Sistemas Distribuídos

Sincronização de Relógios

Edeyson Andrade Gomes
www.edeyson.com.br

Roteiro da Aula

- Definições
- Clock Físico
  - Cristian
  - Berkeley
- Clock Lógico
  - Lamport

Definições

Clock Físico

- Dia Solar
  - Intervalo entre dois "ápices" consecutivos do sol
  - Segundo Solar
    - 1/24*3600 = 1/86400 do dia solar

Clock Físico

- Período de rotação da Terra não é constante
  - Dias ficando mais longos
  - 300 milhões anos atrás
    - Ano terrestre = 400 dias
      - Diminuição da velocidade de rotação devido à fricção do mar e atmosférica
      - Dias mais longos
  - Relógio Atômico
    - Césio 133
      - Precisão

Clock Físico

- Relógio Atômico
  - Segundo:
    - Tempo de 9.192.631.770 transições do Césio 133
    - Igualar segundo solar no ano de criação
    - TAI – Tempo Atômico Internacional
      - Média de todos os ticks de todos os relógios atômicos
      - TAI = Média de Ticks/9.192.631.770

Tempo

- 50 relógios atômicos no mundo
- Tempo Atômico Internacional (TAI)
  - Média dos pulsos dos relógios atômicos desde a meia noite do dia 1 de Janeiro de 1958, dividido por 9.192.631.770
- 86400 segundos do TAI representa hoje 3 milisegundos a menos que o dia solar médio
  - Dias mais longos

UTC

- *leap seconds (pulos nos segundos)*
  - Ajuste do TAI sempre que o tempo solar ficar 800 milisegundos maior
  - Coordenado pelo Bureau International de l´Heure (BIH),

- Sistema de tempo baseado no TAI
  - Permanece em fase com o aparente movimento do sol
  - Universal Time Coordinated
  - Substituiu GMT (Greenwich Mean Time)

Sincronização de Relógio

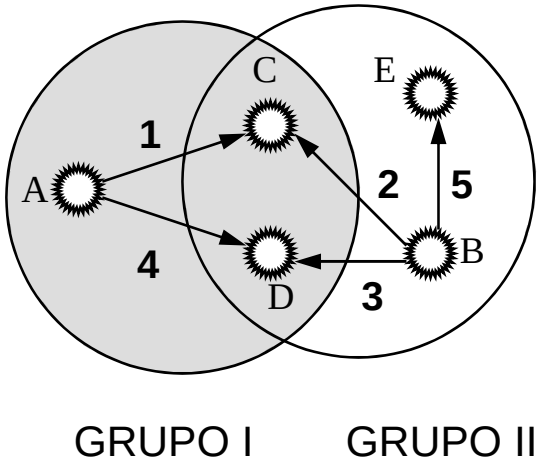
- Ambigüidade de Tempo

Processo A envia msg1
msg1 = dobre o valor de x

Processo B envia a msg2
msg2 = some 100 a x

X tem valor 1000

Processo C executa msg1 e msg2
x = 2100
Processo D executa msg2, msg1
x = 2200

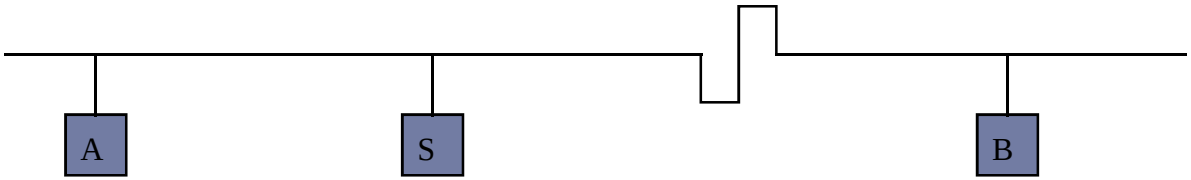


Sincronização de Relógio

- Ambigüidade de Tempo
  - Inexistente entre processos em Sistemas Centralizados
    - Informação direcionada pelo Kernel
    - Se P1 pergunta pelo tempo antes de P2
      - TP1 <= TP2
  - Solução não trivial em Sistemas Distribuídos

Sincronização de Relógio

- Processo B questiona a hora a S no instante T0
- Processo A questiona a hora a S no instante T0 + 1
  - Tempo de B **deveria** ser <= ao tempo de A



  - O request de A chega primeiro a S
    - TA <= TB

Sincronização de Relógio

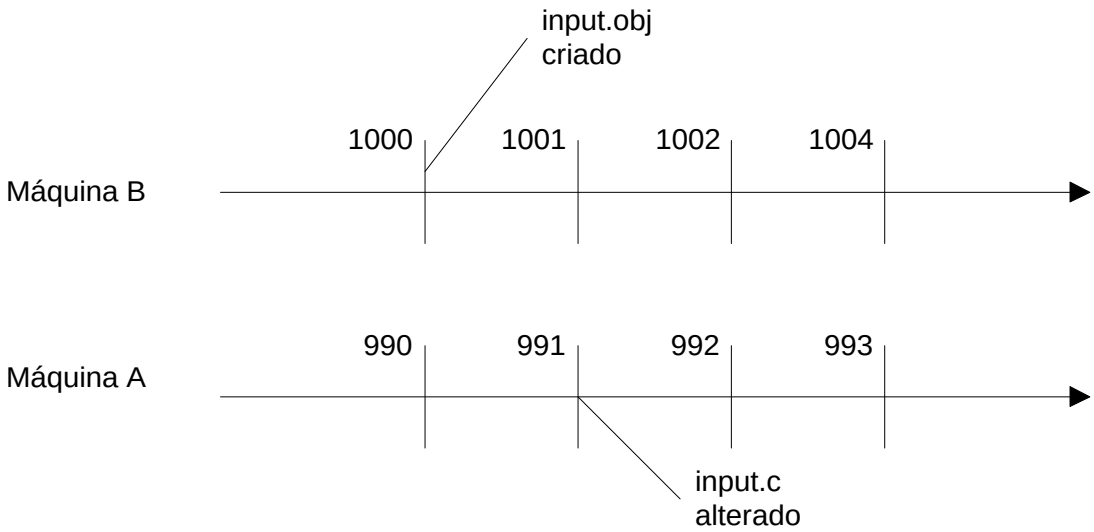
- Exemplo de ambigüidade do Tempo Global
  - Programa *make* do UNIX ou MS VC++
    - Examina arquivos fonte e objeto
    - Compara data e hora de alteração
    - Se o tempo do fonte é maior que o do objeto, cria um novo objeto

Sincronização de Relógio

- Exemplo de ambigüidade do Tempo Global
  - Programa *make* do UNIX ou MS VC++
    - Se input.c tem tempo 1000 e input.obj tem tempo 900, cria um novo input.obj
    - Se input.c tem tempo 1000 e input.obj tem tempo 1010, NÃO cria um novo input.obj
      - Nesse caso o executável não reflete as alterações feitas

Sincronização de Relógio

- Exemplo de ambigüidade do Tempo Global
  - Supondo um ambiente distribuído com a edição na máquina A e a compilação na máquina B
    - O que acontece se o relógio de B adiantar em função de A?



Sincronização de Relógio

Clock X Timer

Clock x Timer

- Timer
  - Cristal de Quartzo
    - Oscilar em freqüência bem definida
      - Depende do tipo do cristal, corte e tensão
        - 50 Hz ou 60 Hz
    - *Leap Second*
      - *Ajuste para 51 Hz ou 61 Hz*

Clock x Timer

- Timer
  - Registradores associados
    - Counter (Contador)
    - Holding Register (Armazenamento)

Clock x Timer

- Counter (Contador)
  - Decrementado por cada oscilação do cristal
  - Quando Counter = 0
    - Interrupção
    - Recarga do Counter pelo valor do Holding Register
  - Interrupções
    - Clock Tick
      - 60 vezes por segundo, por exemplo

Clock x Timer

- Hardware
  - Armazena Data e Hora padrão
  - Ajuste manual pelo usuário
    - Armazenado como deslocamento da data e hora padrão em número de *ticks*
  - Clock tick
    - Adiciona 1 ao tempo da memória
      - Atualiza relógio

Clock x Timer

- Multicomputadores
  - Múltiplos Clocks
    - Freqüências distintas
  - Perda de sincronização dos relógios
    - Diferença
      - Clock Skew

Clock x Timer

- Acesso ao TAI
  - National Institute of Standard Time (NIST)
    - WWV em Fort Collins
      - Precisão de ±1 milisegundo a ±10 milisegundos
  - GPS "civil" com um erro de 1 ms
  - GPS "militar" com um erro de 1 ms

Clock x Timer

- **Os relógios dos computadores não estão sincronizados**

  ***skew* -** diferença entre dois relógios

- **Relógios dos computadores estão sujeitos a desvio**

  ***drift* - c**ontam o tempo a ritmos distintos

- **Ritmo de desvio**

  ***drift rate* -** Diferença, por unidade de tempo, em relação a um relógio de referência

Clock x Timer

- **Relógios de quartzo correntes**
  - *Drift* de 1 seg em 11-12 dias (10^{-6} segs/seg)
- **Relógios de quartzo de alta precisão**
  - *Drift rate* entre 10^{-7} e 10^{-8} segs/seg
- **TAI**
  - *Drift rate* entre 10^{-13} segs/seg

Clock Lógico

Solução de Lamport

Solução de Lamport

- Lamport
  - Sincronização de Clock não necessita ser absoluta
- Processos sem interação
  - Perda de sincronização não interfere

Solução de Lamport

- Tempo real X ordem dos eventos
  - O que interessa aos processos?
  - Alguns processos:
    - Ordem dos eventos

    ➔ Clock Lógico

Clock Lógico X Físico

- Clock Físico
  - Clocks devem ser **iguais** (multicomputadores) e não podem derivar num certo valor do tempo real.
    - Tempo Real

Solução de Lamport

- Relação "happens-before"
  - $a \rightarrow b$ é lida "a acontece antes de b"
  - Todos os processos concordam que o evento a ocorreu antes de b
    - Se a e b são eventos num mesmo processo e "a acontece antes de b", então $a \rightarrow b$ é verdadeira
    - Se a representa um SEND de P1 e b um RECV de P2, sendo P_1 e P_2 processos distintos, então $a \rightarrow b$ é verdadeira.

Solução de Lamport

- Relação "happens-before"
  - Transitividade
    - $a \rightarrow b$ e $b \rightarrow c$, então $a \rightarrow c$

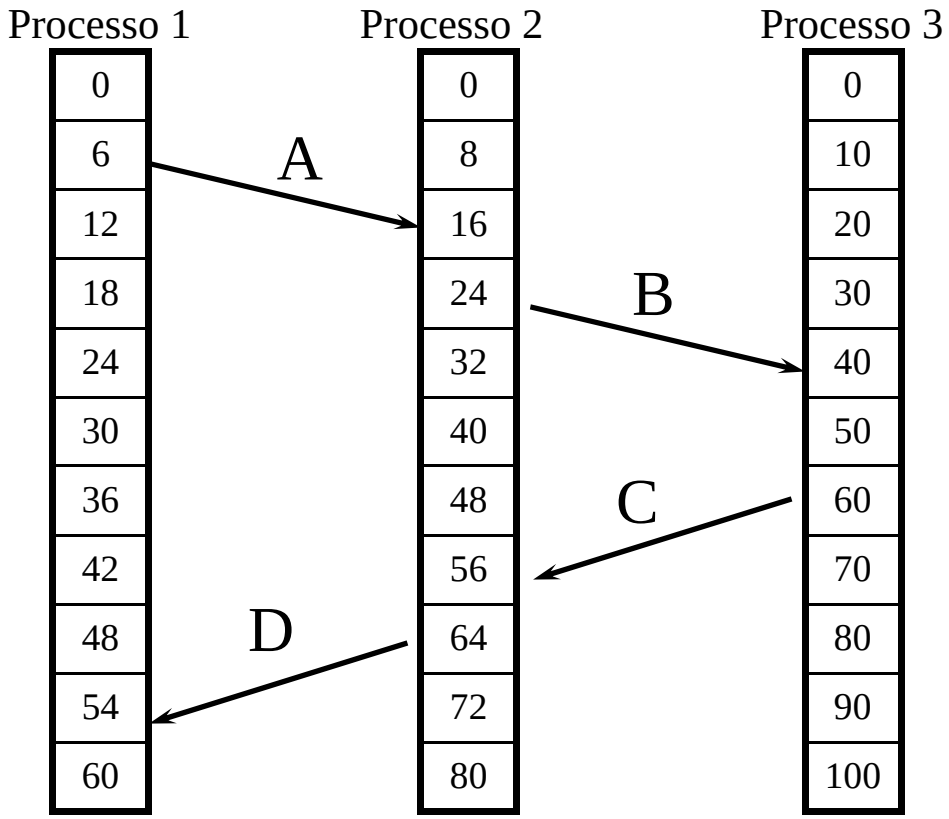
- Se os eventos x e y ocorrem em processos que não trocam mensagens (concorrentes)
  - É falso que:
    - $x \rightarrow y$ e $y \rightarrow x$

Solução de Lamport

- Se $a \rightarrow b$, então $C(a) \rightarrow C(b)$
  - *Clock de a é menor que o de b*
    - Clock como tempo

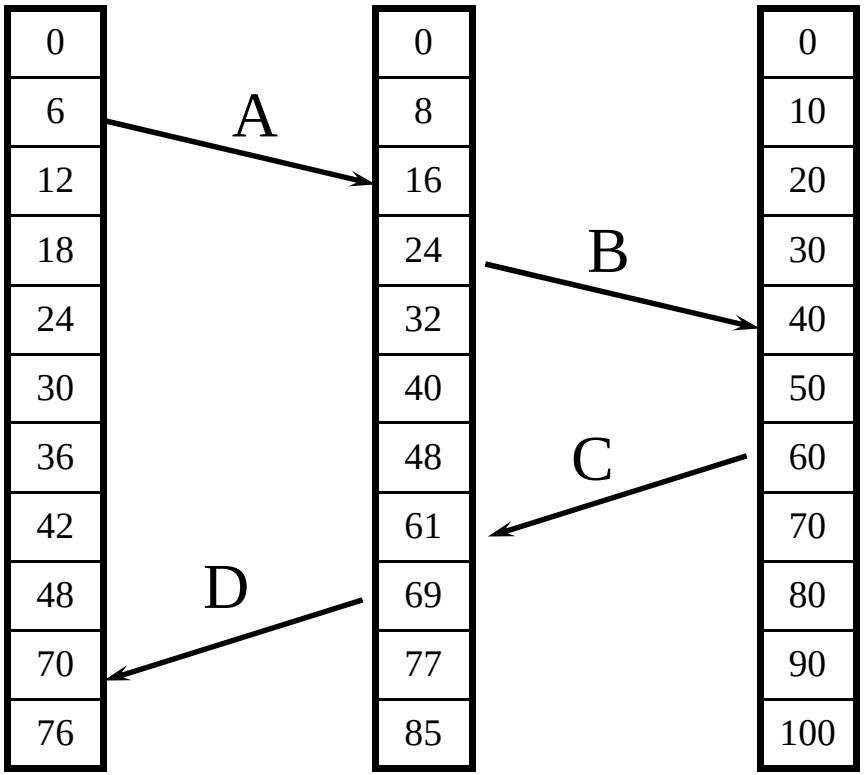
- Clock sempre é incrementado, nunca decrementado.

Solução de Lamport



Processos 1, 2 e 3 executam em máquinas distintas com diferenças de velocidade de Clock.

Solução de Lamport



Solução de Lamport

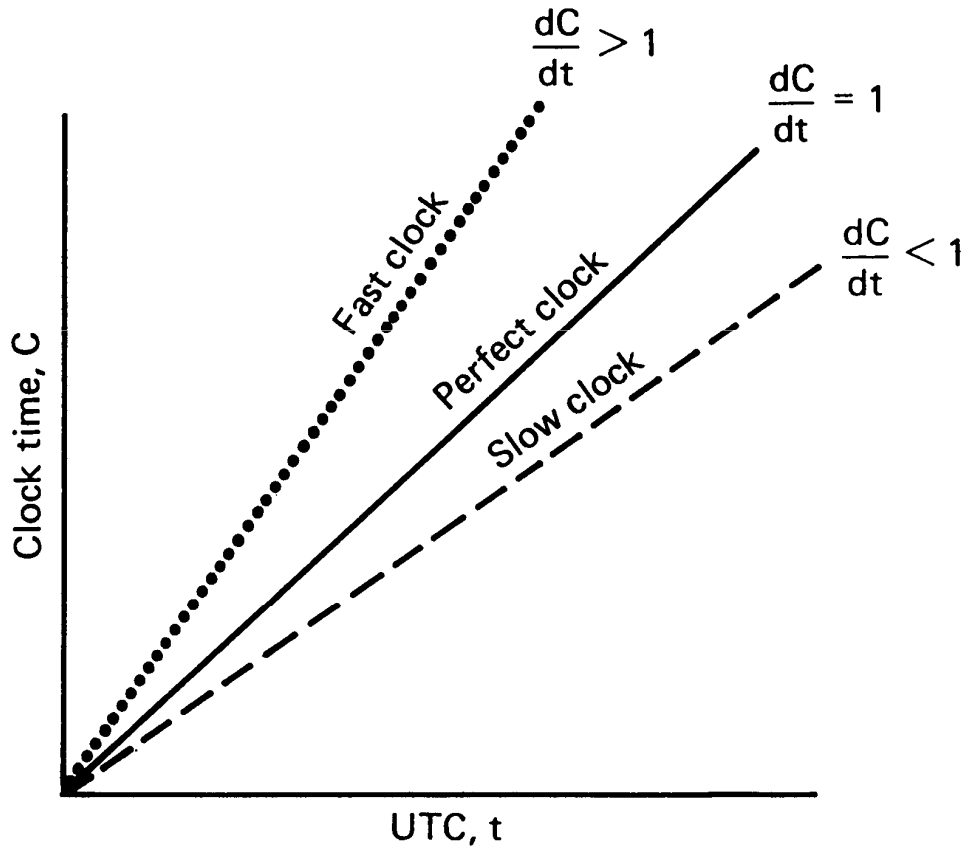
- Requisitos para Tempo Global
  - Entre dois eventos deve haver ao menos um Clock Tick.
    - Múltiplos eventos de SEND ou RECV sucessivos
      - Avançar o Clock
  - Dois eventos não devem ocorrer no mesmo tempo
    - ID do processo

Clock Físico

Clock Físico

- Clock tick (x pulsos por segundo)
  - Adiciona 1 ao tempo da memória
  - X = 60 → 216.000 Ticks por hora
    - Clocks podem atrasar ou adiantar
      - Taxa de erro
    - Comparação com tempo "real"

Clock Físico



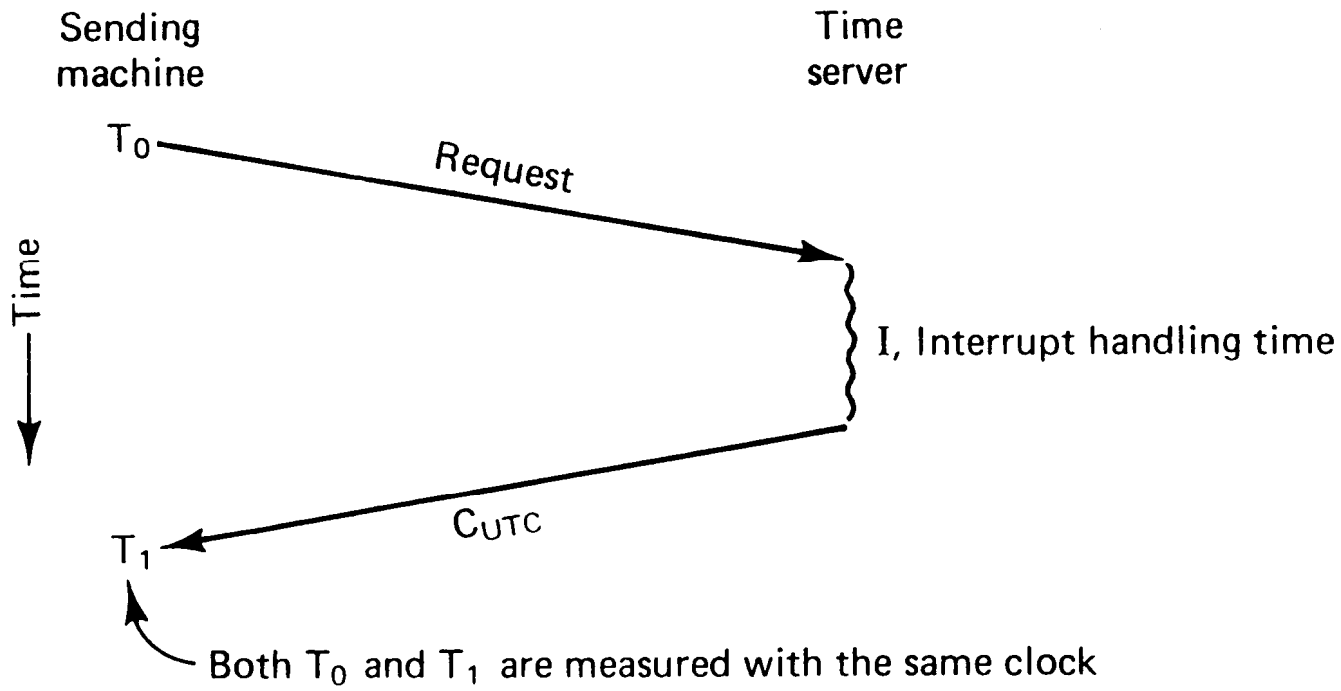
Algoritmo de Cristian

- Servidor de Tempo
  - Receptor WWV
  - Sincronização Externa (fonte externa)
- Clientes questionam a hora do servidor
  - Resposta imediata
- Problemas
  - Ajuste de Clock
    - Clock nunca deve ser atrasado
    - Overhead do Kernel do Servidor

Algoritmo de Cristian

- Relógio do computador *Ci* é sincronizado com uma fonte externa S:
  - $|S(t) - Ci(t)| < D$ para $i = 1, 2, \ldots N$ num intervalo I de tempo real
  - Os relógios *Ci* são exatos dentro do limite *D*

Algoritmo de Cristian



Algoritmo de Cristian

- **Um servidor de tempo ST recebe informação de uma fonte UTC**
  - Processo *p* requisita tempo em *T0* e recebe *t* em *T1* de *ST*
  - *p* ajusta o seu relógio para $t + (T1 - T0)/2$
  - Precisão é $\pm ((T1 - T0)/2 - min)$:
  - O instante mais recente em que ST coloca *t* na mensagem *mreply* é *min* depois de *p* enviar *mrequest*
  - O tempo mais atrasado foi *min* antes de *mreply* ter chegado ao processo *p*
  - *O tempo segundo o relógio de ST quando mreply* chega a *p* está no intervalo $[t+min, t + (T1 - T0) - min]$

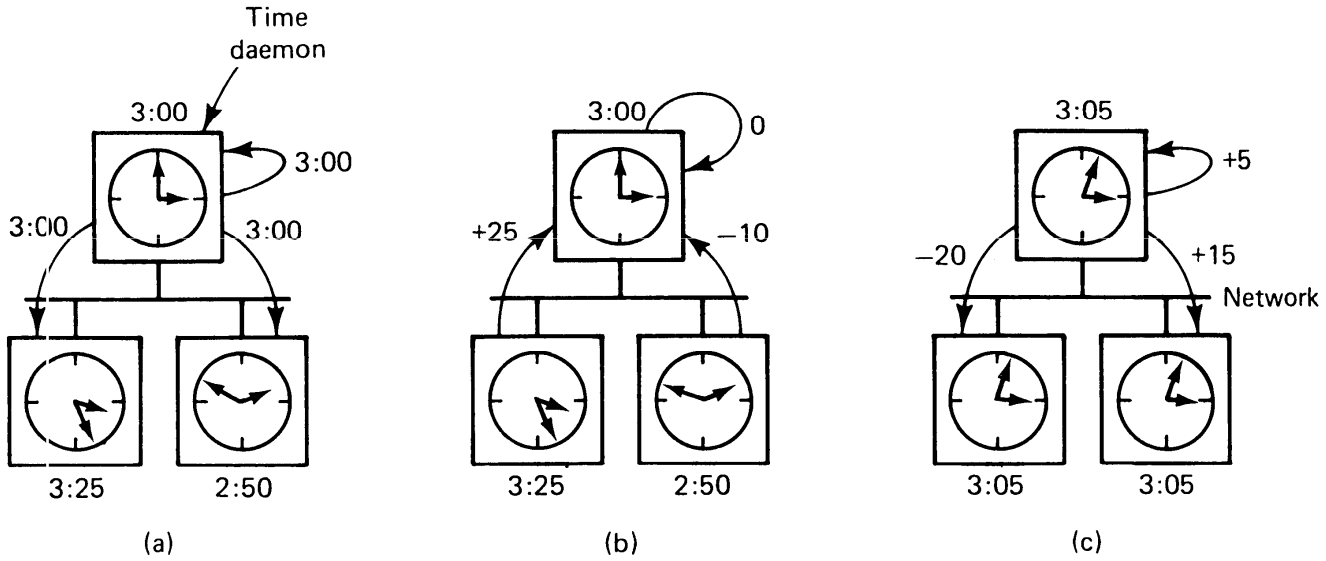
Algoritmo de Berkeley

- Servidor Ativo
  - Sincronização Interna (sincronização entre relógios)
    - Relógios podem não estar sincronizados externamente:
      - Pode existir ***drift*** coletivo
  - Questiona todas as máquinas pela hora local
  - Ajuste pela média
    - Avanço do relógio
    - Atraso via mudança de *ticks*

Algoritmo de Berkeley

- Os relógios de vários computadores sincronizam-se entre si:
  - $|Ci(t) - Cj(t)| < D$ para $i = 1, 2, \ldots N$ num intervalo I de tempo real
  - Os relógios *Ci* e *Cj estão sincronizados dentro do limite D*

Algoritmo de Berkeley



Clock Físico

- **Se um conjunto de processos P está sincronizado externamente com um limite D, então esses processos estão sincronizados internamente com um limite 2D:**
  - $Ci(t) = S(t) + D$, e $Cj(t) = S(t) - D$
  - $Ci(t) - Cj(t) = 2D$

Algoritmo de Médias

- Solução descentralizada
  - Cada máquina transmite, via broadcast, sua hora corrente
  - Cada máquina coleta o envio de todas as outras.
    - Após receber todas as mensagens, computa nova hora.
    - Soluções:
      - Média.
      - Média com descarte de extremos.

Relógio "Correto"

- **Um relógio é correto se o seu ritmo de desvio *rd* (*drift rate*) é limitado**
  - ***rd*** > 0 (ex. 10^{-6} segs/ seg)
- **Erro de medição do intervalo entre *t* e *t'* é limitado:**
  - (1 - ***rd***) (*t'* - *t*) <= *H*(*t'*) - *H*(*t*) <= (1 + ***rd***) (*t'* - *t*) (onde *t'* > *t*)
- Isto impede saltos no valor do relógio
- Ex.: ***rd = 0,01segs/seg t' = 1000 e t = 900***
  - (1 - ***rd***) (*t'* - *t*) = 0,99 * 100 = 99
  - *H*(*t'*) - *H*(*t*) = 100
  - (1 + ***rd***) (*t'* - *t*) = 1,01 * 100 = 101

Sincronização em SD Síncrono

- **Sistema distribuído síncrono é aquele em que os limites seguintes estão definidos:**
  - Tempo de execução de cada passo num processo tem limites inferior e superior
  - Cada mensagem transmitida na rede é entregue dentro de um limite máximo de tempo
  - Cada processo tem um relógio local cujo desvio em relação ao tempo real tem um limite conhecido

Sincronização em SD Síncrono

- **Sincronização interna num sistema síncrono:**
  - O processo p1 envia o seu tempo local ***t*** para o processo p2 numa messagem m
  - p2 ajusta o seu relógio para ***t*** + ***Ttrans*** onde ***Ttrans*** é o tempo de transmissão da mensagem m
  - ***Ttrans*** é desconhecido mas ***min <= Ttrans <= max***
  - Incerteza é: ***i = max - min***
  - Ajustar relógio para o valor: ***t + (max + min)/2***
  - A diferença entre os relógios é: ***skew <= i/2***

Sincronização em SD Síncrono

- p2 pode fazer t2 = t + max:
  - skew pode ser = i
- p2 pode fazer t2 = t + min:
  - skew pode ser = i
- p2 faz t2 = t + ((max+min)/2):
  - (t+max)-(t+(max+min)/2) =
  - t+max-t-max/2-min/2=
  - max/2-min/2=i/2

Sincronização em SD Síncrono

- p2 faz t2 = t + ((max+min)/2):
  - t+((max+min)/2)-(t+min) =
  - t+max/2+min/2-t-min=
  - max/2-min/2 = i/2
- A diferença entre os relógios é:
  - skew <= i/2

Sincronização em SD Síncrono

- **A Internet é um sistema síncrono?**
  - Ttrans = min + x onde x >= 0
  - **Sistema Assíncrono**