Computação Distribuída

Odorico Machado Mendizabal

Universidade Federal de Santa Catarina – UFSC
Departamento de Informática e Estatística – INE

Tempo e Sincronização

Por que sincronização é importante?

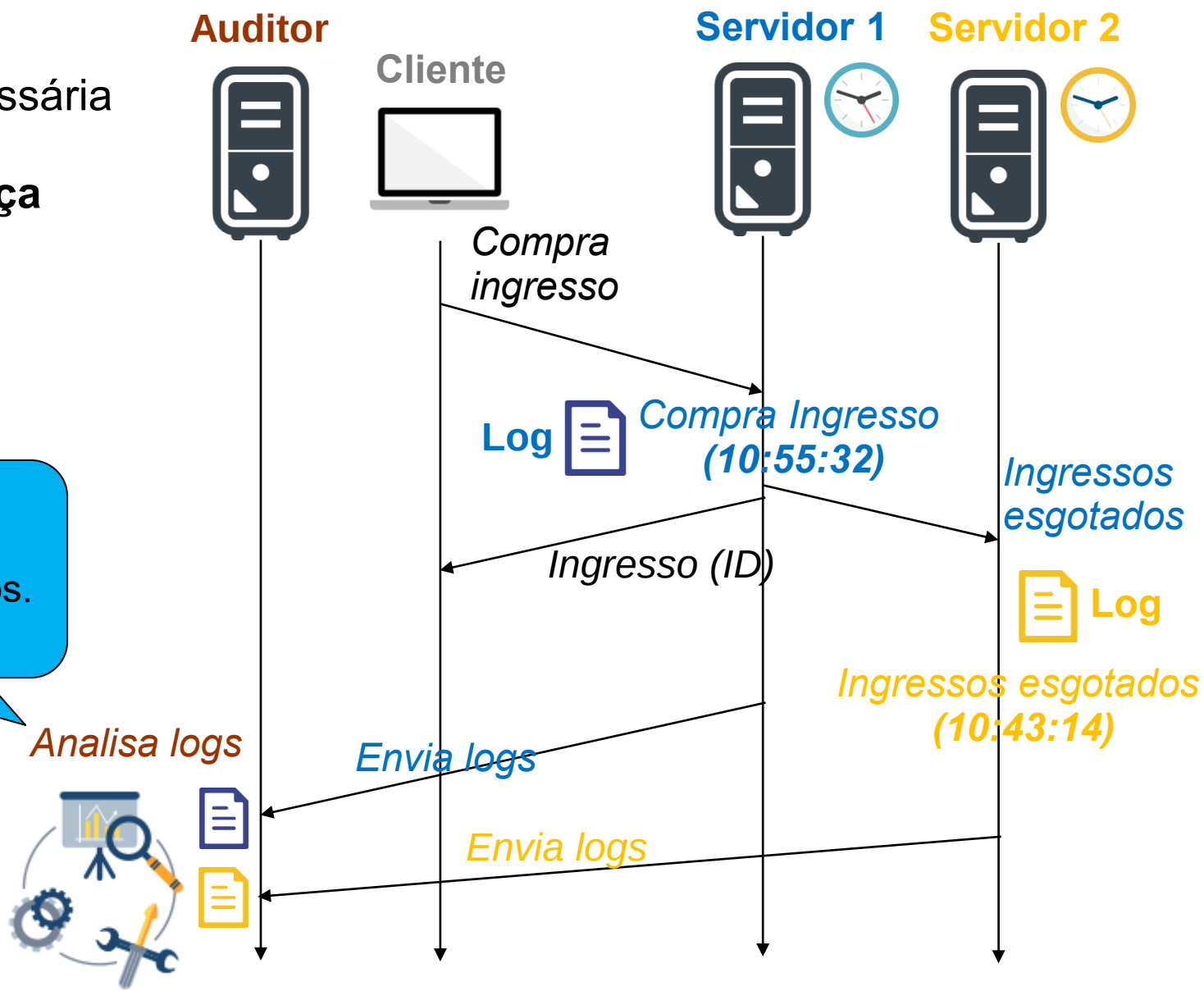
- Exemplo: Realização de um concurso público.
  "O portão para acesso às salas fecha às 13:00"

  Falta de sincronismo
  - Defasagem na leitura dos relógios (candidato e local da prova),
  - Atraso ou antecipação no translado (tempo de deslocamento)

  Consequências
  - Se o candidato chega 30 minutos atrasado:
    - Perde a prova
  - Se chegar 30 adiantado:
    - Haverá um tempo de espera exagerado (ociosidade)

Por que sincronização é importante em sistemas distribuídos?

Sincronização é necessária para prover **correção (*correctness*)** e **justiça (*fairness*)**



Parece que o cliente comprou um ingresso após estarem esgotados. Como é possível?

Sincronização em Sistemas Distribuídos

- Computação distribuída
  - Processos colaborativos
  - Coordenação de ações, sem violar propriedades de segurança (*safety*) e progresso (*liveness*)

  - Podemos coordenar atividades com base no tempo absoluto dos eventos?

Sincronização baseada em tempo (uso de relógios)

Em sistemas centralizados

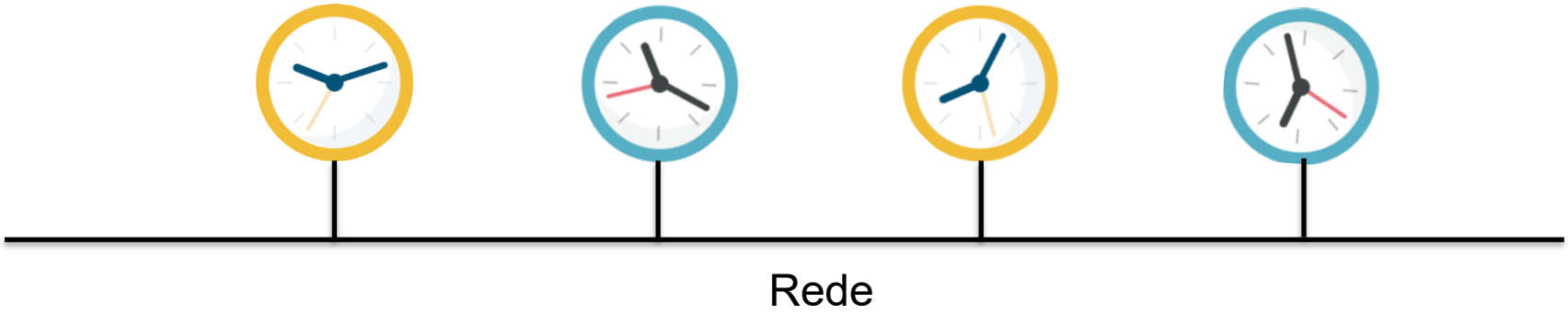
É possível verificar se um evento A ocorreu antes de um evento B?
É possível forçar que o evento A ocorra antes que B?

Sim, há um relógio centralizado comum à todos os processos

Em sistemas distribuídos

É possível verificar se um evento A ocorreu antes de um evento B?
É possível forçar que o evento A ocorra antes que B?

Não há um relógio comum que permita leituras instantâneas



Quais são os desafios?

Nodos distribuídos

- Cada nodo possui o seu próprio relógio (temporizador)
- Características físicas do cristal usado no temporizador e o projeto de HW podem ocasionar defasagem entre relógios em diferentes computadores

Processos em Computação Distribuída seguem um modelo de sistema assíncrono

- Não há limites
  - Para atraso de mensagens
  - Atrasos no processamento

Definições

- Sistema distribuído assíncrono consiste de um número de processos
- Cada processo tem um estado (valores de variáveis)
- Processos executam ações, que modificam o seu estado
  - Execução de uma instrução local
  - Comunicação (envio ou recebimento de mensagem)
- Um evento é a ocorrência de uma ação
- Cada processo tem seu relógio local
  - Eventos locais podem ser associados à *timestamps*
  - Há uma ordem linear entre eventos locais

Clock Skew (desvio) e Clock Drift (derivação)

Clock Skew: Diferença relativa entre o valor de 2 relógios:

Exemplo: Leitura em um instante referencial t:

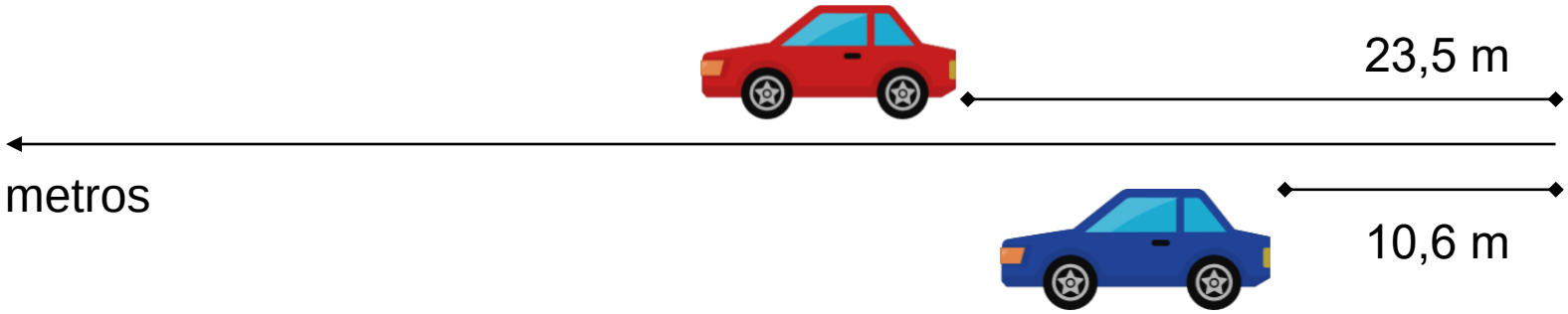
```
Clock1 = 10:00.005
Clock2 =  9:58.000
Skew =       2.005
```

Clock Drift: Diferença na frequência de contagem de dois relógios

- A representação da passagem do tempo é feita com base em algum referencial
- Exemplos:
  - Relógios quartz simples: 10^{-6} seg/seg
    Atrasa 1 segundo a cada 11,6 dias
  - Relógios quartz de precisão: 10^{-8} seg/seg
    Atrasa 1 segundo a cada 3 anos e 2 meses
  - Relógios atômicos: 10^{-11} seg/seg
    Atrasa 1 segundo a cada 3 mil anos

Clock Skew (desvio) e Clock Drift (derivação): Analogia

Clock Skew: Diferença relativa entre a posição de 2 veículos:
Exemplo: Leitura em um instante referencial t:



Clock Drift: Diferença de velociade entre dois carros

Representação do Tempo

Hora Coordenada Universal (UTC)
Baseada na hora atômica, este padrão substituiu o antigo *Greenwich Mean Time* (GMT)

A maioria dos sistemas utiliza este padrão

Para atualizar relógios, estações geram pulsos a cada segundo UTC

- *Através de ondas curtas de rádio (precisão ~10ms)*
- *Via satélite (precisão ~0,5ms)*

Servidores obtém o informação usando receptores de rádio ou via satélite
Ex. Servidores UTC na Internet:

http://www.worldtimeserver.com/?locationid=UTC

Sincronização de Relógios

Computadores podem basear-se em servidores centrais para informar o tempo

Porém, características dos temporizadores locais das máquinas, além da latência de troca de mensagens entre solicitante e servidor de tempo podem ocasionar defasagem entre relógios

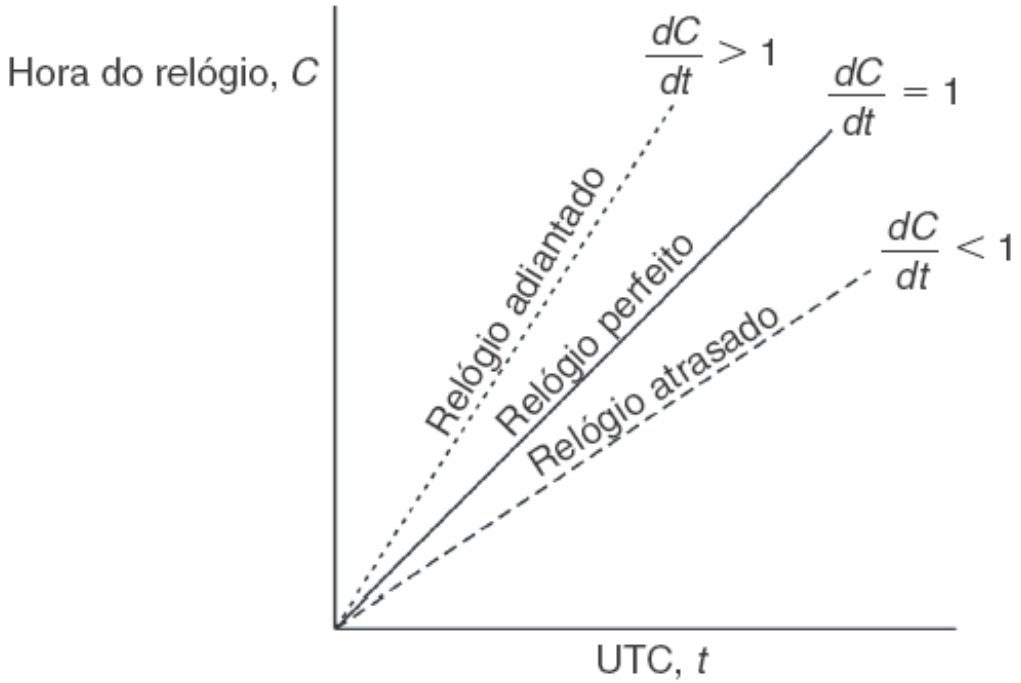


Figura 6.5 Relação entre a hora do relógio e a hora UTC quando as taxas de ciclos de relógios são diferentes.

Sincronização de Relógios

É preciso oferecer um método de ajuste para minimizar a defasagem entre relógios

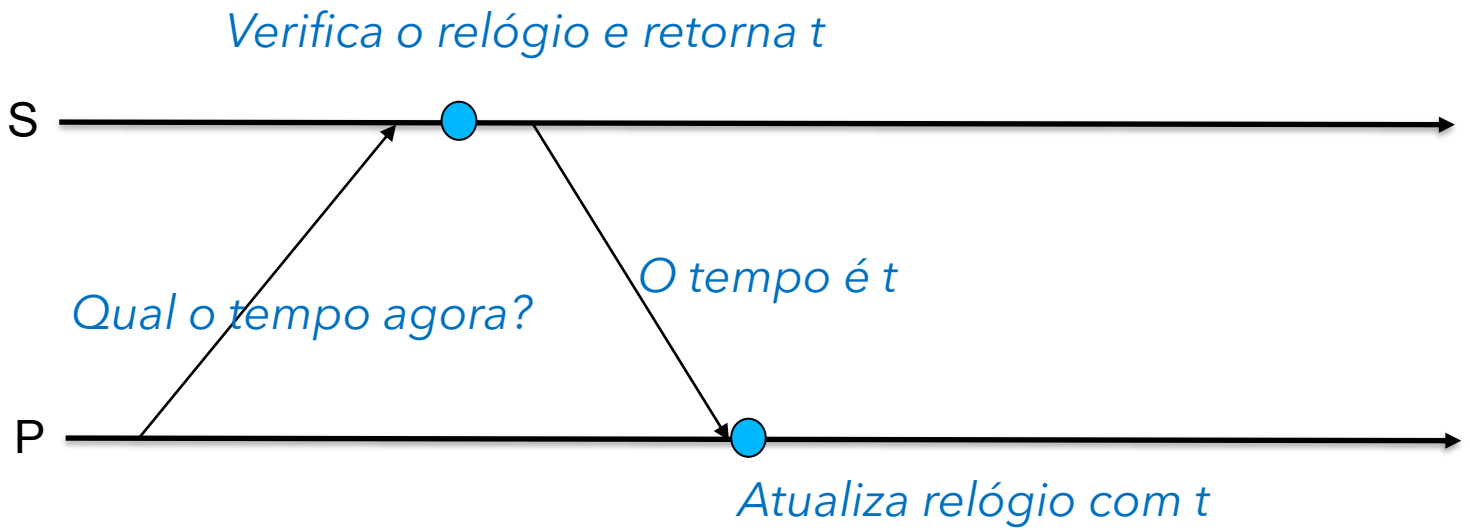
- Sincronização externa:
  - Um relógio externo pode estar conectado a um servidor UTC ou a um relógio atômico
  - Processos consultam o relógio de referência (remotamente)
  - Ex. Algoritmo de Cristian, NTP

- Sincronização interna
  - Cada processo em um grupo tem seu relógio local
  - Processos trocam suas medições de tempo visando acordar sobre um tempo aproximado (com um erro determinado)
  - Ex. Algoritmo de Berkeley

Sincronização externa

Intuição

- Todos os processos P sincronizam com um servidor de tempo S



- Adotar o valor t em P é impreciso
- Esta abordagem não considera a latência das mensagens e processamento em S
- A imprecisão não pode ser medida em sistema assíncronos

Algoritmo de Cristian

Proposto por Cristian (1989)

- Para sistemas assíncronos

- Clientes consultam tempo em servidor de tempo

Distributed Computing (1989) 3:146–158

DISTRIBUTED COMPUTING

Probabilistic clock synchronization

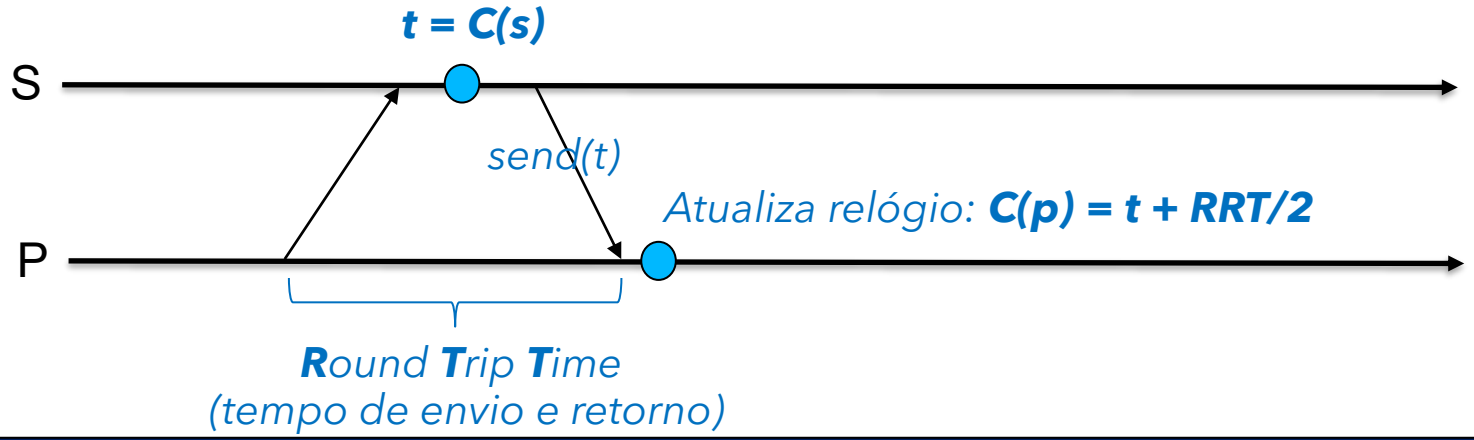
Flaviu Cristian
IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA

Flaviu Cristian is a computer scientist at the IBM Almaden Research Center in San Jose, California. He received his PhD from the University of Grenoble, France, in 1979. After carrying out research in operating systems and programming methodology in France, and working on the specification, design, and verification of fault-tolerant programs in England, he joined

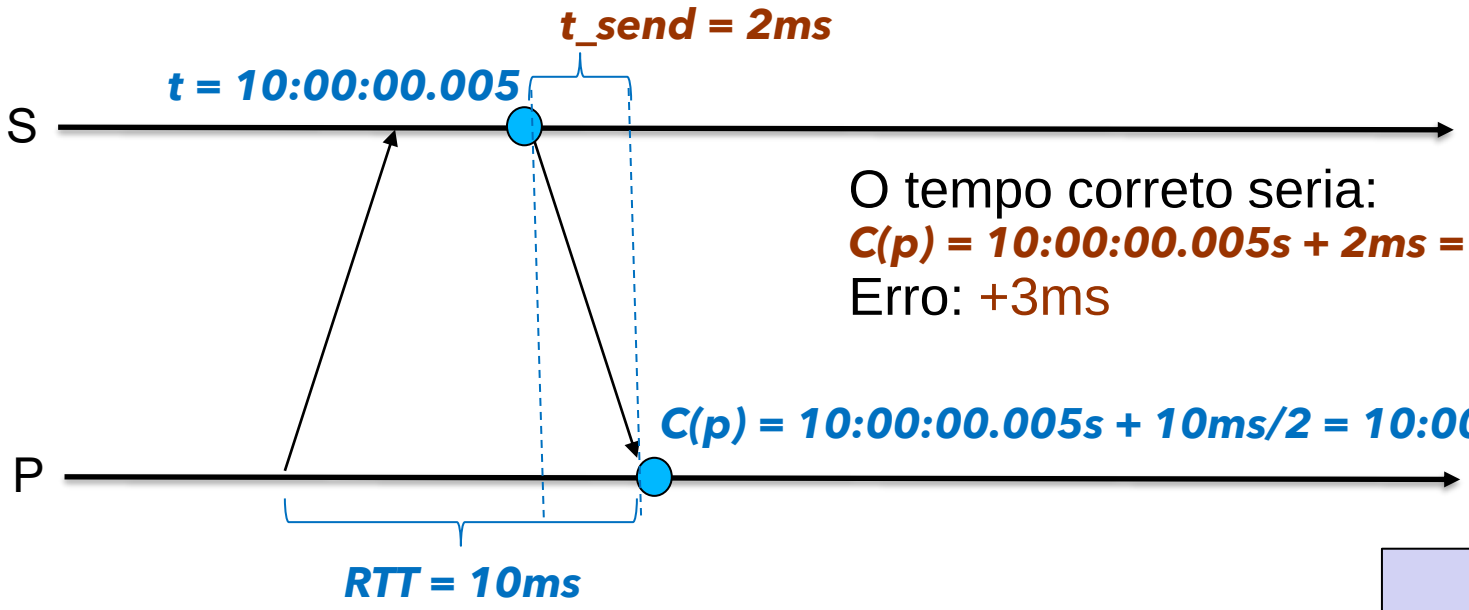
in some given maximum derivation from a time reference external to the system. *Internal* clock synchronization keeps processor clocks within some maximum relative deviation of each other. Externally synchronized clocks are also internally synchronized. The converse is not true: as time passes internally synchronized clocks can drift arbitrarily far from external time.

Clock synchronization is needed in many distributed systems. Internal clock synchronization enables one to measure the duration of distributed

- Servidor de tempo **S** (conectado a uma fonte UTC)
- Cliente **P** requisito o tempo em **S**
- Servidor fornece o tempo **t**
- Cliente ajusta o seu relógio em **t + RTT/2**



Algoritmo de Cristian – Precisão



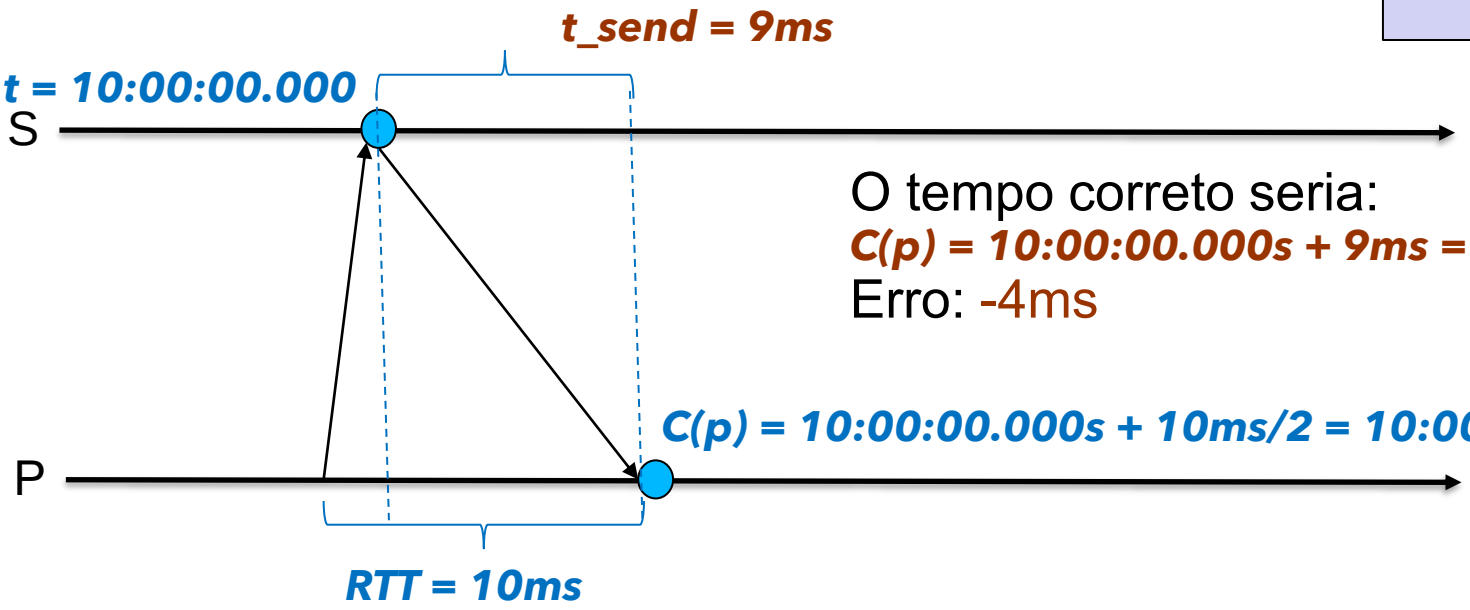
O tempo correto seria:

C(p) = 10:00:00.005s + 2ms = 10:00:00.007

Erro: +3ms

C(p) = 10:00:00.005s + 10ms/2 = 10:00:00.010

Precisão = (RTT/2 – t_{min})



O tempo correto seria:

C(p) = 10:00:00.000s + 9ms = 10:00:00.009

Erro: -4ms

C(p) = 10:00:00.000s + 10ms/2 = 10:00:00.005

Algoritmo de Cristian - Considerações

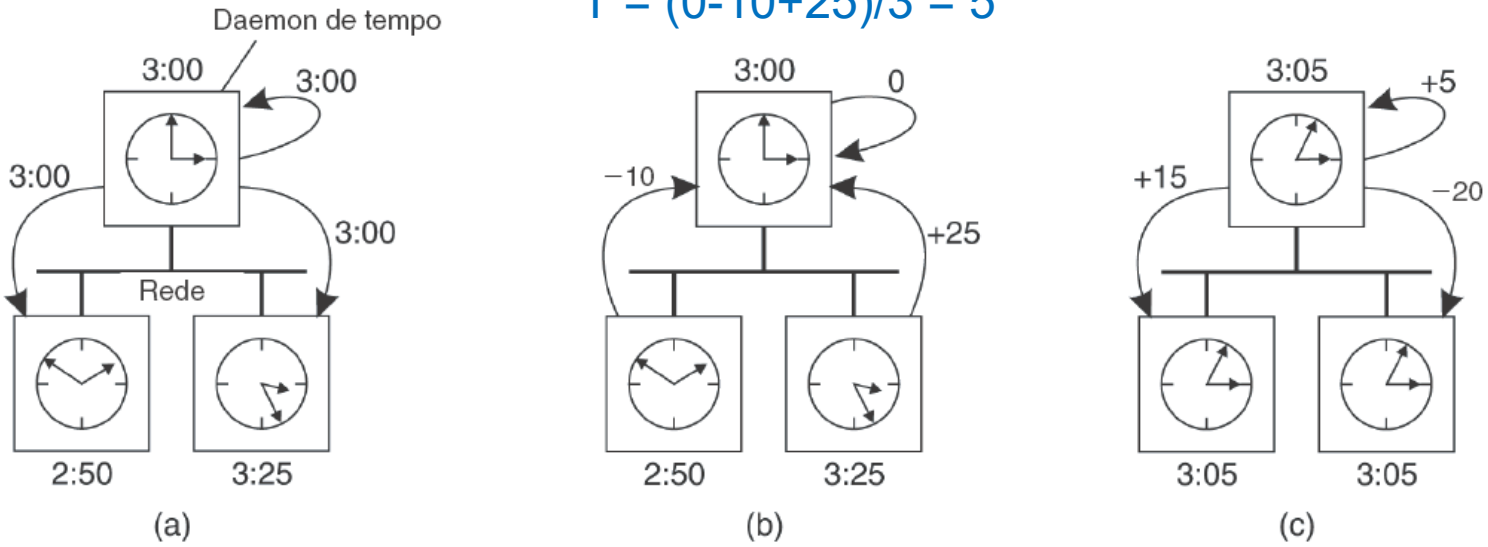
- Tempo médio de latência é calculado baseado em *round trip* da mensagem:
  - A latência das mensagens pode variar muito em sistemas abertos (ex. Internet)
  - Custo do SO para “bufferizar” mensagens, tempo do TCP para enfileirar mensagens no remetente e destinatário, etc. não é observado neste algoritmo
- Para aumentar a precisão, pode-se fazer várias requisições ao servidor, visando obter um RTT mínimo

Problemas

- Servidor S pode ser ponto único de falhas
  - Solução: Replicar o servidor S
- Servidores maliciosos podem difundir horários falsos
  - Solução: algoritmos de tolerância a falhas bizantinas

Sincronização Interna: Algoritmo de Berkeley

- Conjunto de processos em um grupo
- Nodo mestre coordenada a sincronização de relógios
  - Periodicamente pergunta a hora dos demais participantes
  - Com base nas respostas, calcula o tempo médio e diz qual ajuste cada participante deve fazer



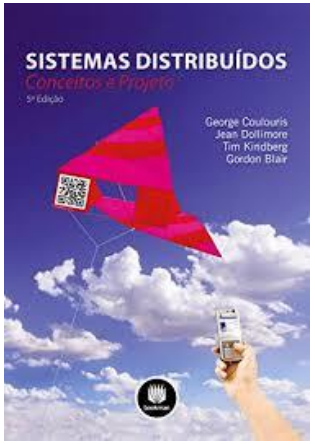
Para ficar em 3:05
T1 = (3:00) **+5**
T2 = (2:50) **+15**
T3 = (3:25) **-20**

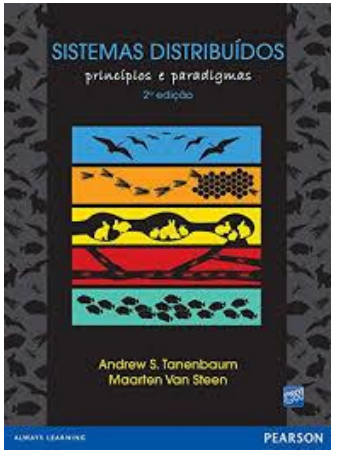
Em alguns casos basta que os computadores concordem com uma determinada hora, não sendo necessário que esta seja a hora certa

Referências

Parte destes slides são baseadas em material de aula dos livros:

- *Coulouris, George; Dollimore, Jean; Kindberg, Tim; Blair, Gordon. Sistemas Distribuídos: Conceitos e Projetos. Bookman; 5ª edição. 2013. ISBN: 8582600534*
- *Tanenbaum, Andrew S.; Van Steen, Maarten. Sistemas Distribuídos: Princípios e Paradigmas. 2007. Pearson Universidades; 2ª edição. ISBN: 8576051427*





- *Imagens e clip arts diversos:*
  https://free-icon-rainbow.com/
  https://www.gratispng.com/