Departamento de Sistemas e Computação – FURB
Curso de Ciência da Computação
Disciplina de Sistemas Distribuídos

# Relógios físicos e lógicos

**Prof. Aurélio Hoppe**

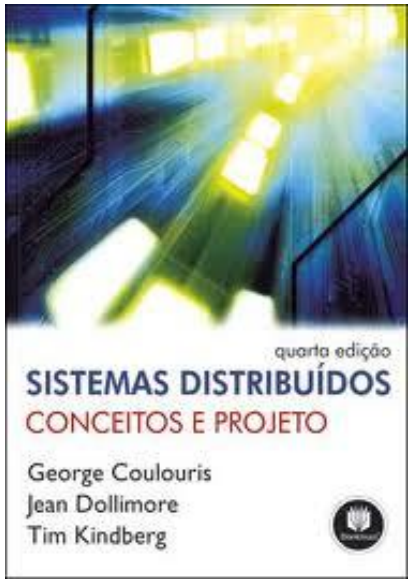
aureliof@furb.br

http://www.inf.furb.br/~aurelio/

Grupo de Processamento de Imagens,
Análise de dados, robótica e
Simulação computacional

# . . . aula de hoje

- Como sincronizar processos
    - Relógios físicos e lógicos



# Tempo

- Tempo é importante em SDs
  - Para etiquetar transações comerciais eletrônica
  - Controle de concorrência com ordenação por timestamp
  - Manter a consistência de dados distribuídos
- Problema:
  - Não existe tempo global em SDs
  - Cada computador tem seu próprio relógio físico
  - Este relógios não são exatos, desviam ao longo do tempo
  - Não conseguimos sincronizá-los perfeitamente

# Tempo

- Veremos
  - Algoritmos para sincronizar relógios físicos de forma aproximativa
  - Relógios lógicos, uma alternativa a sincronização de relógios físicos
  - Algoritmos para determinar o estado global de uma execução distribuída
    - Ex.: determinar o estado de um processo A quando o processo B estava num determinado estado

# Sincronização de relógios Físicos

- O que queremos
  - Ordenar os eventos que ocorrem num SD
  - Para isto, colocamos timestamps = hora + data
- Todo computador num SD tem um relógio interno
  - Utilizados pelos processos para obter o tempo local corrente
  - Processos em computadores diferentes podem colocar timestamps nos eventos
  - Mas os relógios de cada computador fornecem tempos diferentes
- Como sincronizar os relógios dos processos de um SD?
  - Podemos setar todos para o mesmo horário
  - Mesmo assim, os relógios divergem ao longo do tempo a menos que correções sejam aplicadas

# Sincronização de relógios Físicos

- UTC: Tempo universal coordenado
  - Padrão internacional de tempo baseado em relógio atômico mas eventualmente atualizado pelo horário astronômico
  - Difundido por estações de rádio por terra e satélite
  - Computadores com receptores do sinal de rádio podem sincronizar seus relógios
  - Existem servidores de tempo na Internet

- Mais detalhes
  - http://www.bipm.org/en/bipm-services/timescales/time-server.html
  - http://www.worldtimeserver.com/current_time_in_UTC.aspx

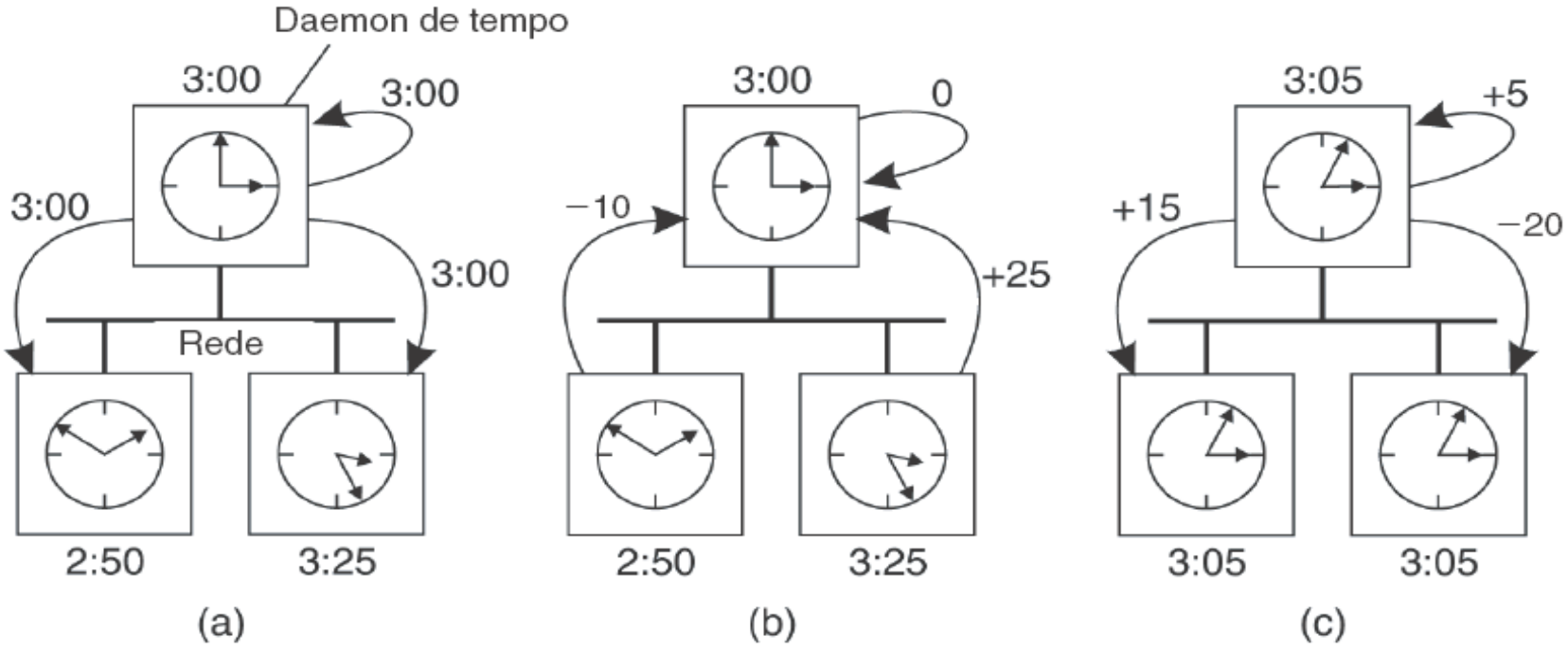
# Relógios Físicos

- Hora atômica Internacional (International Atomic Time) é baseada em relógios muito precisos (taxa de variação de 10 -13).
- Hora Coordenada Internacional (Universal Coordinated Time – UCT) é o padrão para medição do tempo.
- É baseada na hora atômica, mas ocasionalmente é ajustada pela hora astronômia.
- 'Broadcast' da hora é feito através de estações de rádio (WWV) e satélites (GPS).
- Computadores que recebem o sinal sincronizam os relógios.

# Algoritmos de Berkeley

- Algoritmo usado para a sincronização interna de um grupo de computadores.
- 'Servidor de tempo' é ativo (master) e coleta os valores de relógios de outros (slaves).
- Master usa estimativas para estimar o valor dos relógios dos computadores dentro dos grupos.
- Hora atual é resultante de uma média.
- Master envia ao slaves o total de tempo em que os relógios devem adiantar/atrasar.
- Caso o master falhe, um novo computador master é eleito.

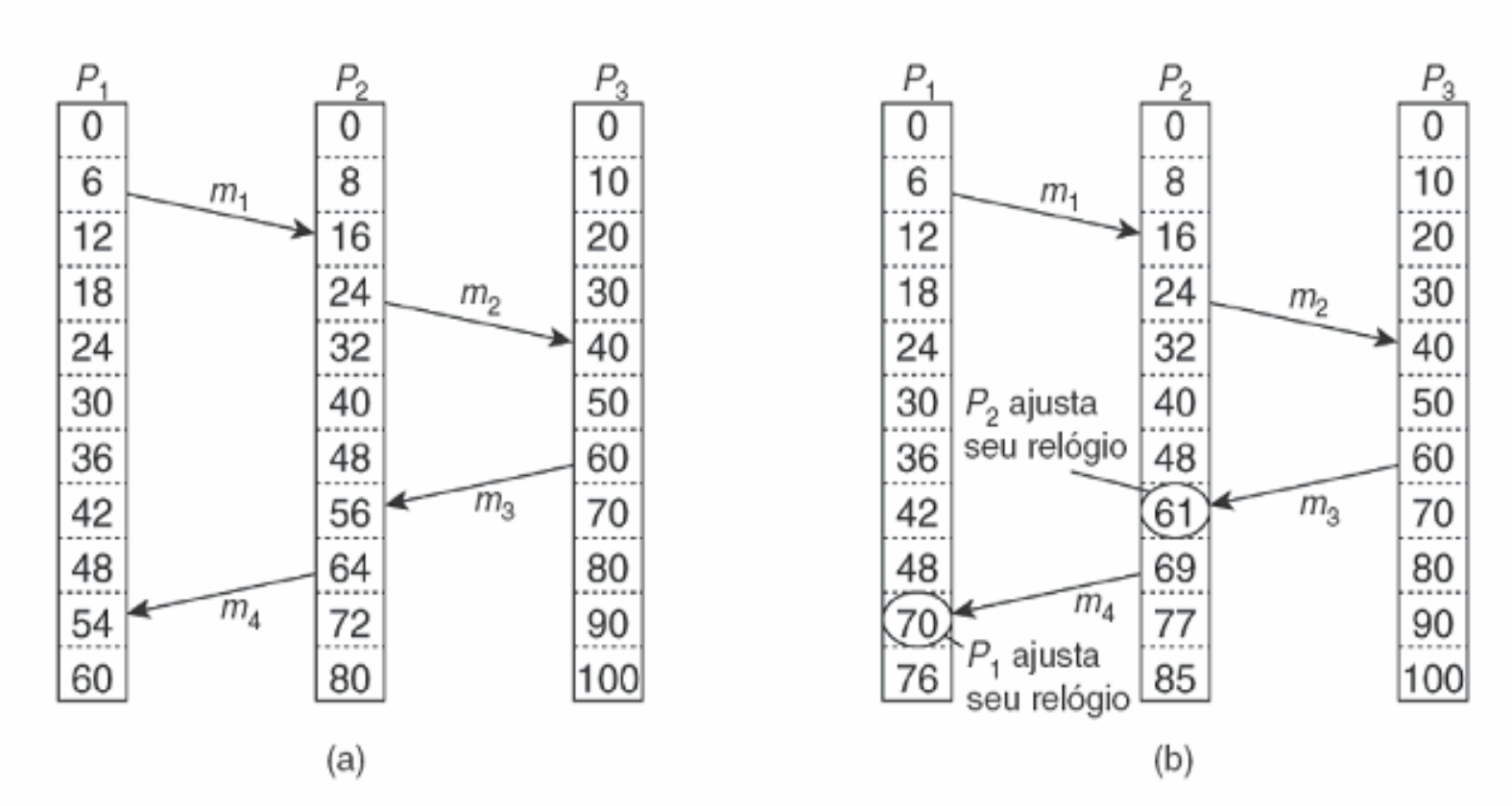
# Algoritmo de Berkeley



# Algoritmo de Lamport

- **A relação "precede" → deve ser observada em duas situações:**
  - Se "a" e "b" são eventos no mesmo processo, e "a" ocorre antes de "b", então "a" → "b" é verdadeira
  - Se "a" é um evento de uma mensagem sendo enviada por um processo, e "b" é o evento relativo a recepção por outro processo, então "a" → "b" também é verdadeira

# Algoritmo de Lamport



# Algoritmo de Lamport

- **Passos para a implantação de relógios lógicos de Lamport:**
  - Antes de processar um evento$_i$ atulizar C
  - $C_i \leftarrow C_i + 1$
  - Quando um processo $P_i$ envia uma mensagem m para $P_j$, ajusta o timestamp de m *ts* (*m*) igual a $C_i$ após executar o passo anterior
  - Ao receber a mensagem m, o processo $P_i$ ajusta seu contador local como
  - $C_i \leftarrow \max \{C_i, ts\ (m)\ \}$, e depois executa o primeiro passo, entregando a mensagem à aplicação

# próxima aula . . .

- Algoritmos de Sockets