

Sincronização em SD

Sistemas Distribuídos

2024

Professor Vinícius Hax

- ❖ Comunicação indireta

- ❖ Lembrando:

- ❖ Na comunicação DIRETA quem manda a mensagem tem que dizer para quem a mensagem se direciona
- ❖ Na comunicação INDIRETA não precisamos dizer para quem é a mensagem. Ela vai para um lugar compartilhado e os interessados buscam ler.

- Sincronização
- Protocolo de sincronia de relógio
- Exclusão mútua

- Significado: sin (junto) + cronos (tempo) = juntos no tempo
- Então o tempo é essencial para a sincronia

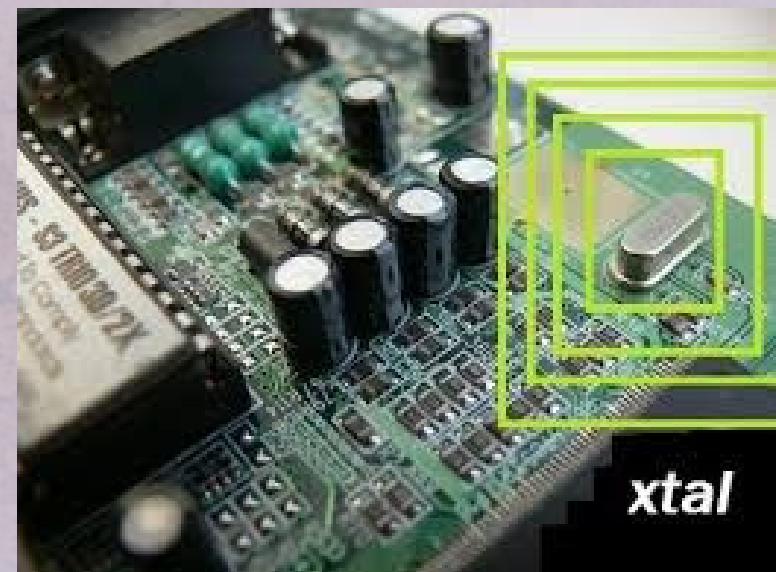
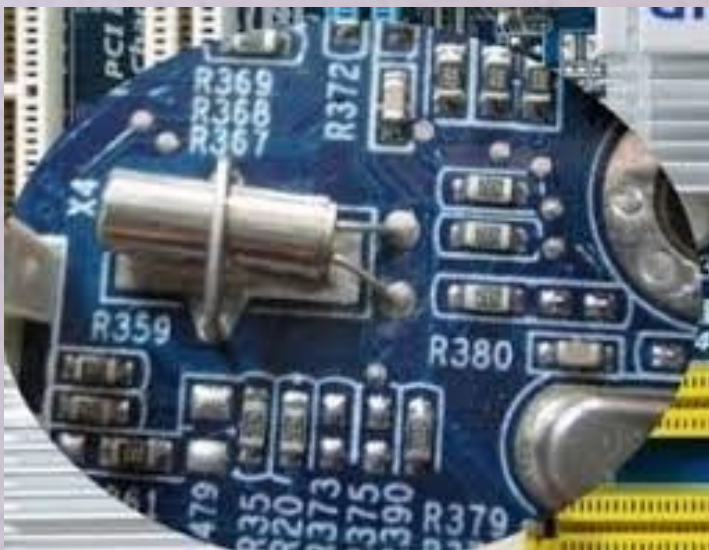
- Em alguns sistemas saber a hora absoluta é essencial
 - Sistema make, que compila softwares sob demanda com base no horário de edição
 - Sistemas de logs
- Em outros a ordem dos acontecimentos é o que realmente importa
 - Antes de fazer um saque no banco tem que ter havido um depósito
 - Antes de um administrador fazer algo no sistema ele tem que ter obtido as permissões necessárias

Problemas famosos decorrentes de erro de sincronia

- ❖ Bug do milênio
 - 🟡 Em alguns sistemas a data era representada por dois dígitos: 1997 virava 97, 1999 virava 99, etc
 - 🟡 Na virada do milênio 2000 viraria 00. Como 00 é menor do que 99, dependendo do sistema é como se o relógio estivesse em 1900!
- ❖ Sistemas Unix poderão ter um problema semelhante em 2038
 - 🟡 Ver https://en.wikipedia.org/wiki/Year_2038_problem

Relembrando ...

- Como os computadores armazenam a hora, mesmo sem Internet? Existe um componente (cristal) que possui uma vibração periódica que fica na placa mãe

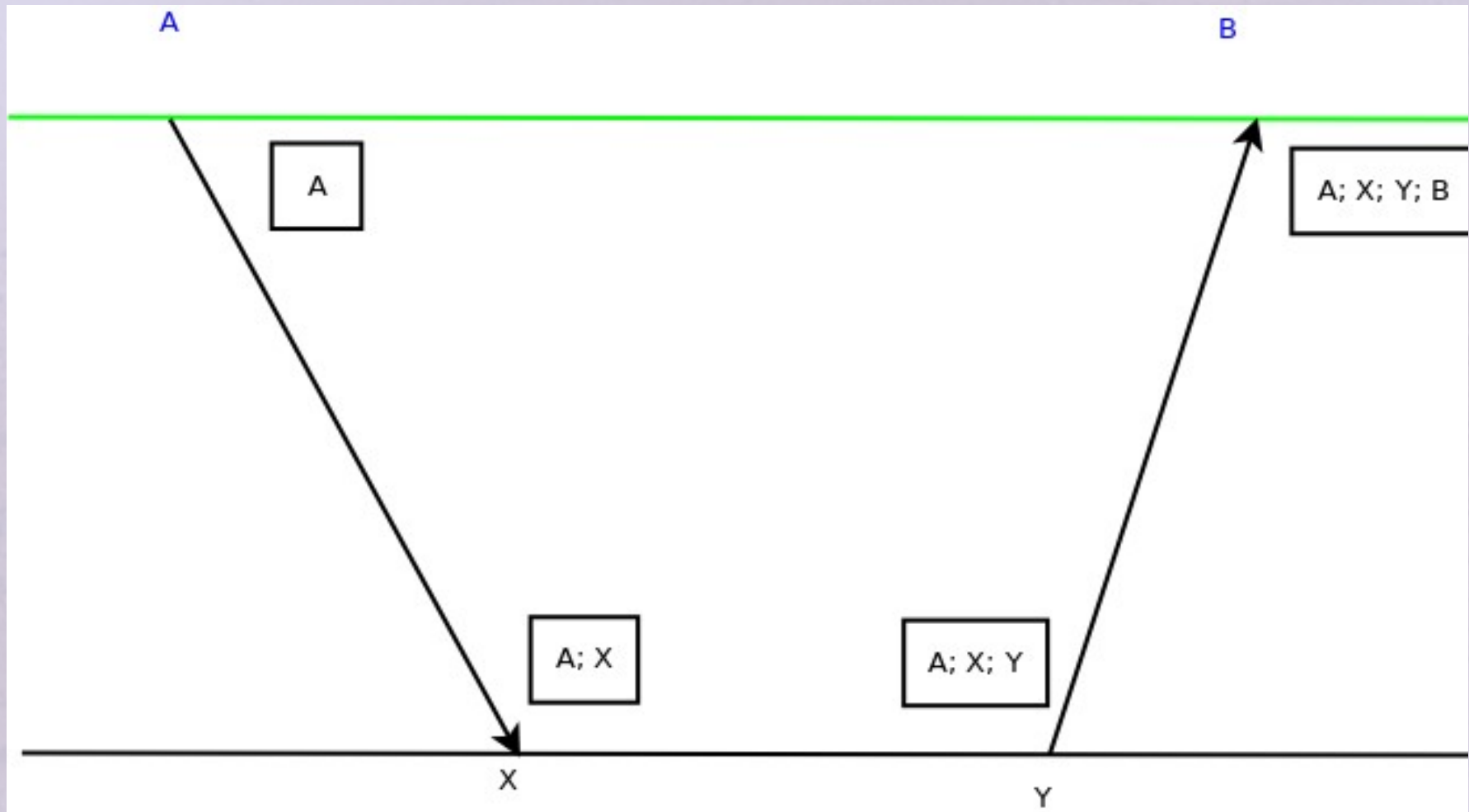


- Ao desligar o computador uma bateria alimenta o cristal que mantém a contagem de tempo. Ao ligar o relógio segue atualizado

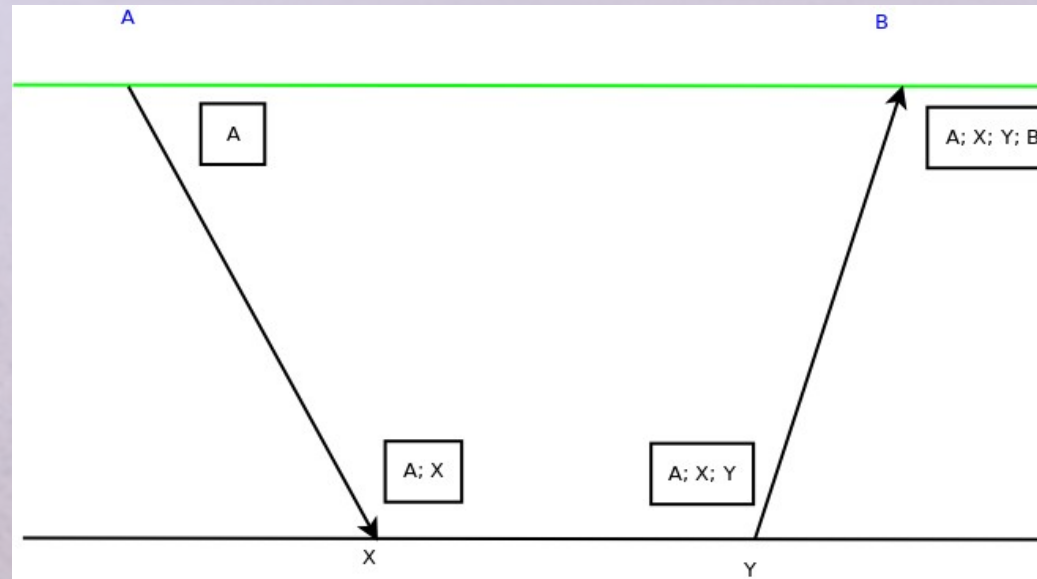
Sincronização de relógios com Internet

- Quando se possui acesso à Internet é possível sincronizar relógios através de um protocolo específico para esse fim: o NTP (Network Time Protocol)

Funcionamento do NTP

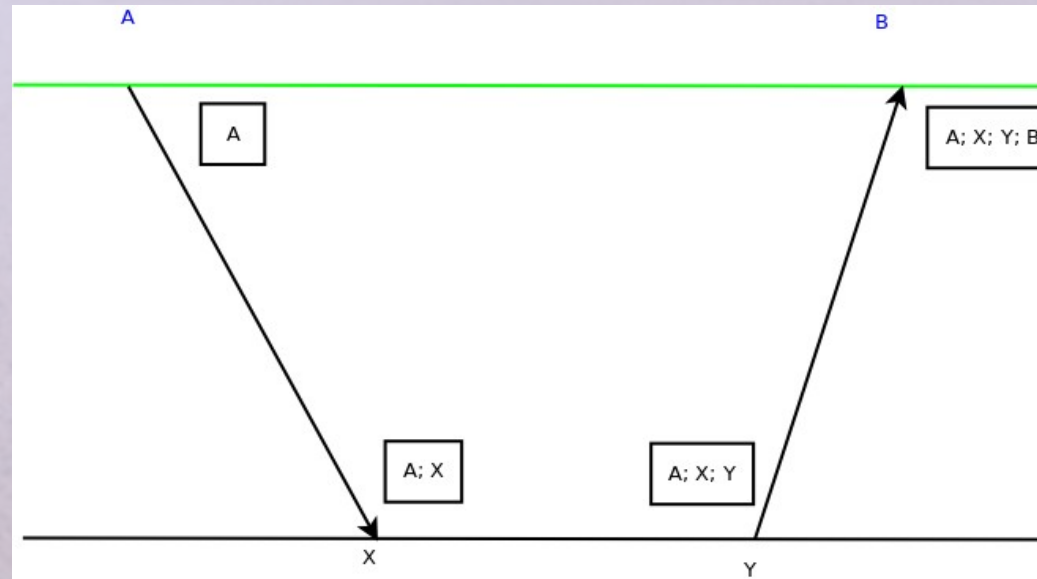


Funcionamento do NTP (2)

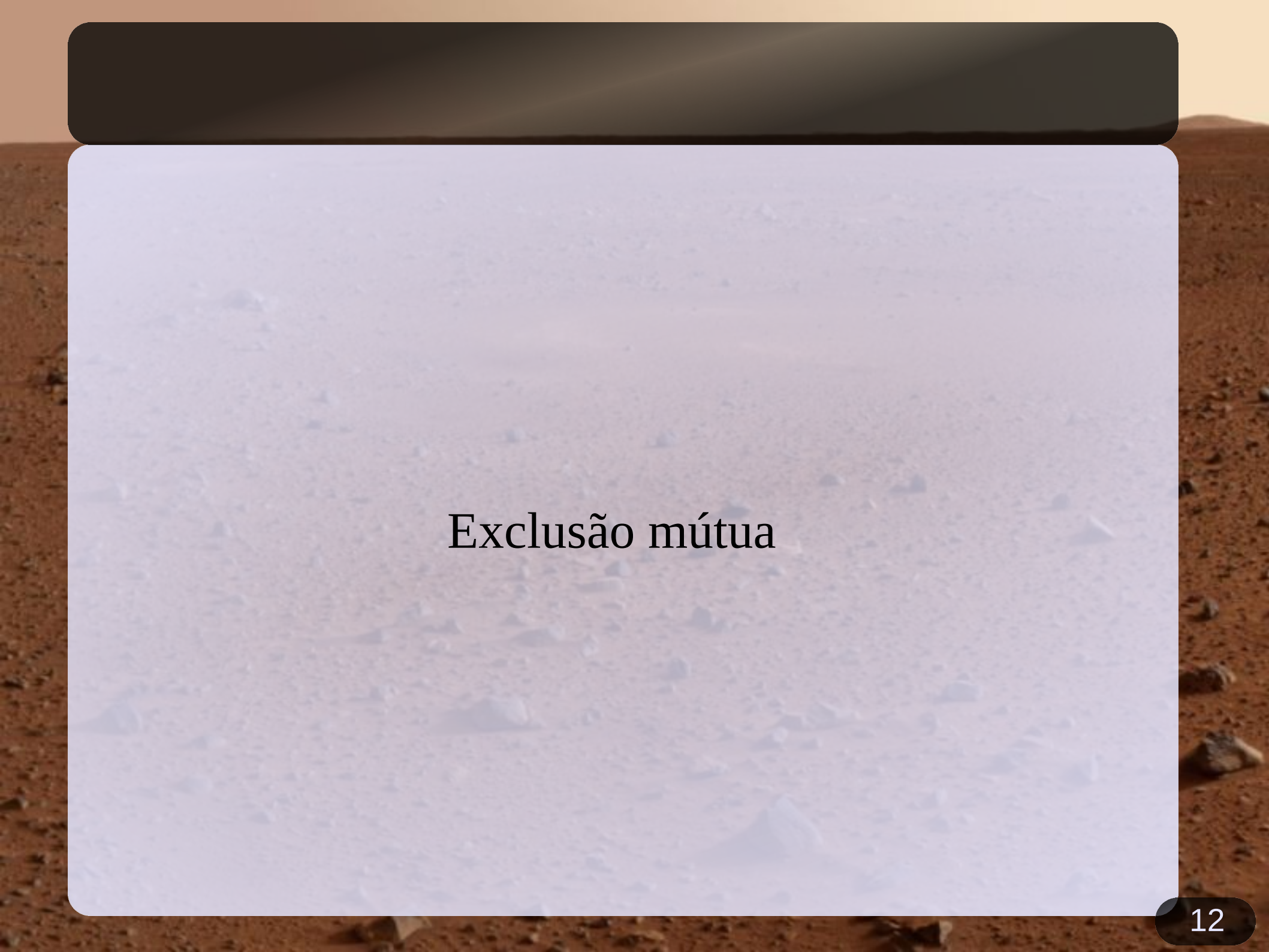


- Tendo o delay podemos calcular a diferença no relógio pois:
- $A + \text{delay} + \text{dif} = X$
- $\text{dif} = X - \text{delay} - A$

Funcionamento do NTP (3)



- $B - A = \text{delay_ida} + (Y - X) + \text{delay_volta}$
- Supondo delay_ida igual ao delay_volta , podemos calcular o delay
- $B - A = 2 * \text{delay} + (Y - X)$
- $\text{delay} = [B - A - (Y - X)] / 2$

The background of the slide is a photograph of a desert landscape, likely Mars, showing reddish-brown soil and small rocks. A large, semi-transparent white rectangle is centered on the slide, serving as a background for the text. At the top of the slide, there is a solid black horizontal bar.

Exclusão mútua

- Um problema clássico de sistemas distribuídas é o da exclusão mútua
- Esse problema acontece quando vários processos precisam acessar um recurso de maneira exclusiva, por um tempo
- Exemplo:
 - Três servidores precisando escrevendo em dois arquivos de log compartilhado pela rede
 - Eles não podem escrever ao mesmo tempo senão os caracteres ficariam embaralhados
 - Como gerenciar as permissões?
- Dois problemas comuns: deadlock e starvation

Problemas de sincronização

❶ Deadlock

- ❶ Processo A está com recurso X mas precisa do recurso Y para concluir uma operação. Processo B está com recurso Y mas precisa do recurso X para concluir.
- ❶ Se nenhum processo desistir ou se não houver uma intervenção externa ambos ficarão “travados até a morte”: deadlock
- ❶ Exemplo: dois alunos querem fazer o TCC com PHP e MySQL. Aline pegou o único livro de PHP da biblioteca. Ao mesmo tempo Bernardo pegou o único livro de MySQL. Quando Aline notou que queria o livro de MySQL e Bernardo o de PHP, nenhum consegue pegar o livro do outro mas nenhum quer devolver o seu livro
- ❶ Possível solução: intervenção do bibliotecário

Problemas de sincronização

- Starvation: Dois processos querem o mesmo recurso mas o algoritmo sempre prioriza um processo em detrimento de outro
- Exemplo: Imagine um supermercado que só tem um caixa e ele é um caixa prioritário para idosos, por exemplo. Poderia ocorrer, que um jovem muito azarado fique esperando por horas nesse supermercado se 20 idosos forem fazer compras em sequência.

Os algoritmos de exclusão mútua tem que evitar, ou se não for possível minimizar tanto deadlocks quanto starvation

Algoritmos de exclusão mútua

- Principais tipos
 - Centralizado
 - Distribuído
 - Baseado em token

Algoritmo centralizado

- Um processo fica responsável por controlar quem acessa os recursos
 - Como esse processo tem todas as informações necessárias é possível intervir e precaver tanto deadlocks quanto starvation
 - Possíveis problemas: ?

Algoritmo centralizado

- Um processo fica responsável por controlar quem acessa os recursos
 - Como esse processo tem todas as informações necessárias é possível intervir e precaver tanto deadlocks quanto starvation
 - Possíveis problemas: ... E se o processo travar? Tudo fica centralizado nele logo o sistema depende do mesmo

Algoritmo distribuído

- Antes de usar um recurso o processo pergunta na rede se alguém está usando o recurso
 - Se ninguém responder que está usando, o processo que perguntou pode começar a usar
 - Problemas? ...

Algoritmo distribuído

- Antes de usar um recurso o processo pergunta na rede se alguém está usando o recurso
 - Se ninguém responder que está usando, o processo que perguntou pode começar a usar
 - Problemas?
 - Cada processo tem que manter a lista de todos os outros processos
 - E se o processo que está usando o recurso não conseguir responder porque travou?

Algoritmo distribuído

- Antes de usar um recurso o processo pergunta na rede se alguém está usando o recurso
 - Se ninguém responder que está usando, o processo que perguntou pode começar a usar
 - Problemas?
 - Cada processo tem que manter a lista de todos os outros processos
 - E se o processo que está usando o recurso não conseguir responder porque travou?
 - Esse aspecto pode ser resolvido definindo que um processo sempre responde, seja que está usando o recurso ou que não está. Com isso sabemos quem está indisponível.

Algoritmo distribuído

- Antes de usar um recurso o processo pergunta na rede se alguém está usando o recurso
 - Se ninguém responder que está usando, o processo que perguntou pode começar a usar
 - Problemas?
 - Cada processo tem que manter a lista de todos os outros processos
 - E se o processo que está usando o recurso não conseguir responder porque travou?
 - Esse aspecto pode ser resolvido definindo que um processo sempre responde, seja que está usando o recurso ou que não está. Com isso sabemos quem está indisponível.
 - Como todos se envolvem em todas os acessos, o volume de mensagens pode ser muito grande

Algoritmo baseado em token

- Cada recurso é associado a uma “ficha”;
- Criar um “círculo virtual” com todos os projetos e ir passando uma “ficha” de permissão no círculo, um por vez. Cada processo que receber a ficha de um recurso pode usá-lo ou então passar a ficha adiante. Não é possível usar o recurso sem a ficha.
- Problemas?

Algoritmo baseado em token

- Cada recurso é associado a uma “ficha”;
- Criar um “círculo virtual” com todos os projetos e ir passando uma “ficha” de permissão no círculo, um por vez. Cada processo que receber a ficha de um recurso pode usá-lo ou então passar a ficha adiante. Não é possível usar o recurso sem a ficha.
- Problemas?
 - Como diferenciar problemas na ficha de um acesso demorado, mas válido, com a ficha?

- Imagem cristal 1:
<https://www.clubedohardware.com.br/forums/topic/1376087-cristal-defeituoso-na-placa-m%C3%A3e/>
- Imagem cristal 2:
<http://picsource.com.br/archives/10305>

Referências e imagens

- ❖ “Sistemas distribuídos”, Tanenbaum & Van Steen