# Sistemas Distribuídos
# Aula 9: Ordenação de Eventos & Relógios Lógicos

**Prof. Elias P. Duarte Jr.**
**Universidade Federal do Paraná (UFPR)**
**Departamento de Informática**
**www.inf.ufpr.br/elias/sisdis**

# Sumário

- Relembrando nossa definição de Sistema Distribuído
- Eventos que acontecem antes (ou depois) de outros eventos
- A relação matemática “aconteceu-antes-de” (“*happened before*”)
- Ordenação parcial de eventos
- Ordenação total de eventos: Relógios Lógicos

# Relembrando Sistemas Distribuídos

- Em nossa definição primeira: um Sistema Distribuído é um conjunto de processos (i.e. programas em execução) que cooperam para a realização de alguma tarefa
- Nossos sistemas distribuídos são estáticos (constituídos de N processos)
- Os processos estão *fully-connected*: qualquer processo pode comunicar diretamente com qualquer outro sem passar por intermediários
- Os processos se comunicam trocando mensagens: *send*(msg), *receive*(msg), *deliver*(msg)

# Eventos e sua Ordem

- Muitas vezes é importante dizer se um evento aconteceu antes de outro em um Sistema Distribuído
- O que é um “evento”? Definição muito flexível!
- Para nós na disciplina: evento = execução de instrução
- Um processo é um programa em execução, portanto executa uma sequência de instruções
- Importante: não há um relógio global
- Cada processo tem seu relógio local

# “Aconteceu Antes De”: Tempo

- Os eventos acontecem ao longo do tempo físico
- Para dizer que um evento *e* aconteceu antes de outro evento *f:*
  - suas ocorrências tem que ser medidas usando o sistema de tempo comum, físico
- Isso só é possível (a princípio ;-) usando o mesmo relógio
- Mas não há relógio compartilhado!

# Ordenação Parcial de Eventos

- Relembrando: um evento é a execução de uma instrução
- Portanto: um processo consiste de uma sequência de eventos
- Considerando 1 único processo, é fácil determinar qual evento aconteceu antes de qual outro
- Basta usar o relógio local: um relógio comum a todos os eventos daquele processo

# A Relação "happened-before"

- Vamos definir uma relação matemática entre eventos: aconteceu-antes-de ou *happened-before*
- Se o evento a aconteceu antes do evento b, representamos assim:

$$a \rightarrow b$$

# Estendendo para Múltiplos Processos

- Como estender a relação aconteceu-antes-de para múltiplos processos?
- Não trivial: pois não temos relógio comum aos processos que permita medir (trivial em 1 único processo)
- Mas o que podemos usar dos Sistemas Distribuídos para ajudar nesta tarefa?

# A Transmissão de Mensagens

- Se uma mensagem foi recebida por um processo y, então, obrigatoriamente, ANTES ela foi transmitida por um processo x

**send(msg) acontece antes de receive(msg)**

- Uma palavra/definição importante (importantíssima!) que surge aqui é a “causalidade”
- A transmissão de uma mensagem pela origem ***causa*** o recebimento daquela mensagem no destino

# A Relação → Para Múltiplos Processos

- Definição: A relação aconteceu-antes-de (→) definida sobre um conjunto de eventos de um sistema distribuído que consiste de múltiplos processos é a menor relação que satisfaz as seguintes condições:

  **(1)** se a & b são eventos de um mesmo processo e usando o relógio local foi medido que a aconteceu antes de b, então: a → b

# A Relação → Para Múltiplos Processos

**(2)** Se o evento a corresponde ao envio de uma mensagem, e o evento b corresponde ao recebimento daquela mensagem, então: a → b

**(3)** Se a → b & b → c

Então a → c

- Veja que há eventos em um sistema distribuído que nem a → b nem b → a,  são ditos concorrentes

# Ordem Parcial

- A relação aconteceu-antes-de define uma ordem parcial dos eventos de um sistema distribuído
- Por que é “parcial”? Porque deixa eventos concorrentes
- Veja o seguinte exemplo de instruções executadas por 3 processos (A, B e C), o que podemos dizer?

Processo A: inst A1; send B; inst A2; recv C;

Processo B: inst B1; recb A; inst B2; send C;

Processo C: inst C1; send A; inst C2; recb B;

# Relogios Lógicos

- Podem sem pensados como uma forma de enumerar os eventos de um Sistema Distribuído
- Não são relógios de tempo físico!
- Vamos assinalar números sequenciais para eventos: *timestamps*

# Relógio Lógico de 1 Processo

- Considere um processo $P_i$
- O relógio lógico $C_i$ deste processo é uma função que assinala um valor $C_i(a)$ para um evento a de $P_i$

# Relógio Lógico do Sistema Distribuído

- A função C (sem índice) representa o relógio lógico do Sistema Distribuído, para todos os processos
- Esta função C só é correta se for consistente com a relação →
- Em outras palavras, deve satisfazer a condição de relógio (*clock condition*):

  Para quaisquer eventos a & b, se a → b

  então C(a) < C(b)
- Note que a condição reversa não é garantida!

# Implementando Um Relógio Lógico

- Para implementar um relógio lógico basta fazer o seguinte:

1) Cada mensagem msg transmitida carrega o *timestamp* do seu envio send(msg)

2) Um processo Pi deve incrementar Ci entre dois eventos consecutivos

3) Um processo Pj que recebe a mensagem msg com timestamp Tm seta Cj maior que o seu valor presente **e** maior que Tm

# Ordenação Total de Eventos

- Até este momento: há eventos que não foram ordenados, estes eventos (de diferentes processos) têm o mesmo *timestamp*
- Pergunta: Como ordenar estes eventos com o mesmo *timestamp*??

# Ordenação Total de Eventos

- Até este momento: há eventos que não foram ordenados, estes eventos (de diferentes processos) têm o mesmo *timestamp*
- Pergunta: Como ordenar estes eventos com o mesmo *timestamp*??
- Resposta: Use qualquer estratégia :-)
- Por exemplo, ordene pelos identificadores dos processos :-0

# A Ordem Total

- Para dois eventos a & b, dizemos que a => b se e somente se:

  (i) Ci(a) < Ci(b) ou

  (ii) Ci(a) == Ci(b), mas o evento a é de Pi e o evento b de Pj, com i<j

- A relação => é consistente com →

  se a → b então a => b

# Questionando a Ordenação...

- Ordenar os eventos concorrentes pelos ids dos seus processos pode não corresponder à realidade, no tempo físico!
- Em geral: tudo bem! O que queremos é que todos os processos tenham uma mesma visão comum sobre a ordem total de todos os eventos
- É extremamente útil: facilita a coordenação dos processos

# Exercício 1: Relógios Lógicos

- Ordene todos os eventos do sistema distribuído abaixo usando relógios lógicos:

  Processo A: inst A1; send C; recv B; inst A2; send D; recv D; inst A3;

  Processo B: send A; inst B1; recv C; inst B2;  recv D;

  Processo C: inst C1; inst C2; recb A; inst C3; send B;

  Processo D: inst D1; recv A; send A; inst D2; send B

# Exercício 2: Relógios Lógicos

- Ordene todos os eventos do sistema distribuído abaixo usando relógios lógicos:

  Processo A: send C; inst A1; recv D; inst A2; send B;

  Processo B: send D; inst B1; recv A; inst B2;

  Processo C: inst C1; recv A; inst C2;

  Processo D: recv B; inst D1; send A; inst D2;

# Referência original dos Relógios Lógicos:

Leslie Lamport, “Time, Clocks, and the Ordering of Events in a Distributed System,” *Communincations of the ACM*, Vol. 21, No. 7, pp. 558-565, 1978.

*Descritos também em virtualmente todos os livros de Sistemas Distribuídos.*

# Conclusão

- Nesta aula estudamos como derterminar o que aconteceu antes do que em um sistema distribuído
- Vimos que o envio e recebimento de mensagens são a base para definir qual evento “aconteceu-antes-de” outro
- A relação matemática entre eventos “aconteceu-antes-de” é a base da ordenação parcial de eventos
- Mostramos como realizar uma ordenação total de eventos
- Fizemos exercícios de ordenação total de eventos

**Obrigado!**
**Página da Disciplina**
**Sistemas Distribuídos:**
**www.inf.ufpr.br/elias/sisdis**