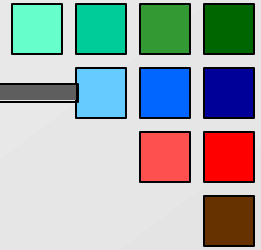
Sistemas Distribuídos

Ricardo Ribeiro dos Santos

ricrs@ec.ucdb.br

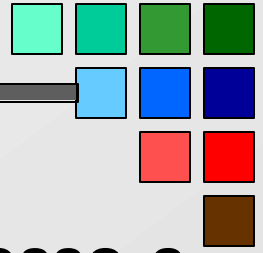
Curso de Engenharia de Computação – UCDB – Setembro/2003

Tópicos



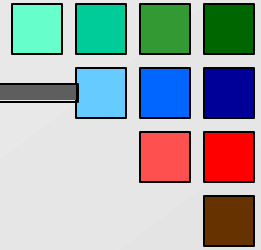
- Sincronização em Sistemas Distribuídos
  - Sincronização de Relógio
  - Estados Globais
  - Algoritmos de Eleição
  - Exclusão Mútua
  - Transações Distribuídas

Sincronização em SD



- Sincronização é importante para controlar o acesso a recursos compartilhados
- Processos precisam concordar na ordem dos eventos
- Muitas soluções de sincronização para sistemas centralizados não podem ser utilizadas em SD
- Vários fatores acabam dificultando o controle de sincronização entre processos de um SD
  - Comunicação não-confiável
  - Separação dos componentes do sistema

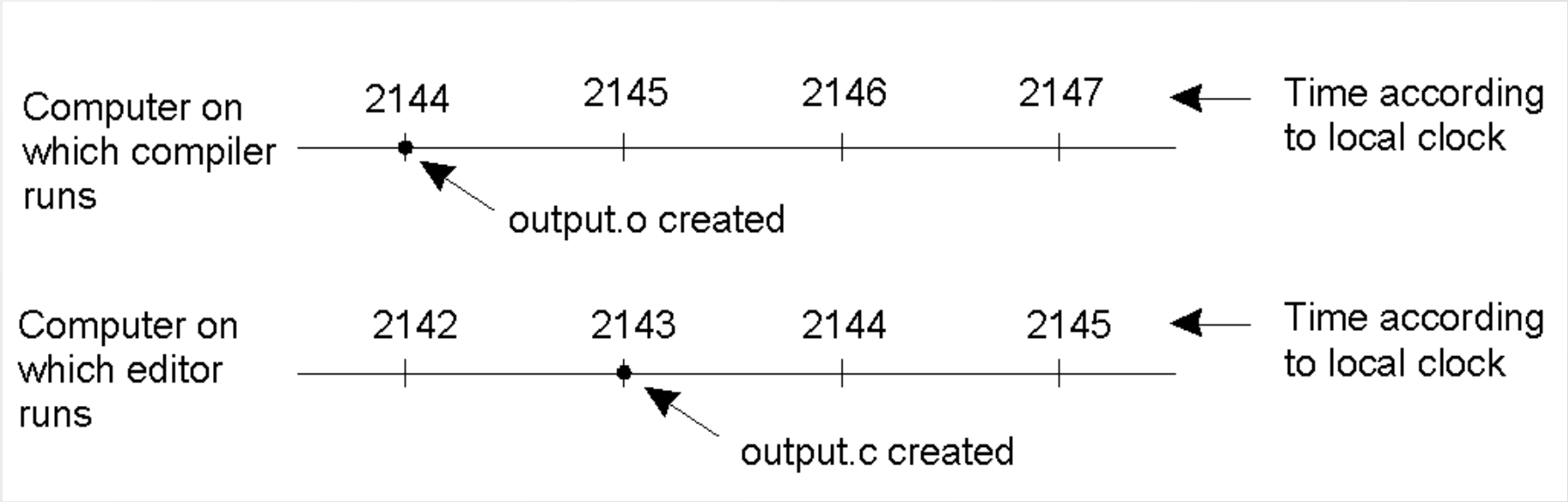
Sincronização de Relógio



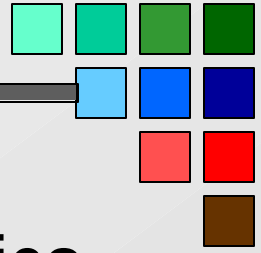
- Aplicações distribuídas possuem as seguintes características:
  - Informação relevante está espalhada em múltiplas máquinas
  - Processos tomam decisões baseadas somente nas informações locais
  - Pontos únicos sujeitos a falhas no sistema devem ser evitados
  - Não há um relógio em comum ou outro tipo preciso de tempo global

Sincronização de Relógio

- Exemplo: falta de tempo global
  - comando make do Unix

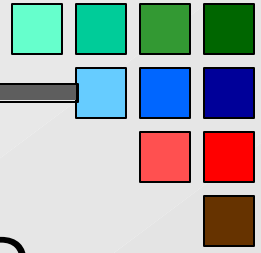


Relógios Lógicos



- Cada computador possui seu próprio relógio físico
- Para diversas aplicações a exatidão desse relógio (em relação ao tempo real) não é importante
  - O importante é que esse relógio ditará a ordem!
- Em sistemas centralizados a consistência interna é mantida pois os tempos relativos das aplicações estão consistentes
- Em SD, relógios em diferentes computadores podem oscilar em diferentes freqüências
  - Dificuldade em manter consistência

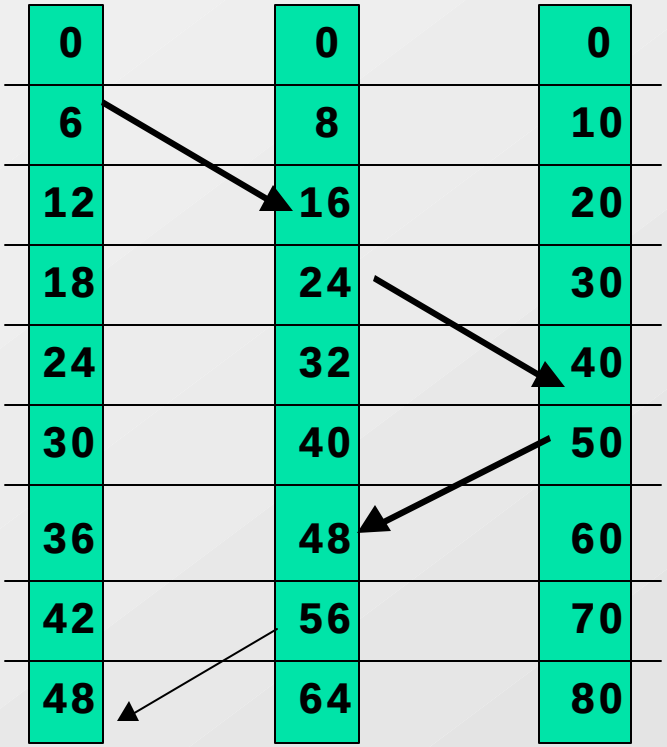
Relógios Lógicos



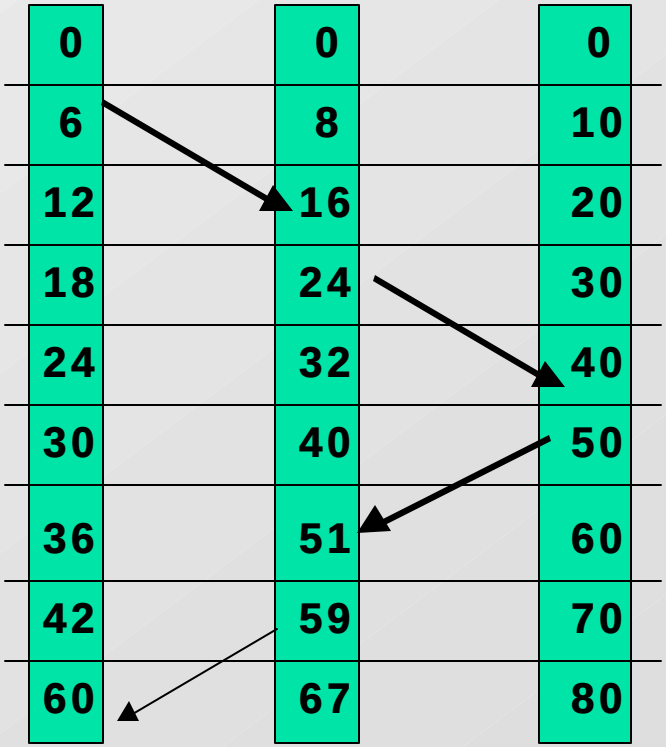
- Algoritmo de Lamport para sincronização em SD
  - Utilização de um tempo único e não-ambíguo
  - Definição de uma relação entre eventos (*happens-before*)
    - $a \rightarrow b$ (a acontece antes de b)
    - Se **a** e **b** são eventos de um mesmo processo **p**, e **a** ocorre antes de **b** ($a \xrightarrow{p} b$), então $a \rightarrow b$
    - Essa relação é transitiva. Se $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$
    - Tempo em que um evento acontece é dado por: C(a)
    - Assim, C(a) < C(b)
    - O tempo do relógio deve sempre andar para frente!

Relógios Lógicos

- Algoritmo de Lamport para sincronização em SD
  - Exemplo:

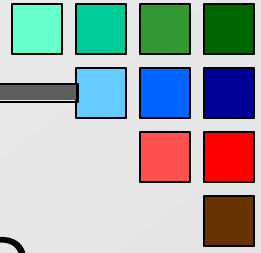


Três processos executando em diferentes taxas de clock



Adoção do algoritmo de Lamport

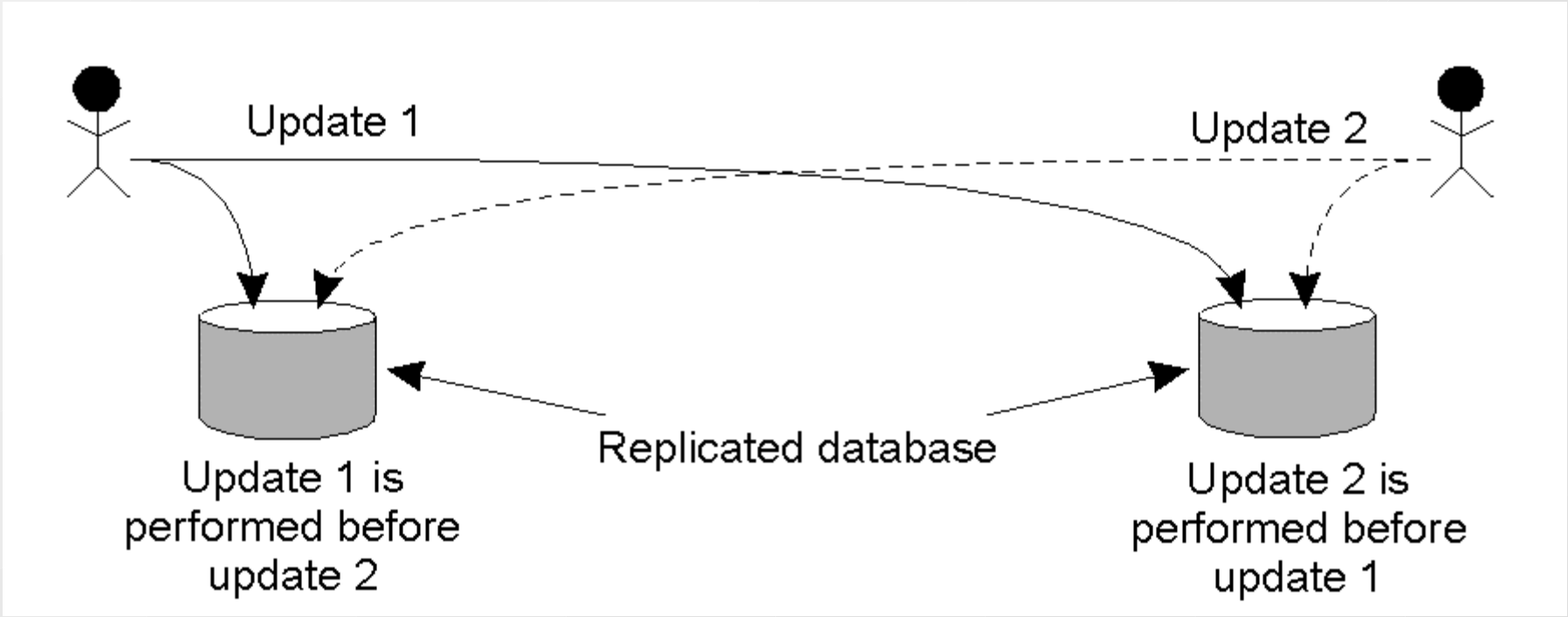
Relógios Lógicos



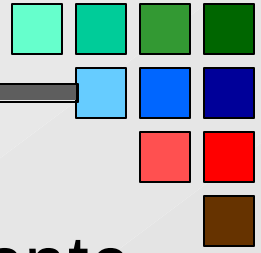
- Algoritmo de Lamport para sincronização em SD
- Solução de Lamport:
  - Uma vez que a mensagem C foi enviada no tempo 60, ela deve chegar no tempo 61 ou maior
  - Cada mensagem carrega o tempo de envio, de acordo com o relógio do transmissor

Relógios Lógicos

- Algoritmo de Lamport para sincronização em SD
  - Aplicação: Multicast totalmente ordenado

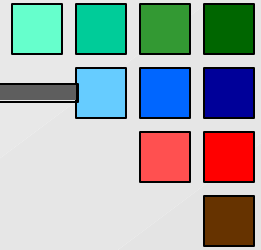


Relógios Físicos



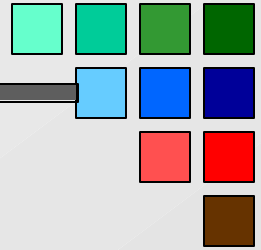
- Para algumas aplicações o tempo real é importante
- Nesse caso, relógios físicos externos são necessários
  - Por razões de eficiência, redundância e tolerância a falhas, múltiplos relógios físicos são utilizados
- Utilização do relógio atômico
  - Oscilações do átomo de césio 133
- UTC (*Universal Coordinated Time*): Tempo preciso
  - NIST (*National Institute of Standard Time*) envia por broadcast um pulso no começo de cada segundo UTC

Relógios Físicos



- Alguns algoritmos foram propostos para tentar minimizar os problemas de sincronização em relógios físicos
  - Algoritmo de Cristian
  - Algoritmo de Berkeley

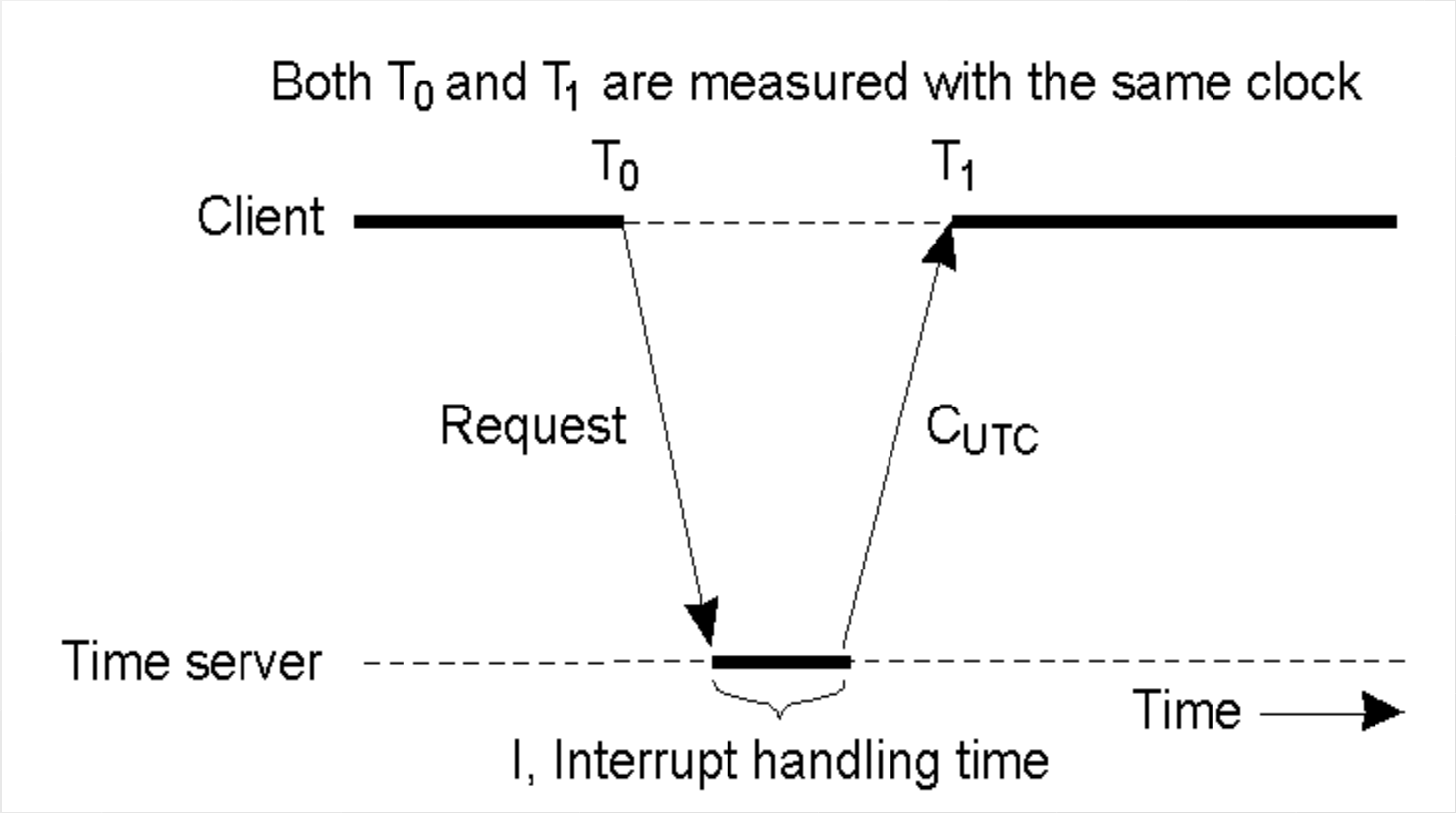
Relógios Físicos



- Algoritmo de Cristian
  - Pressupõe a existência de um servidor de Tempo que possui um servidor WWV
  - As demais máquinas enviam pedidos perguntando a hora corrente
  - Servidor responde com a hora atual *C*
  - Atraso da rede é mensurado
  - Transmissor registra o intervalo de tempo em que fez o pedido (T0) e obteve a resposta (T1)
  - Hora atual pode ser calculado por: C + (T1-T0)/2

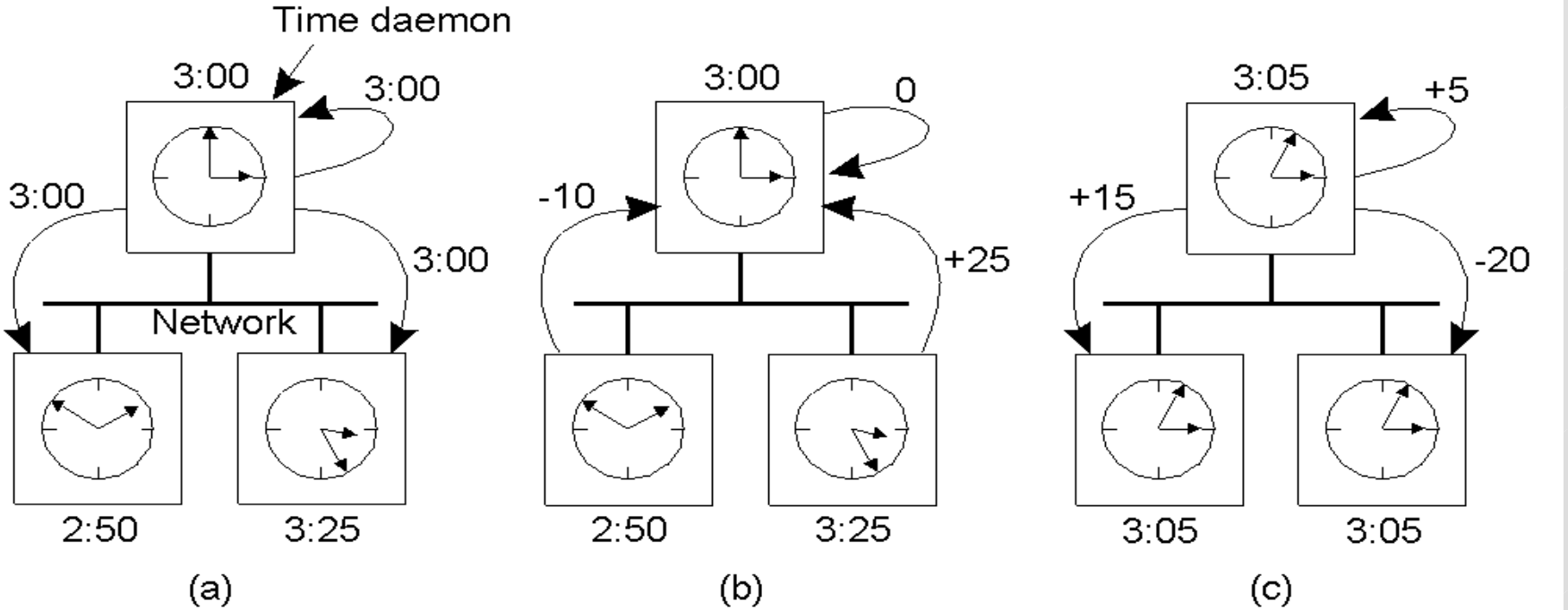
Relógios Físicos

- Algoritmo de Cristian

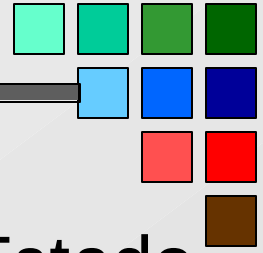


Relógios Físicos

- Algoritmo de Berkeley
  - O servidor pergunta pelo tempo das demais máquinas
  - Servidor calcula o tempo médio
  - “Pede” para outras máquinas adiantarem seus relógios



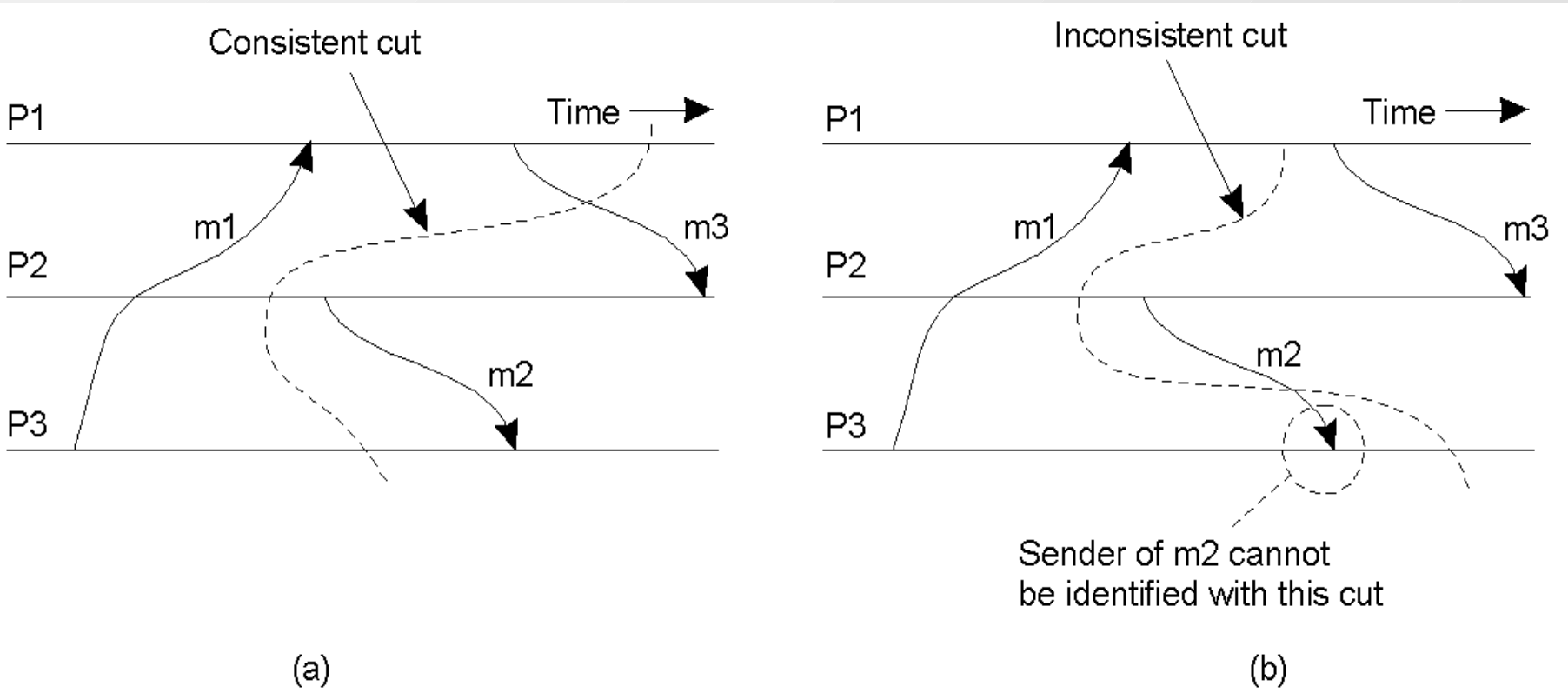
Estado global



- Em muitas situações é importante conhecer o Estado Global do Sistema
  - Por exemplo: Processamento local é terminado e nenhuma mensagem está circulando pela rede, pode-se verificar o estado global para concluir se o sistema está em *deadlock* ou se as tarefas foram corretamente terminadas
- Estado Global consiste do Estado Local dos processos juntamente com as mensagens em trânsito
- Informações de Estado Local dependem do que é importante saber sobre o processo

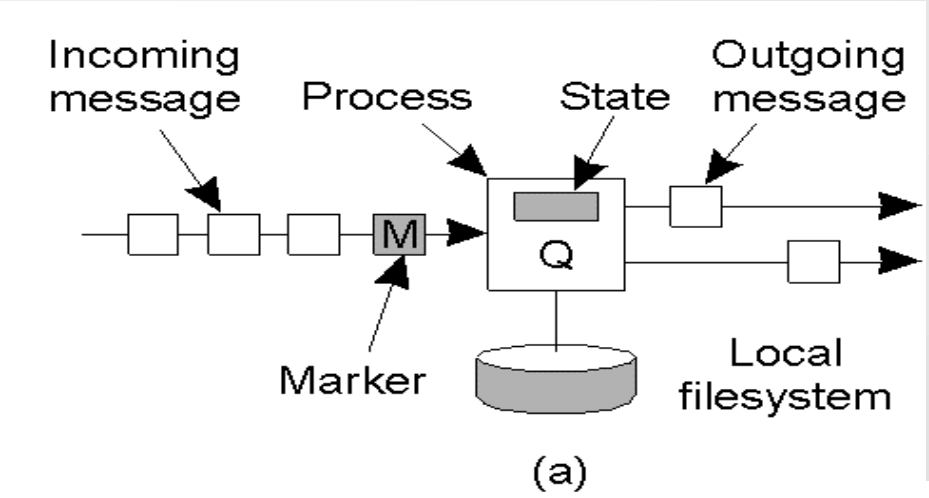
Estado global

- Uma abordagem para alcançar o Estado Global pode ser graficamente representada através de “cortes” no sistema

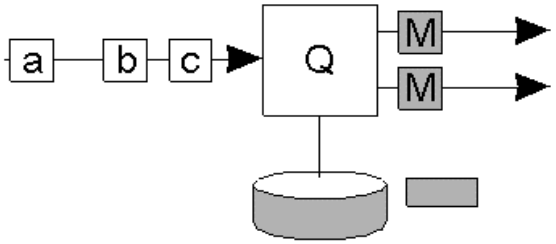


Estado global

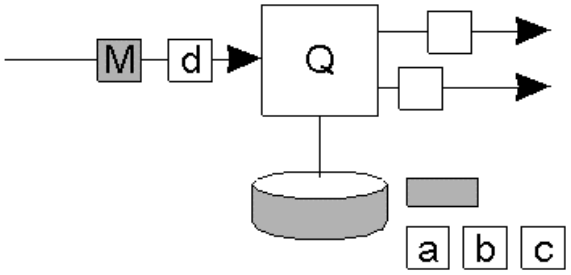
- Exemplo de algoritmo para obter Estado Global
- *Snapshot* Distribuído:



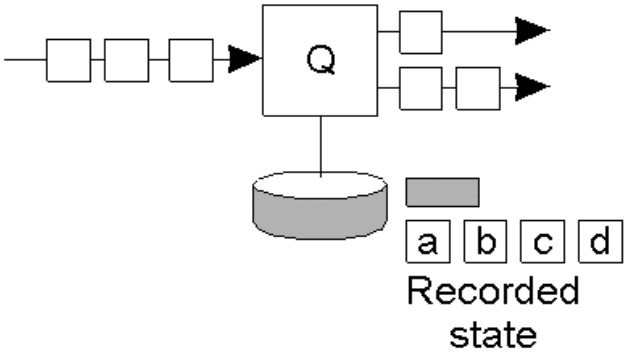
(a)



(b)

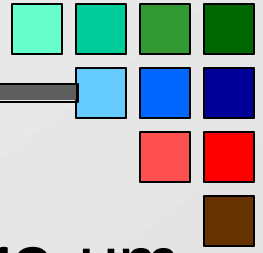


(c)



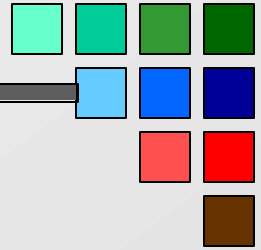
(d)

Algoritmos de Eleição



- Muitas aplicações distribuídas requerem **sempre** um (processo) coordenador
- Com isso, é importante adotar algoritmos que determinam quando um processo será um coordenador
- Essa determinação pode ser feita de diversas maneiras:
  - Processo com maior ID, Número de Rede, Mais antigo, ...
- De forma geral, o objetivo de um algoritmo de eleição é garantir que todos os processos concordam na posse de um novo coordenador

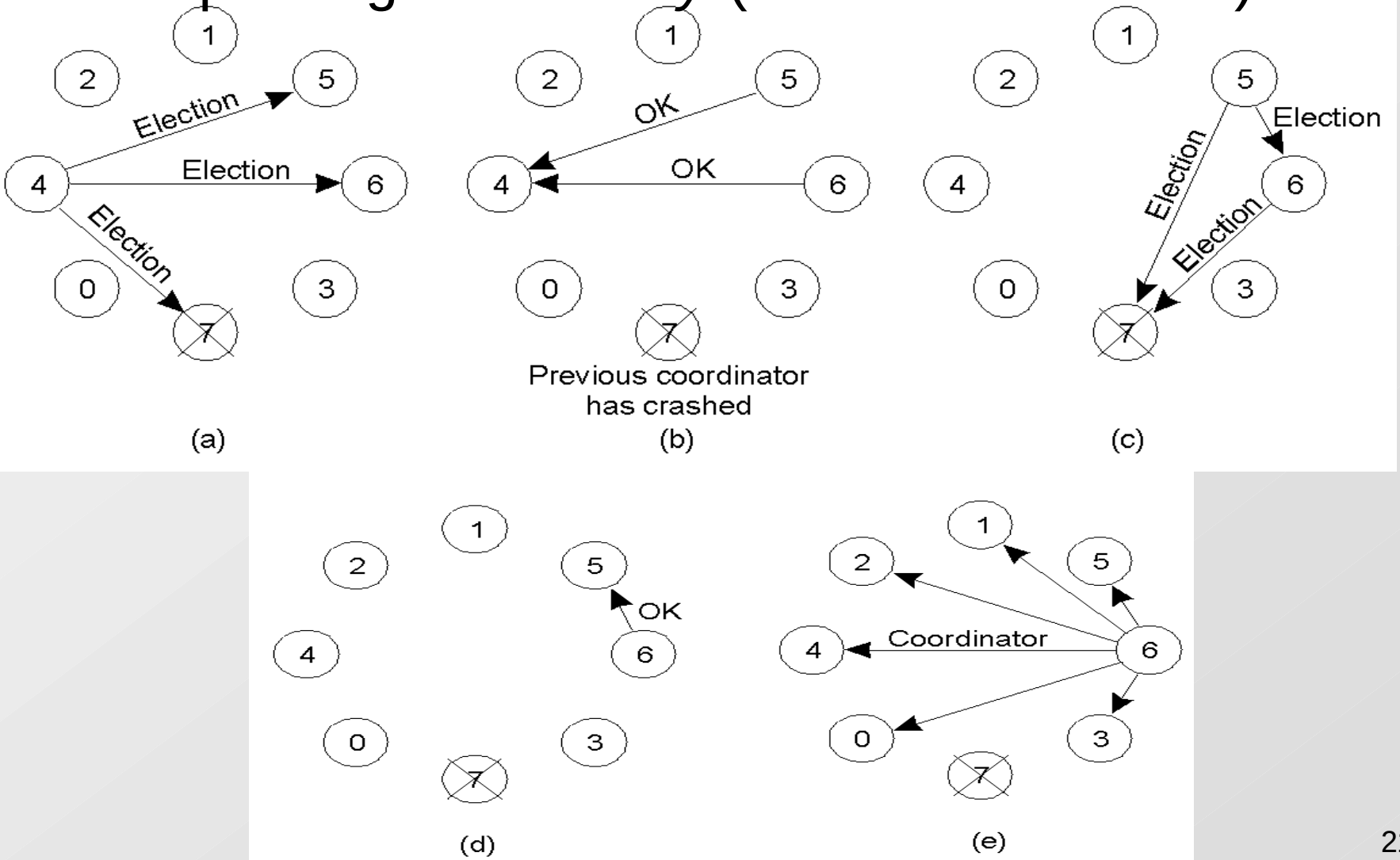
Algoritmos de Eleição



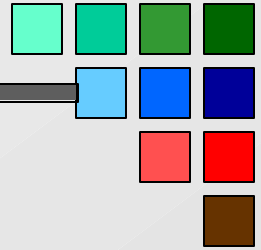
- Exemplo: Algoritmo *bully* (autoritário/valente)
  - Quando um processo nota que o coordenador não responde, uma eleição é iniciada
  - P envia uma mensagem de ELEIÇÃO para todos os processos com número maior que o seu
  - Se ninguém responde, P ganha a eleição e se torna o coordenador
  - Se alguém responde, a tarefa de P está terminada e a eleição pode continuar a partir dos processos com maior número
  - Quando a eleição termina, o ganhador envia uma mensagem para todos os processos informando quem é o novo coordenador

Algoritmos de Eleição

- Exemplo: Algoritmo *bully* (autoritário/valente)



Algoritmos de Eleição



- Exemplo: Algoritmo do anel
  - Processos podem ser fisicamente ou logicamente ordenados
  - Processos sabem que é o seu sucessor
  - Quando um processo nota que o coordenador não responde, constrói uma mensagem de ELEIÇÃO, inclui seu número e envia para seu sucessor
  - A mensagem circula pelo anel e cada processo inclui seu número
  - O iniciador da eleição recebe a mensagem de volta e determina quem é o coordenador
    - Baseado em qual processo possui o maior número
  - Após isso, envia a mensagem COORDENADOR pelo anel

Algoritmos de Eleição

- Exemplo: Algoritmo do anel