UNIVERSIDADE DE ÉVORA

Sistemas Operativos II

Sincronização de Relógios
e
Exclusão Mútua a nível Distribuído

Introdução

a noção de tempo é **importante** nos SD

interessa calcular a hora que certos eventos ocorreram, em diferentes máquinas, para:

- saber se um evento E1 é anterior ou posterior a outro evento E2
- reconstituir uma sequência ordenada (rollback, auditoria)
- saber se dois eventos são simultâneos
- Execução *time-triggered*

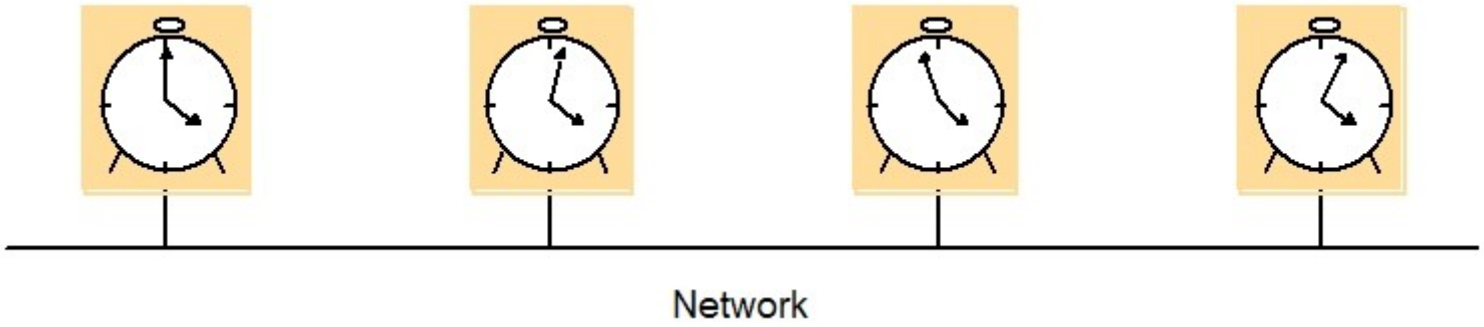
Introdução

- um SD é constituído por, entre outros, por um conjunto de processos
- os processos estão em execução em máquinas diferentes, que não partilham memória nem processador (nem relógio)
- estado do processo: cada processo tem um estado s_i , que eventualmente vai sendo atualizado com o tempo
- operações do processo
  - *send*: enviar mensagem
  - *receive*: receber mensagem
  - *transform*: aterar o estado do processo
- evento: ocorrência de uma ação do processo (comunicação ou transformação)
- histórico do processo: sequência de eventos do processo, ordenadas pelo tempo de ocorrência (**notação:** e1->e2 == e1 *precede* e2)
- a ordenação pode fazer-se... mas atribuir um *timestamp* para o evento não é tão simples!!!

Desvio na hora das máquinas do SD

Cada máquina tem um relógio

- a leitura do tempo pode dar valores diferentes em cada máquina



Relógios

skew: diferença instantânea na leitura de dois relógios

clock drift: desvio resultante da contagem do tempo usando frequências (1/período) diferentes. Para um mesmo relógio, a frequência pode variar com a temperatura...

drift rate: taxa que mede a variação do offset ou desvio da leitura de um relógio face a uma referência horária perfeita, por unidade de tempo medida pelo relógio de referência

- para relógios usuais com cristal de quartzo: 10^{-6}s/segundo (1 em cada 1000000)
- relógios de alta precisão (cristal de quartzo): 10^{-7} ou 10^{-8}

relógios atómicos: usam osciladores atómicos

- drift rate: 10^{-13}
- usados para medir o tempo decorrido, designado *International Atomic Time* (IAT)

Relógios e UTC

segundos, meses e anos

- unidades pensadas em função de fenómenos astonómicos: rotação e translação da Terra. Estes fenómenos não têm sempre a mesma duração, devido a diversos factores

Consequência: IAT e tempo astronómico tendem a divergir

Coordinated Universal Time

- norma para contagem do tempo
- baseada no IAT mas adiciona ou retira um segundo, ocasionalmente, para acertar com o tempo astronómico
- sinais de UTC são enviados em broadcast desde estações terrestres ou satélites (GPS)

Sincronização de Relógios

- um computador pode acertar o seu relógio utilizando um receptor para captar os sinais UTC, GPS

- em alternativa, um computador pode receber o tempo oficial através de uma linha telefónica ou da rede, a partir de organizações como
  - National Institute for Standards and Technology, USA
  - Observatório Astronómico de Lisboa, Portugal
    - ainda mais importante para serviços em que o *timestamp* tem um valor legal (leilões, serviços jurídicos, bolsa, ...)

Modos de Sincronização

- Modos de sincronização sobre um intervalo de tempo real I:
  1. interna: para um limite de sincronização D>0, |Ci(t) – Cj(t)| < D, para i,j=1,2... N (máquinas) e para todos os instantes t em I
  2. externa: para um limite de sincronização D>0, com uma fonte UTC S, |S(t)-Ci(t)|<D, para i=1,2...N e para todos os instantes t em I

- Relógios sincronizados de modo interno não estão necessariamente sincronizados de modo externo

- Se cada nó de um sistema está sincronizado de modo externo (com a mesma fonte) com limite D, então esse sistema está internamente sincronizado com um limite 2D

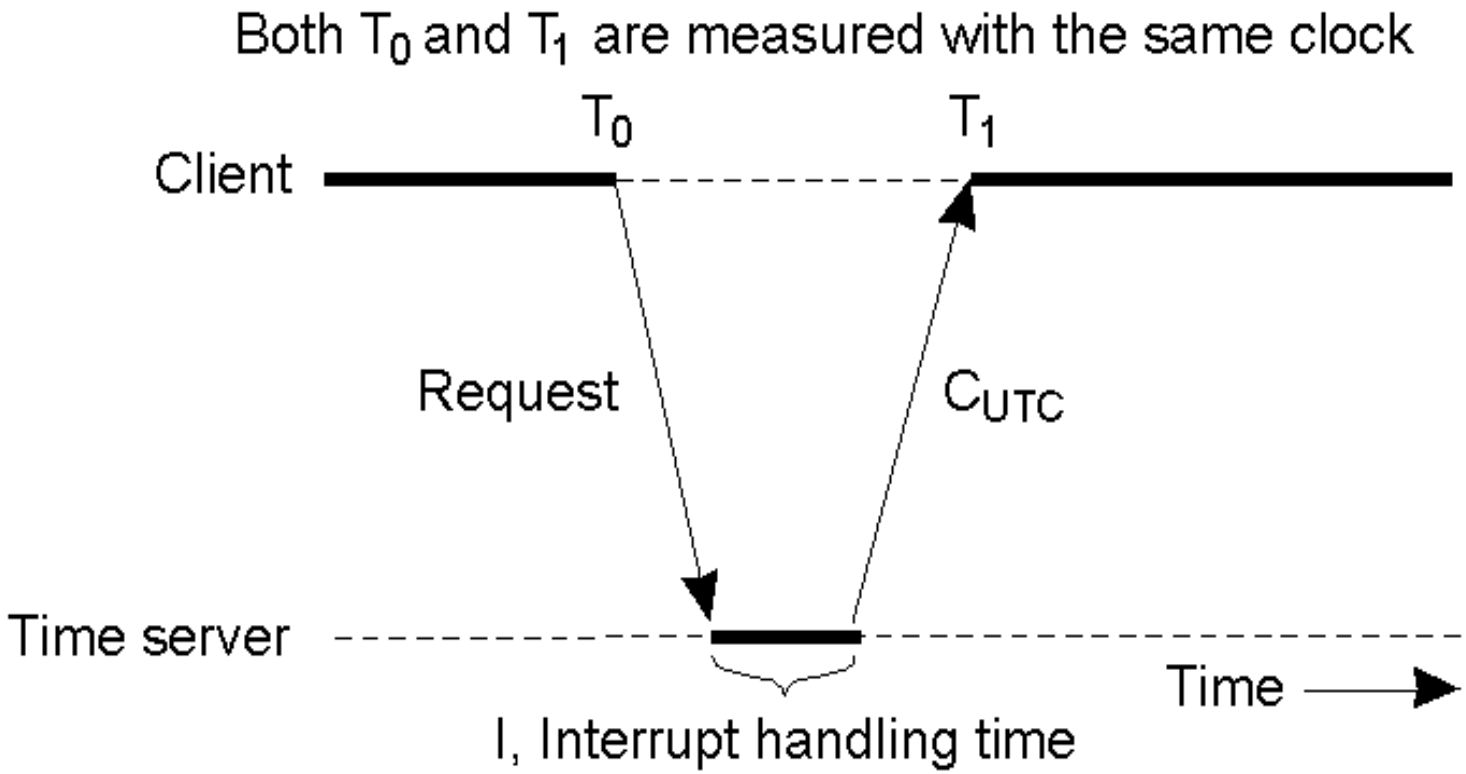
Algoritmos de Sincronização

- Algoritmo de Cristian
- Algoritmo de Berkeley
- Protocolo NTP

Algoritmo de Cristian

- probabilístico: sincroniza se o tempo para a troca de mensagens (um par) cliente-servidor é suficientemente pequeno quando comparado com a precisão desejada
- servidor de tempo UTC S
- processo p envia pedido m_r e recebe um tempo *t* em m_t
- p regista o tempo de viagem de m_r e m_t: $T = Tm_r + Tm_t$
- estimativa do tempo em p: $t + T/2$
  - assume o mesmo tempo para as duas mensagens (!)
    - razoável: excepto se enviadas por redes diferentes ou se ocorre um afunilamento repentino aquando da 2ª mensagem
- Opção para melhorar a precisão:
  - informação horária prestada por um grupo de servidores sincronizados
  - cliente faz multicast do pedido para o grupo de servidores e usa a primeira resposta obtida

Algoritmo de Cristian



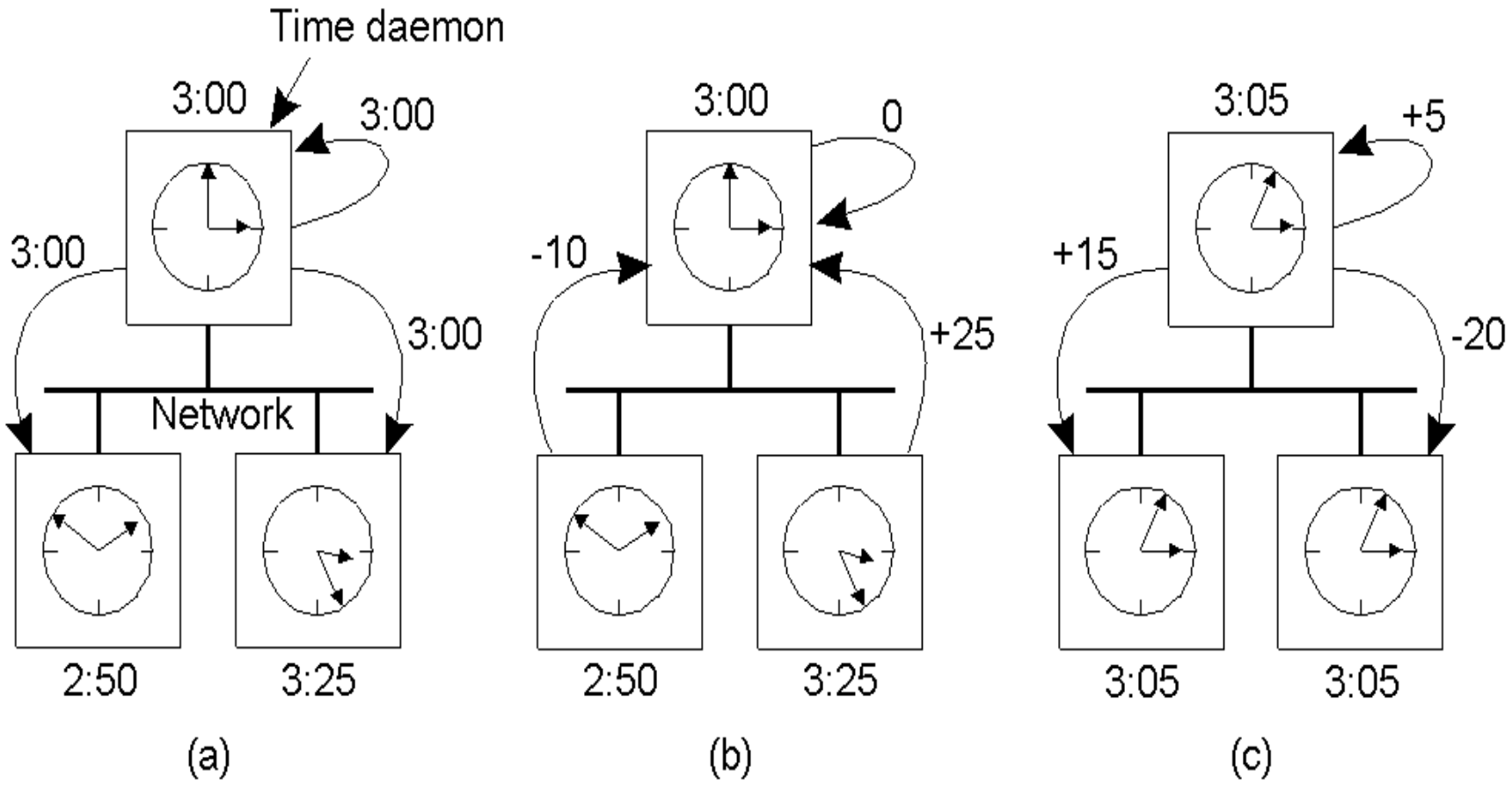
Algoritmo de Berkeley

para sincronização interna de um grupo de computadores

- uma máquina é escolhida para coordenar – master
- o *master* interage periodicamente com todos os elementos do grupo (*slaves e* ele próprio), para saber a diferença relativamente a cada um
- master estima a hora em cada slave, pela observação do tempo de viagem das mensagens e pelo valor recebido (como em Cristian)
- faz a média de todos os valores (incluindo o seu tempo)
- em vez de enviar o tempo atualizado aos slaves (o que estaria sujeito ao tempo de envio variável), o master envia a cada um o valor exato que este deve usar para ajustar o seu relógio

- se o master falhar, outro será escolhido para assumir a sua função

Algoritmo de Berkeley



Network Time Protocol (**NTP**)

protocolo para distribuir informação horária sobre a Internet (RFC 1305)

Funcionalidades:

- permite a sincronização precisa de clientes, que se encontram espalhados pela Internet, com UTC
- serviço fiável que resiste a perdas de conectividade
- escalável em termos de nº de clientes e nº de servidores
- proteção contra interferência (acidental ou maliciosa) na informação horária

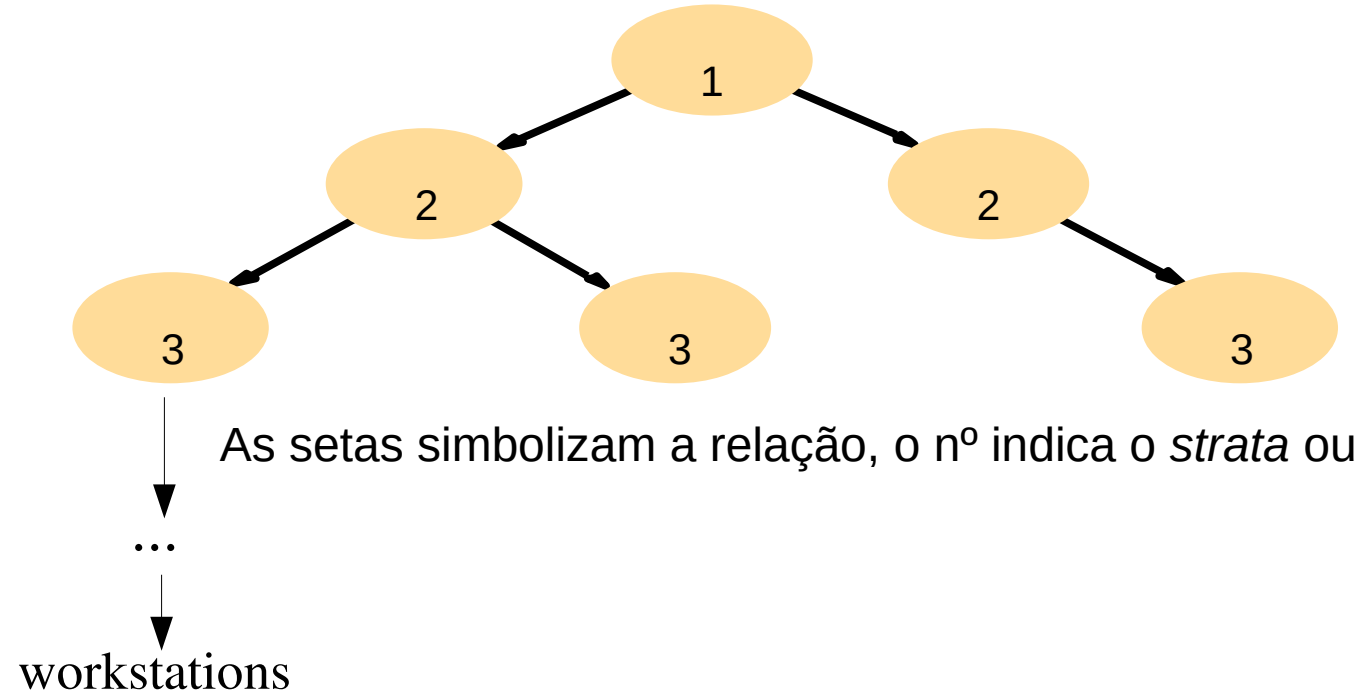
Serviço suportado por uma rede de servidores na Internet:

- primários: ligados diretamente a uma fonte de UTC
- secundários: sincronizados com a informação dos primários

Sincronização em Rede com **NTP**

os níveis superiores têm:

- *strata* inferior (1º)
- informação horária mais precisa



Sincronização em Rede com **NTP**

os Servidores NTP sincronizam-se com um dos modos:

- multicast
  - usa-se em redes locais de alta velocidade. Um ou mais servidores enviam periodicamente o tempo num broadcast. Os servidores noutras máquinas da rede acertam o seu relógio assumindo um pequeno delay. Baixa precisão.
- procedure-call
  - um servidor aceita pedidos de outros computadores, aos quais responde com a informação horária que tem (como Cristian). Utilizado quando se pretende maior precisão que no modo multicast, ou simplesmente não é possível multicast.
- modo simétrico
  - para sincronização entre servidores que fornecem a informação em redes locais e em níveis mais altos da NTP subnet, onde se pretende a máxima precisão.
  - Um par de servidores trabalha de modo simétrico, troca mensagens com informação horária. O tempo das mensagens também é considerado.

Sincronização em Rede com **NTP**

em todos os modos

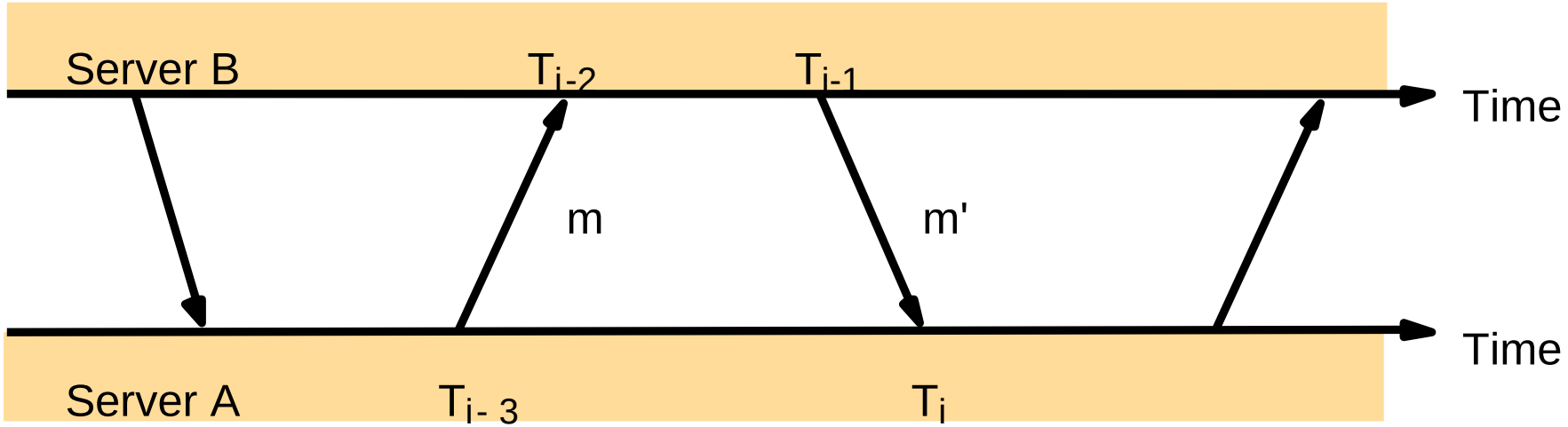
- mensagens trocadas no protocolo de transporte UDP

Ainda o modo simétrico

- os servidores trocam pares de mensagens
- cada mensagem tem *timestamps* com:
  - hora local em que a última mensagem deste par de servidores foi enviada e recebida e a hora local a que a atual mensagem é transmitida.
- o servidor que recebe a mensagem anota a hora a que a recebe
- Ficam 4 tempos: Ti-3, Ti-2, Ti-1, Ti
- pode haver atrasos significativos entre a chegada de uma mensagem e o envio da seguinte, isso não será problema

Sincronização em Rede com **NTP**

Mensagens trocadas entre um par de servidores NTP (modo simétrico)



Para cada par de mensagens, os servidores calculam

- o_i: estimativa do *offset* entre os dois relógios
- d_i: delay, tempo total de transmissão das duas mensagens

Sincronização em Rede com **NTP**

Supondo que o *offset* efetivo entre os relógios A e B é **o**, e que os tempos reais de transmissão de m e m' foram t e t', respectivamente:

- $T_{i-2} = T_{i-3} + t + o$ e ainda $T_i = T_{i-1} + t' - o$

E então:

- $d_i = t + t' = T_{i-2} - T_{i-3} + T_i - T_{i-1}$

E ainda:

- $o = o_i + (t' - t) / 2$

  onde $o_i = (T_{i-2} - T_{i-3} - T_i + T_{i-1}) / 2$
- t e $t' >= 0$
- $o_i - d_i/2 <= o <= o_i + d_i/2$

Portanto: o_i é uma estimativa e d_i uma medida da precisão dessa estimativa

Sincronização em Rede com **NTP**

os servidores fazem uma filtragem aos sucessivos pares de valores (oi , di)

- dos últimos 8 pares de valores, escolhe-se o oi que corresponde ao menor di

O *offset* calculado relativamente a um único servidor não é necessariamente usado para controlar o relógio local. Em geral um servidor NTP troca mensagens com **vários** outros servidores.

Para além de filtrar os valores com pouca precisão de um servidor, o algoritmo pode também filtrar os servidores com quem se estabelecem as trocas de mensagens, caso se demonstrem pouco fiáveis

O protocolo NTP é usado para acertar o relógio local e mais...

- ao descobrir-se a drift rate de um relógio (ex: ganhar 1 seg/hora), a sua frequência pode ser ajustada por hardware ou software para reduzir o nº de acertos futuros

...

NOTA: Noções de Concorrência em Java:

- **Threads** *(link no moodle)*

Concorrência

- Pode levar a resultados inconsistentes

- Género de concorrência:
  - **Local**: várias Threads em execução num mesmo processo
  - **Distribuída**: execução de vários processos (possivelmente em diferentes máquinas), operando sobre recursos partilhados

- Necessidade
  - Mecanismo para garantir exclusão mútua a nível do SD
    - Baseia-se em troca de mensagens

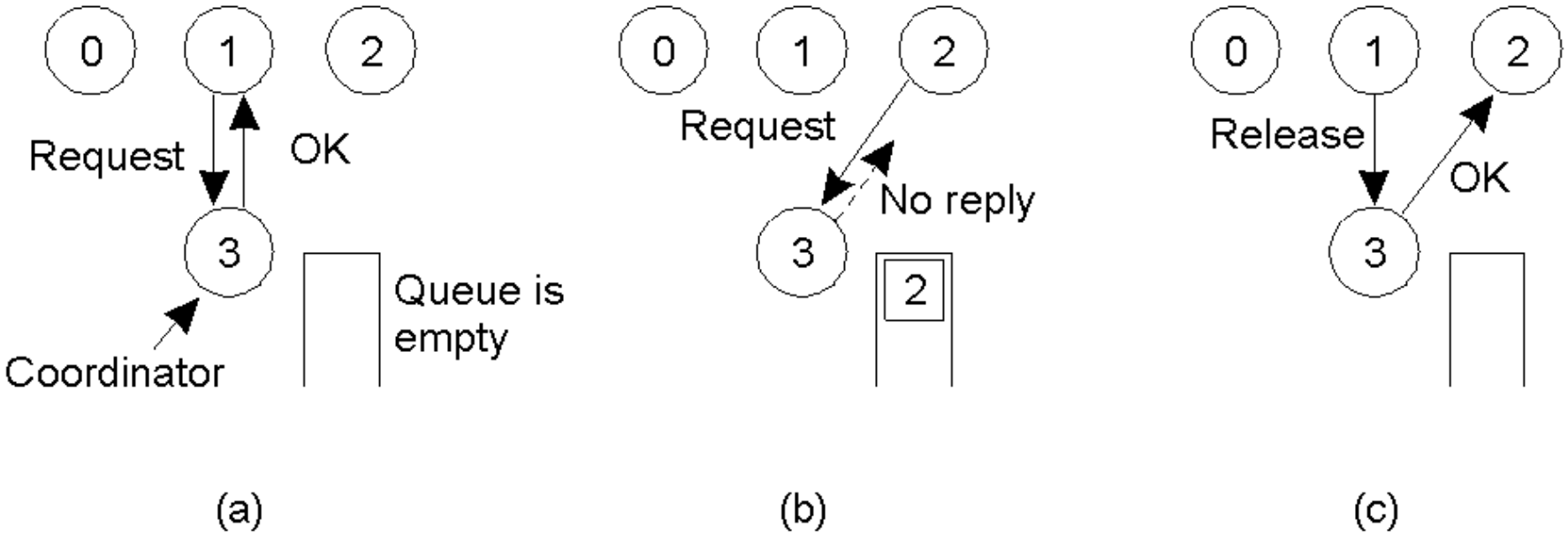
Exclusão mútua a nível distribuído

- Requisitos desejáveis
  - *Safety*: no máximo, pode haver 1 processo na zona crítica, a qualquer momento
  - *Liveness*: os pedidos para cesso à ZC terão uma resposta
    - Evitar deadlock; evitar starvation
  - *Ordering*: se o pedido de P1 para entrar na ZC chegou primeiro que P2, então a autorização para entrar deve ser dada em primeiro a P1

- Aspetos a considerar na avaliação do desempenho do algoritmo
  - Largura de banda consumida
  - Delay para o cliente, nos pedidos de entrada (e saída) da ZC
  - Efeito do algoritmo no *throughput* geral do sistema distribuído

Exclusão Mútua

- Algoritmo Centralizado

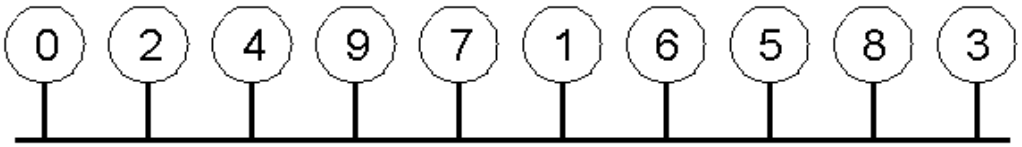


Um dos processos coordena o acesso à zona crítica

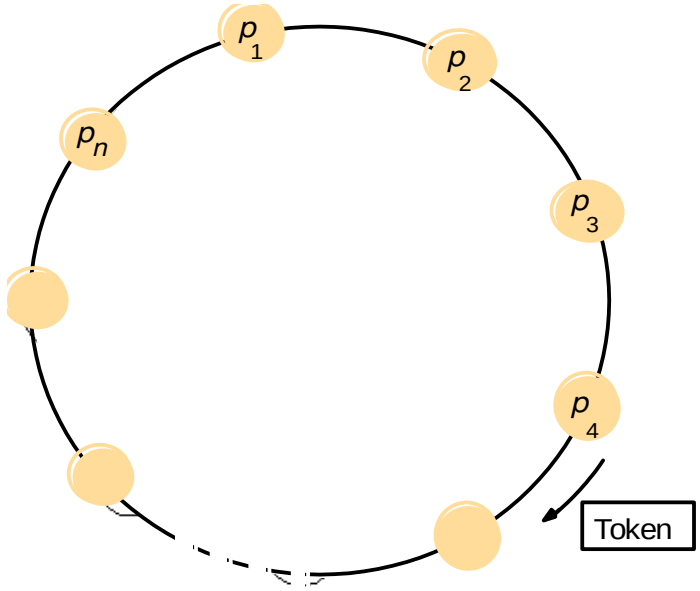
Os processos interessados, pedem acesso ao coordenador, que mantem uma fila de espera para aquela zona de exclusão

Exclusão Mútua

- Algoritmo de Coordenação em Anel



(a)



(b)

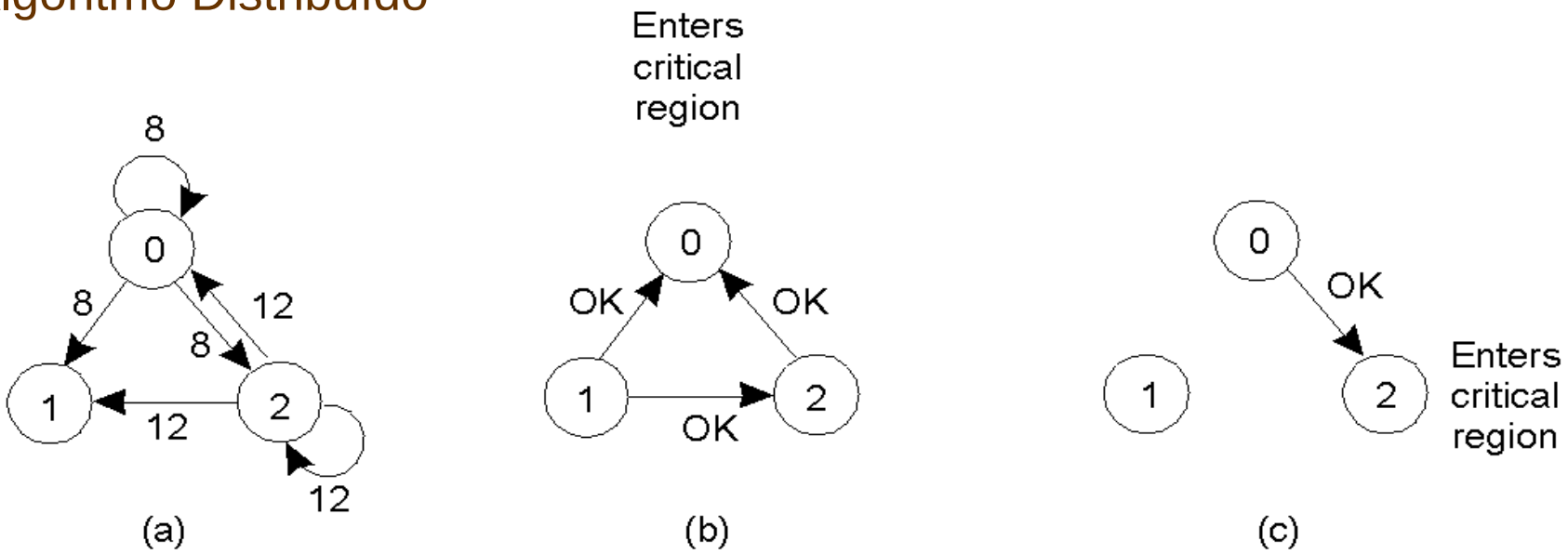
Os processos estabelecem ligações lógicas em forma de anel e passam um ***token***

O **detentor** do *token* pode aceder à **zona de exclusão**.

Se não pretender entrar, passa de imediado o *token*.

Exclusão Mútua

- Algoritmo Distribuído



Os processos estão a par dos outros processos e das intenções deles

Cada interessado em entrar na ZC faz multicast para todos os outros, incluindo um timestamp na mensagem

Os receptores respondem a confirmar que pela sua parte pode ser (OK). Caso haja outro interessado (P2), este vai adiar a sua entrada se o seu timestamp é posterior ao outro (P0) – e nesse caso P0 irá dar-lhe o OK após terminar a execução na ZC.

Exclusão Mútua

- Análise dos algoritmos

Algoritmo	**Mensagens para entrar e sair**	**Delay até entrar (em nº de mensagens)**	**Problemas**
Centralizado	3	2	Crash do coordenator
Distribuído	2 (n – 1)	2 (n – 1)	Crash de qualquer processo
Coord. Em Anel	1 to ∞	0 to n – 1	Falha de omissão na passagem do *token* Falha em qualquer processo

Consenso em SD

- Se vários processos tiverem de chegar a consenso sobre um valor
  - Como coordenar a negociação?

- Cada processo *Pi* começa com um estado de indefinição e propõe um valor *Vi, existindo uma lista de regras para apreciação das propostas Vi*
  - Comunicam uns com os outros, trocando os valores propostos
  - Cada *Pi* fica com o valor *di* em função de cada decisão
    - São aplicadas as regras (exemplo: escolher o maior, ou o que tiver id mais baixo)
  - Incrementalmente, a decisão vai sendo tomada entre todos

- Requisitos
  - O processo termina
  - Acordo
  - Integridade