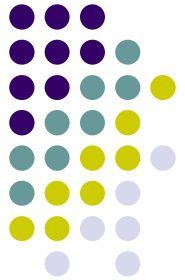


- Lamport definiu a seguinte relação: “acontece-antes”:

  a → b (**a** acontece antes de **b**)

- Significa que todos os processos concordam que primeiro o evento **a** ocorreu e depois disto, o evento **b** ocorreu. Esta relação pode ser observada em duas situações:
  - Se **a** e **b** são eventos no mesmo processo, e **a** ocorre antes de **b**, então a→b é verdadeiro.
  - Se **a** é o evento de uma mensagem sendo enviada por um processo, e **b** é o evento da mensagem sendo recebida por outro processo, então a→b é também verdadeiro.

Algoritmo de Lamport



- A relação “acontece-antes” é transitiva, logo:

  se a→b e b→c, então a→c

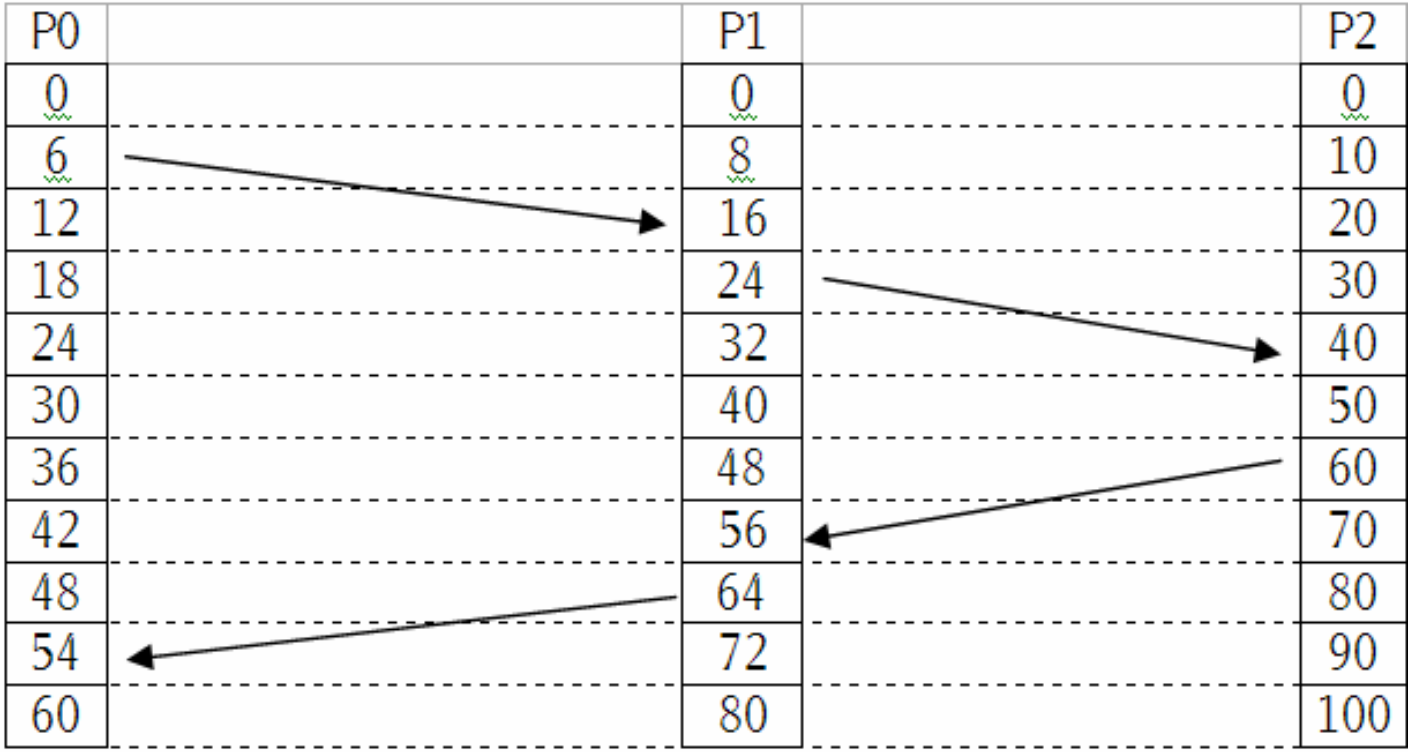
- Se dois eventos **x** e **y**, acontecem em diferentes processos que não trocam mensagens, então x→y não é verdadeiro, nem y→x é verdadeiro.
- Estes eventos são ditos concorrentes, o que significa que nada pode ser dito sobre quando eles aconteceram.

Algoritmo de Lamport

- É necessário um modo de medição de tal forma que para todo evento **a**, possa ser assumido que ele ocorreu em um tempo C(a) no qual todos os processos concordem.

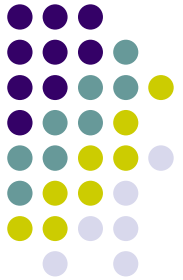
Se a→b, então C(a) < C(b).

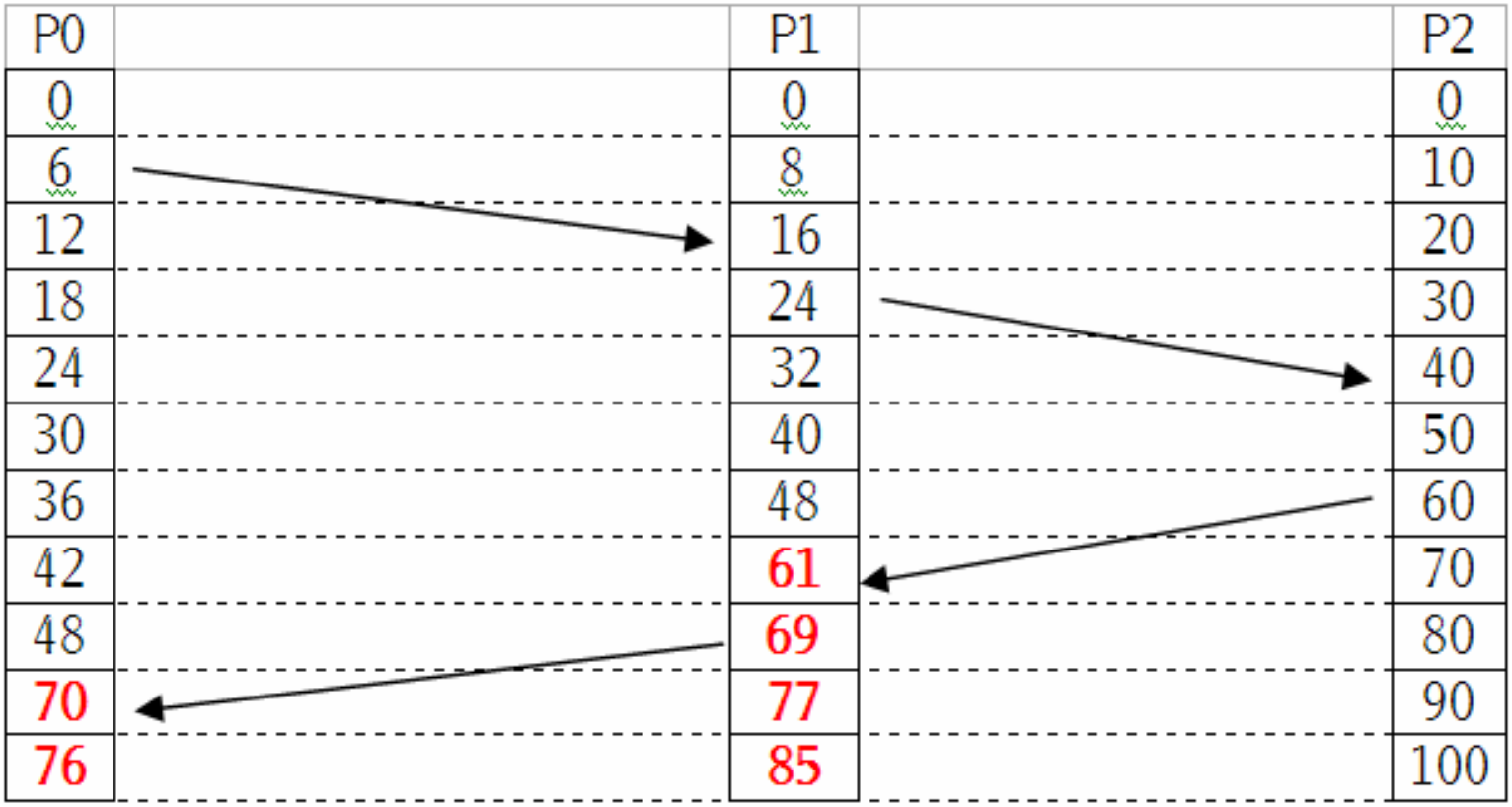
Algoritmo de Lamport



- O que acontece de estranho entre as trocas de mensagens?

Algoritmo de Lamport

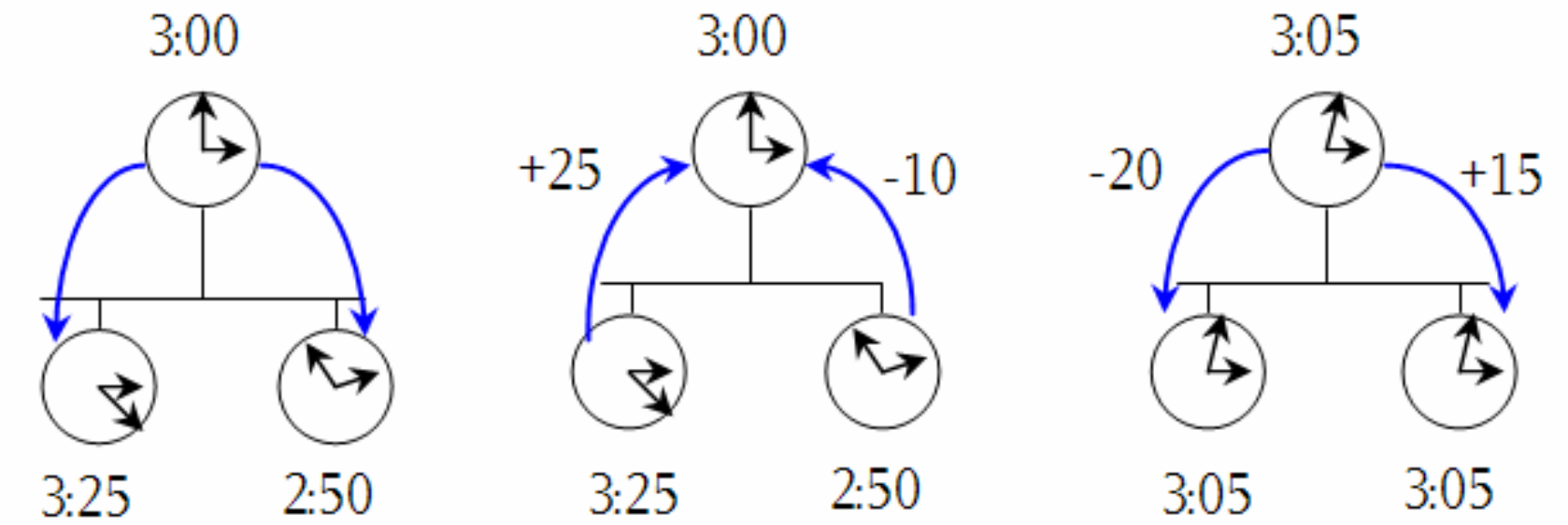




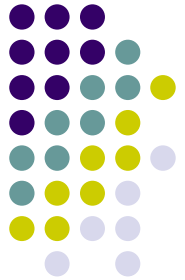
Algoritmo pra Sincronização de CLOCK – Algoritmo de Berkeley

- Nenhuma máquina tem um receptor de Tempo Universal Coordenado.
- O Servidor de Tempo é ativo, e requer, periodicamente de cada máquina, o tempo do seu relógio.
- O Servidor de Tempo calcula a média dos tempos (considerando o tempo dele mesmo) e diz para cada máquina como ajustar seu relógio para ter seu tempo igual à média calculada.

Algoritmo de Berkeley

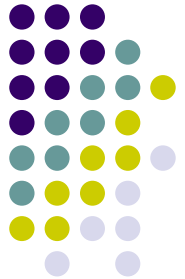


Exclusão Mútua – Algoritmo Centralizado



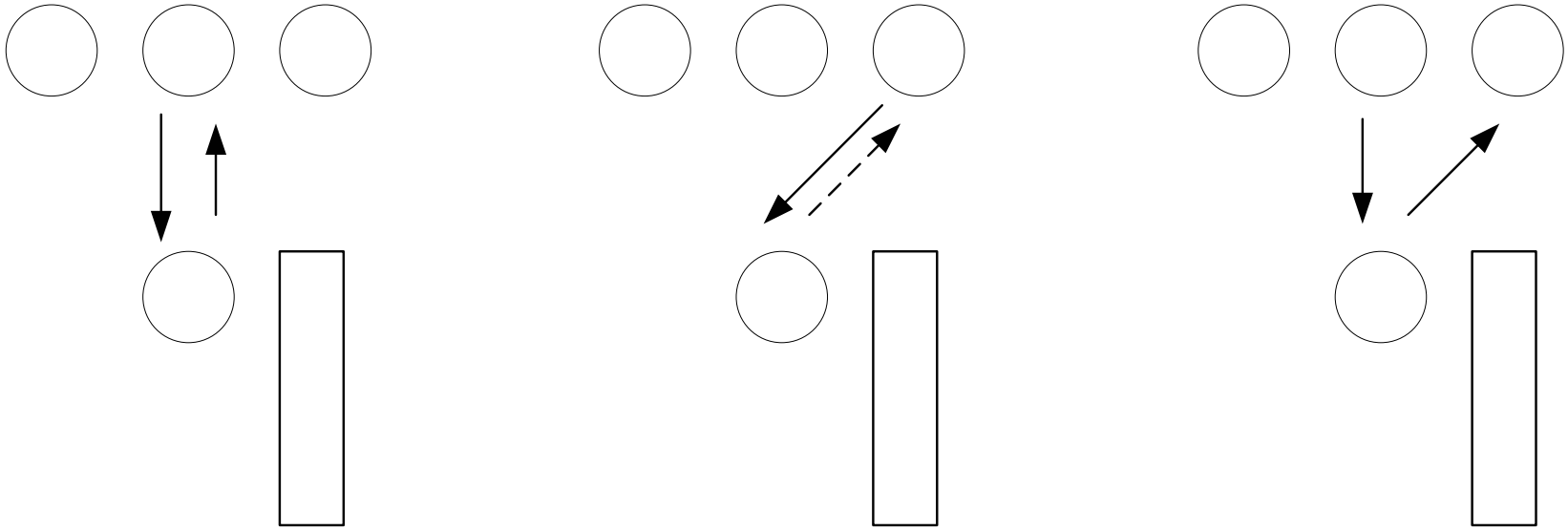
- A melhor maneira de se conseguir exclusão mútua em um sistema distribuído é imitar o que é feito no sistema centralizado (com um único processador).
- Um processo é eleito como **Coordenador**. Quando um processo quer entrar na região crítica, ele envia uma mensagem requisitando ao **Coordenador** permissão para isso.
- Se nenhum outro processo está na região crítica, o **Coordenador** envia uma resposta dando permissão.

Algoritmo Centralizado



- Ao receber a mensagem o processo requisitante entra na região crítica.
- Caso outro processo peça permissão para entrar na região crítica, e o **Coordenador** sabendo que outro processo está na região, não envia resposta bloqueando este processo até que ele possa entrar na região.
- Uma outra opção é enviar uma mensagem negando a solicitação.
- Quando o processo deixa a região, ele envia para o **Coordenador** uma mensagem liberando a região crítica.

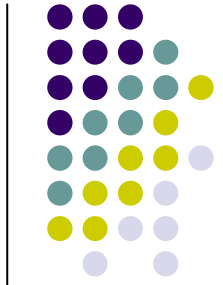
Algoritmo Centralizado



Exclusão Mútua – Algoritmo Distribuído

- O Algoritmo centralizado tem o problema de uma falha no **Coordenador** inviabilizar o mecanismo (todo elemento centralizador é um ponto crítico de falhas).
- Algoritmo distribuído - Quando um processo quer entrar na região crítica, ele constrói uma mensagem contendo o nome da região, número do processo e o tempo corrente.
- A mensagem é enviada para todos os outros processos. Quando um processo recebe uma mensagem de requisição de outro processo sua ação vai depender de sua situação relativa à região crítica

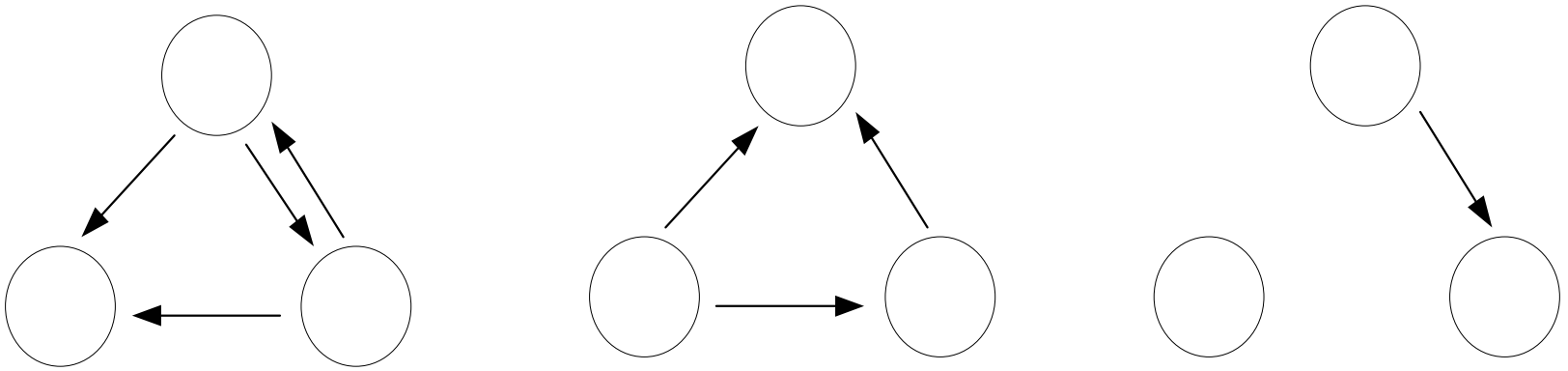
Algoritmo Distribuído



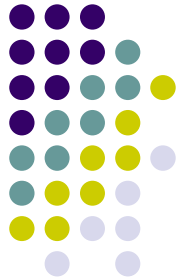
- Possibilidades:
  - Se o receptor não está na região crítica e não quer entrar, ele envia de volta uma msg OK;
  - Se o receptor já está na região, ele não responde e coloca a requisição na fila;
  - Se o receptor quer entrar na região crítica, mas ainda não o fez, ele compara o tempo da msg que chegou com o tempo da msg que ele enviou para os outros processos. O menor tempo vence.

Algoritmo Distribuído

- Quando todas as permissões chegam o processo pode entrar na região crítica. Quando ele sai da região crítica, ele envia uma msg OK para todos os processos na sua fila.
- Qual o problema deste algoritmo??

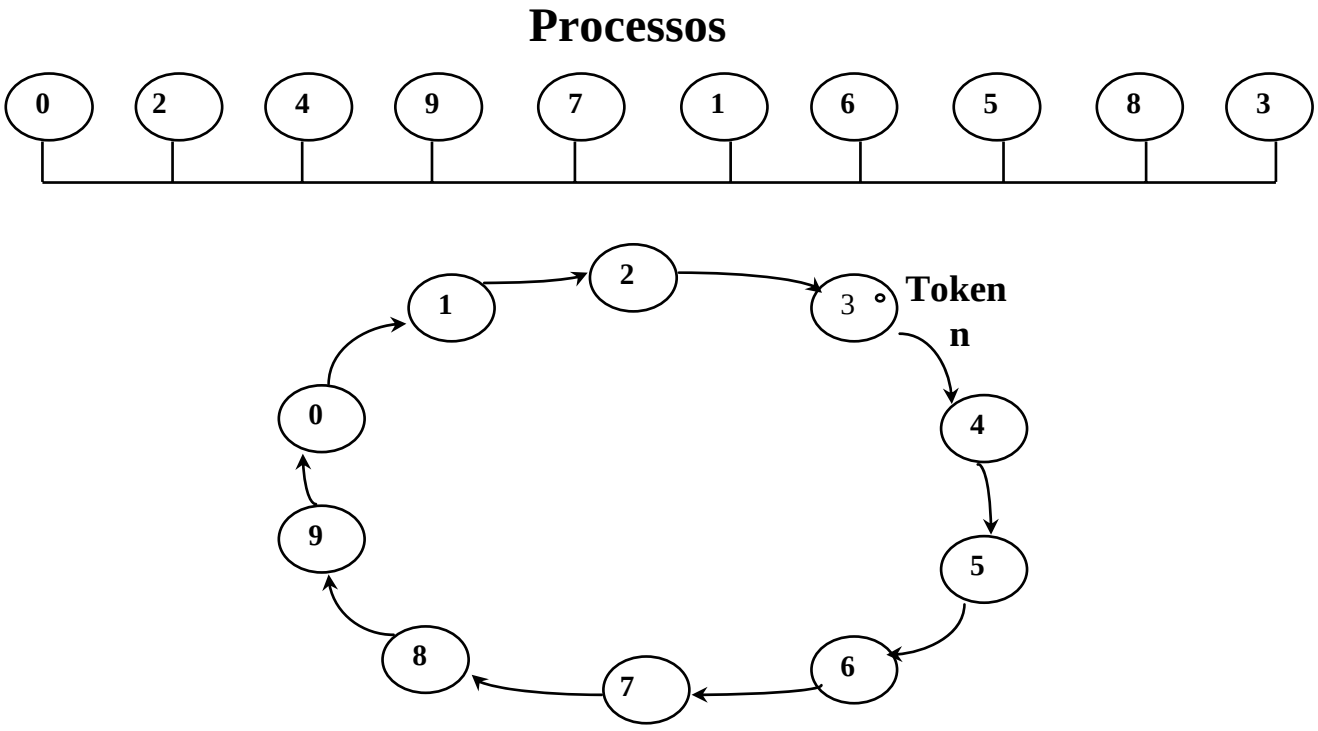


Exclusão Mútua – Algoritmo Token Ring



- É construído um anel lógico por software no qual a cada processo é atribuído uma posição no anel.
- Quando o anel é inicializado, o processo 0 ganha o **token**. O **token** circula no anel (passa do processo **k** para o **k+1**). Quando o processo ganha o **token** ele verifica se ele quer entrar na região crítica. Caso positivo, ele entra na região, realiza o seu trabalho e ao deixar a região passa o **token** para o elemento seguinte do anel.
- Não é permitido entrar em uma segunda região crítica com o mesmo **token**.
- Se o processo não quer entrar na região crítica ele simplesmente passa o **token**. Como conseqüência quando nenhum processo quer entrar na região crítica o **token** fica circulando pelo anel.

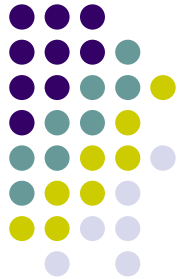
Algoritmo Token Ring



Algoritmo Token Ring

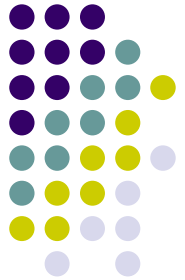
- Problemas:
  - Se o token é perdido ele precisa ser regenerado. A detecção de um token perdido é difícil.
  - Se um processo falhar também ocorrem problemas. A solução é fazer o processo que recebe o token confirmar o recebimento. O processo que falhou pode ser retirado do anel, e o “token” enviado para o processo seguinte. Essa solução requer que todos os processos conheçam a configuração do anel.

Comparação dos Algoritmos



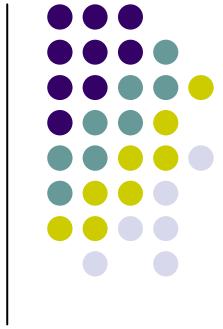
Algoritmo	Mensagens	Atrasos	Problemas
Centralizado	3	2	Falha do Coordenador
Distribuído	2.(n-1)	2.(n-1)	Falha de Qualquer Processo
Token Ring	1 a infinito	0 a n-1	Perda do Token

Eleição do Coordenador



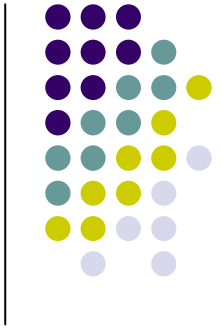
- Muitos algoritmos distribuídos requerem um processo como coordenador.
- Geralmente não importa qual seja o processo coordenador, mas um deles tem que exercer esta função.

Algoritmo do Ditador



- Quando um processo nota que o coordenador não está respondendo a uma requisição, ele inicia uma eleição. A eleição é convocada por um processo P da seguinte forma:
  - **P** envia uma mensagem de ELEIÇÃO para todos os processos com números maiores que o seu;
  - Se nenhum responde, **P** ganha a eleição e se torna coordenador;
  - Se um processo com número maior responder, ele assume a coordenação.

Algoritmo do Ditador



- Quando um processo recebe uma mensagem de ELEIÇÃO de um processo de menor número, ele envia uma mensagem OK de volta indicando que ele vai tomar o comando. Depois disso ele inicia uma eleição.
- Eventualmente todos os processos abandonam a disputa, com exceção de um que é o novo coordenador.
- Ele envia uma mensagem a todos os processos avisando que é o novo coordenador.
- Se um processo que estava “fora do ar” volta, ele inicia uma eleição. Caso ele seja o processo ativo de número mais alto rodando no sistema, ele ganha a eleição e assume a coordenação.