Sistemas Distribuídos
Sincronização

Disciplina: Sistemas Distribuídos
Prof.: Edmar Roberto Santana de Rezende

Faculdade de Engenharia de Computação
Centro de Ciências Exatas, Ambientais e de Tecnologias
Pontifícia Universidade Católica de Campinas

Sincronização

Visão geral

- Comunicação entre processos em um sistema distribuído:
  - Protocolos de comunicação
  - Troca de mensagens (incluíndo RPC)
  - Comunicação em grupo
- Como tratar:
  - Regiões críticas
  - Alocação de recursos
- Como prover:
  - Cooperação entre os processos
  - Sincronização entre os processos

Sincronização

Sincronização de clock

- Sincronização em sistemas distribuídos requer uso de algoritmos distribuídos:
  1. Informações relevantes estão dispersas por múltiplas máquinas
  2. Processos devem tomar decisões baseados somente em informações locais
  3. A existência de um ponto único deve ser evitada
  4. Não existe um *clock* em comum ou tempo global preciso
- Indesejável:
  - enviar todas as informações para um único processo gerente que as examina e toma decisões
  - → Muitos nós implicam em uma enorme sobrecarga no gerente

Sincronização

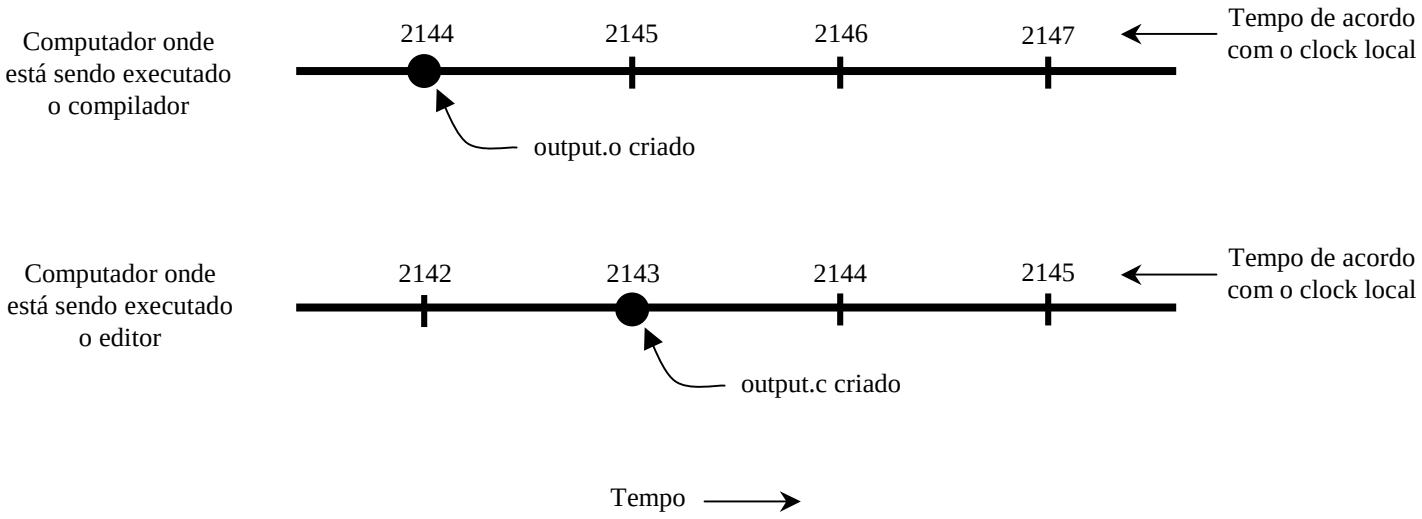
Sincronização de clock

- Indesejável:
  - um único ponto de falha
  - → diminui a confiabilidade no sistema
- Sincronização sem centralização:
  - exige uma abordagem diferente da que é utilizada pelos sistemas operacionais
  - → não há um *clock* em comum
- Em sistemas centralizados:
  - não há inconsistência em relação ao tempo
- É possível sincronizar todos os clocks em um sistema distribuído?

Sincronização

Sincronização de clock

- Exemplo:
  - programa *make*



Sincronização

Clocks lógicos

- Clock:
  - gerado por um cristal de quartzo que oscila em uma frequência bem definida
- Em um sistema centralizado:
  - todos os processos compartilham o mesmo clock
  - relógio adiantado ou atrasado

    → não causa incosistências no funcionamento do sistema
- Em um sistema distribuído:
  - cada computador possui seu próprio clock
  - impossível garantir que os cristais oscilem na mesma frequência

    → causa incosistências no funcionamento do sistema

Sincronização

Clocks lógicos

- Como sincronizar todos os clocks?
  - Paper clássico: Lamport (1978)
    → Algoritmo para sincronização de clocks
  - Extensão: Lamport (1990)

- Lamport:
  - a sincronização de clocks não precisa ser absoluta
  - dois processos que não interagem
    → não necessitam de sincronização de seus clocks
  - processos não precisam concordar com um tempo exato
    → a ordem dos eventos é que é importante
    Ex: make

Sincronização

Clocks lógicos

- Clocks lógicos:
  - em muitos casos:
    - é suficiente que todas as máquinas concordem com o mesmo tempo

      → não é necessário que seja o tempo real
- Clocks físicos:
  - em alguns casos:
    - não é suficiente que todas as máquinas concordem com o mesmo tempo

      → o tempo visto pelo sistema não pode estar distante do tempo real mais do que um certo limite
- Algoritmo de Lamport:
  - sincroniza clocks lógicos

Sincronização

Clocks lógicos

- Algoritmo de Lamport:
  - relação “acontece antes de” (ou “acontecimento-anterioridade”):
    - ***a*** → ***b***
      a “acontece antes de” ***b***
    - todos os processos concordam que primeiro ocorre o evento ***a***, e somente após este evento, ocorre o evento ***b***
    - pode ser observado diretamente em duas situações:
      1. Se ***a*** e ***b*** são eventos em um mesmo processo, e ***a*** ocorre antes de ***b***, então ***a*** → ***b*** é verdadeiro
      2. Se ***a*** corresponde ao envio de uma mensagem por um processo e ***b*** corresponde ao recebimento desta mensagem por outro processo, então ***a*** → ***b*** é verdadeiro
    - esta relação é transitiva:
      - Se ***a*** → ***b*** e ***b*** → ***c***, então ***a*** → ***c***

Sincronização

Clocks lógicos

- Algoritmo de Lamport:
  - relação “acontece antes de” (ou “acontecimento-anterioridade”):
    - se dois eventos ***x*** e ***y***
      a) acontecerem em processos diferentes, **e**
      b) os processos não trocam mensagens (mesmo indiretamente)
      então nem ***x*** → ***y*** nem ***y*** → ***x*** são verdadeiros

    → tais eventos são considerados **concorrentes**
    - significa que:
      - nada pode ser dito (ou nada precisa ser dito) a respeito de quando tais eventos ocorrem, ou qual ocorre antes ou depois

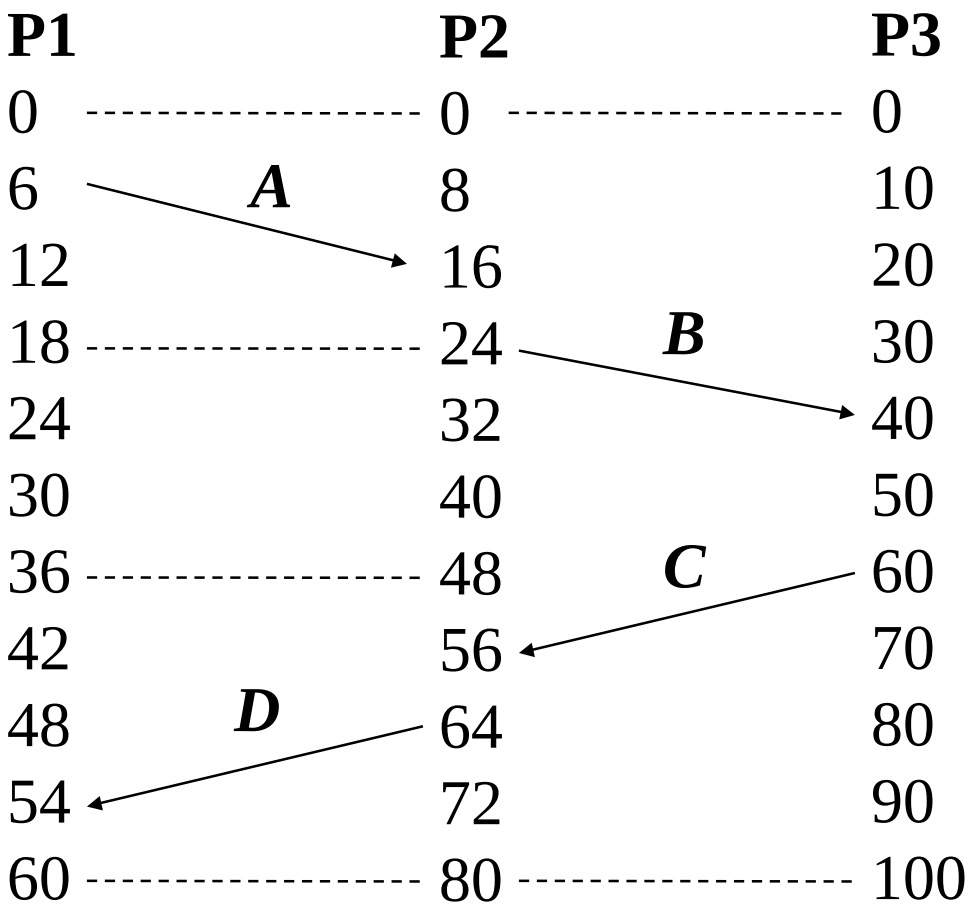
Sincronização

Clocks lógicos

- Algoritmo de Lamport:
  - medida de tempo:
    - para cada evento ***a*** podemos atribuir um valor ***C(a)***
    - → ***C(a)*** corresponde a um instante de tempo com o qual todos os processos concordem
    - Se ***a*** → ***b***, então ***C(a)*** < ***C(b)***
    - logo, podemos redefinir as condições anteriores como:
      1. Se ***a*** e ***b*** são eventos em um mesmo processo, e ***a*** ocorre antes de ***b***, então ***C(a)*** < ***C(b)***
      2. Se ***a*** corresponde ao envio de uma mensagem por um processo e ***b*** corresponde ao recebimento desta mensagem por outro processo, então ***C(a) < C(b)***
    - ***C*** deve ser sempre crescente, nunca decrescente
    - correções nos clocks podem ser feitas adicionando-se um valor positivo ao clock, nunca subtraíndo

Sincronização

Clocks lógicos



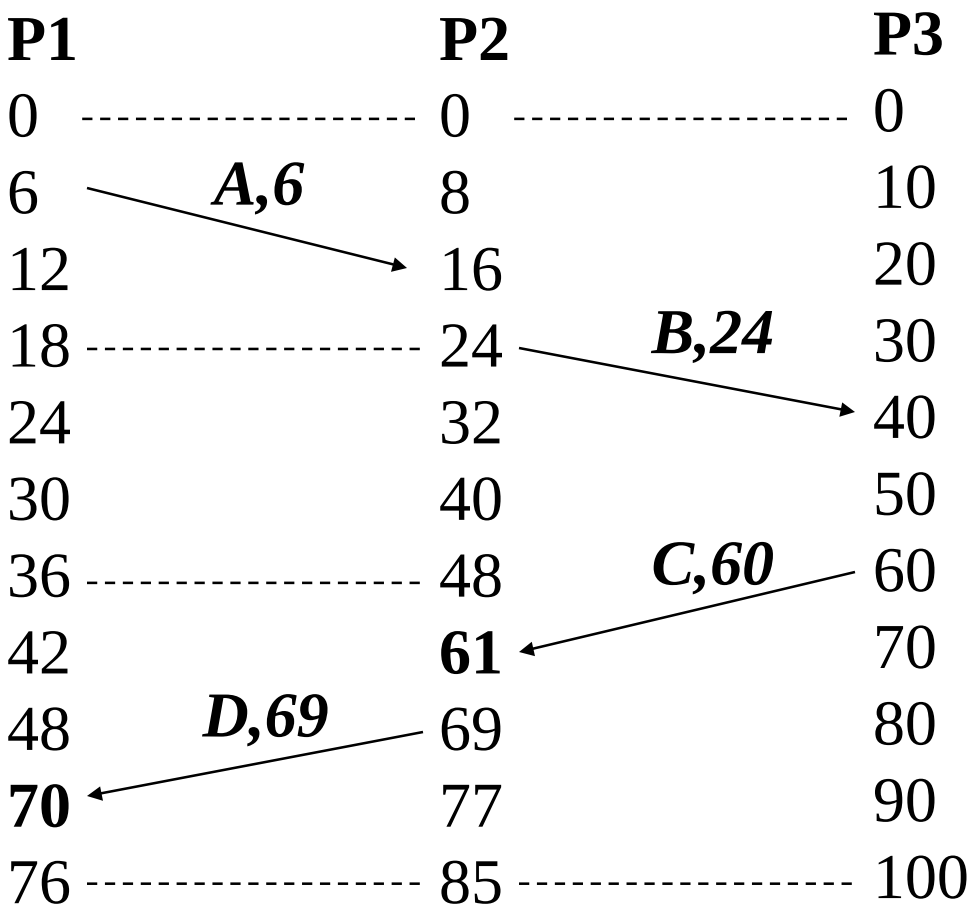
Sincronização

Clocks lógicos

- Algoritmo de Lamport:
  - segue diretamente a relação “acontece antes de”
  - se C deixa sua máquina no tempo 60
    - → deve chegar ao seu destino no tempo 61 ou mais tarde
    - cada mensagem leva consigo:
      - instante de tempo da transmissão
      - → de acordo com o clock do emissor
    - quando a mensagem chega:
      - se o tempo do receptor é anterior ao da mensagem
      - → receptor adianta seu clock (clock da mensagem + 1)

Sincronização

Clocks lógicos



Sincronização

Clocks lógicos

- Algoritmo de Lamport:
  - com uma pequena modificação, implementa tempo global:
    - entre dois eventos o clock precisa rodar pelo menos uma vez

    Ex: Se um processo envia ou recebe uma mensagem em rápida sucessão, então ele precisa avançar seu clock de no mínimo uma unidade entre os dois eventos

  - em algumas situações, mais uma característica é necessária:
    - dois eventos não podem nunca ocorrer exatamente no mesmo instante de tempo
    - Solução:
      - pode-se acrescentar o número do processo aos bits de mais baixa ordem do tempo, separados por ponto decimal

      Ex: Se dois eventos acontecerem nos processos 1 e 2 no tempo 40, o tempo do primeiro passa a ser 40,1 e do segundo 40,2

Sincronização

Clocks lógicos

- Algoritmo de Lamport:
  - podemos redefinir a condições anteriores:
    1. Se ***a*** ocorrer antes de ***b*** no mesmo processo, então ***C(a)*** < ***C(b)***
    2. Se ***a*** e ***b*** são o envio e o recebimento de uma mensagem, então ***C(a) < C(b)***
    3. Para quaisquer eventos ***a*** e ***b***, ***C(a) ≠ C(b)***

  - este algoritmo oferece uma forma de ordenação total de todos os eventos no sistema

  → muitos outros algoritmos precisam deste tipo de ordenação para evitar ambiguidades