BCC 362 – Sistemas Distribuídos

Joubert de Castro Lima – joubertlima@gmail.com
Professor Adjunto – DECOM

UFOP

Figuras e textos retirados do livro:

Sistemas Distribuídos do Tanenbaum

INTRODUÇÃO - SINCRONIZAÇÃO

Examples of synchronization: It is important that multiple processes do not simultaneously access a shared resource, such as printer, but instead cooperate in granting each other temporary exclusive access.

Another example, is that multiple processes ma sometimes need to agree on the ordering of events, such as whether message *m1* from process *P* was sent before or after message *m2* from process *O*.

.CLOCK SYNCHRONIZATION

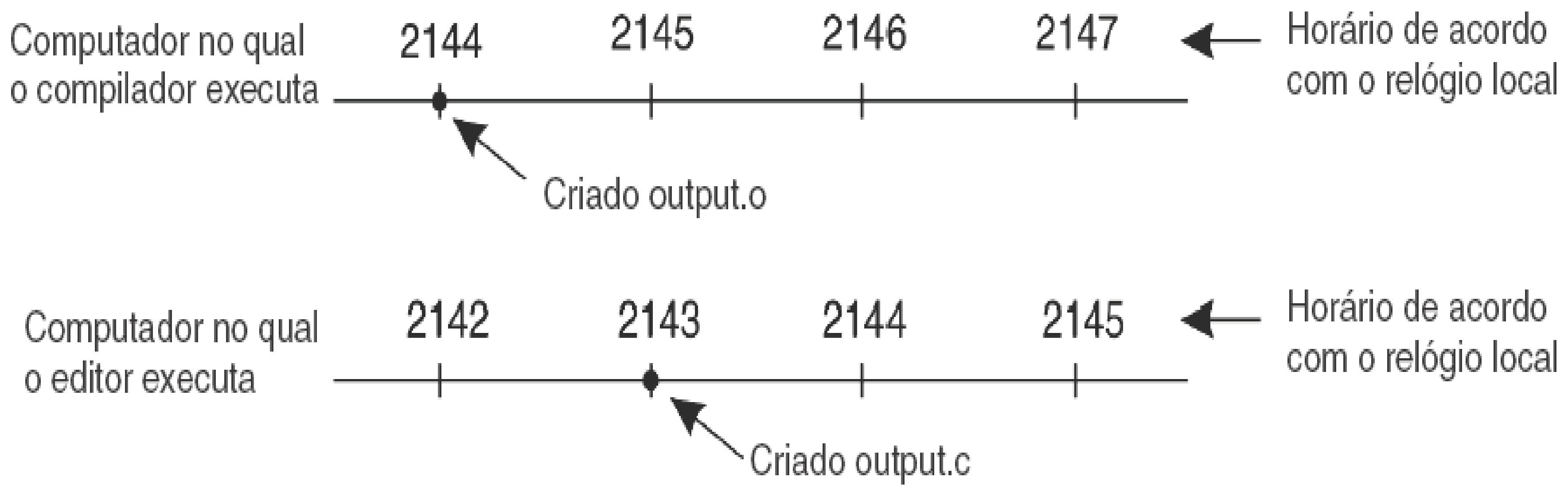

Computador no qual o compilador executa
2144
2145
2146
2147
Horário de acordo com o relógio local
Criado output.o
Computador no qual o editor executa
2142
2143
2144
2145
Horário de acordo com o relógio local
Criado output.c

Figura 6.1 Quando cada máquina tem seu próprio relógio, um evento que ocorreu após outro evento pode, ainda assim, receber um horário anterior.

Example of *make* program call.

** Basic concepts

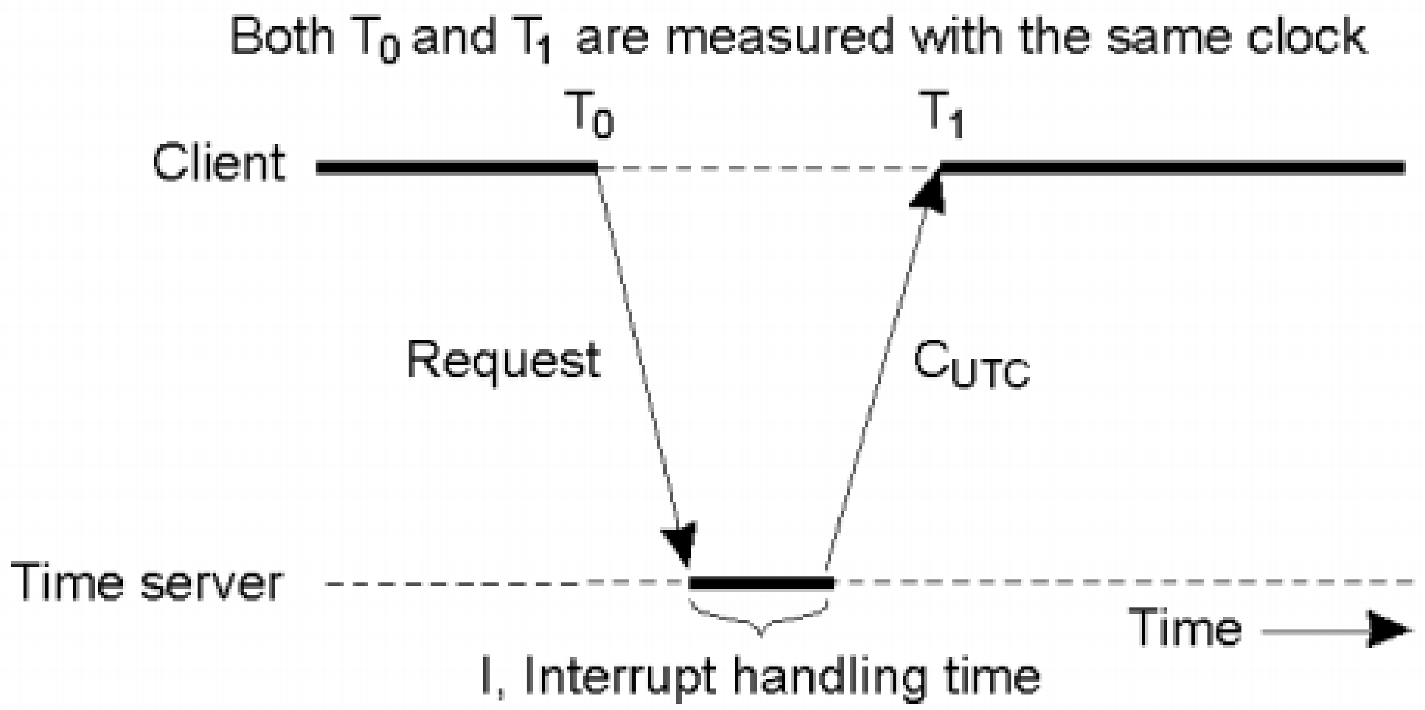
Solar time – The interval between two consecutive transit of the sun. Problem, this time is not exact.

TAI – International Atomic Time. It is the exact universal time. Problem, it is not synchronized with our concept of time.

UTC – Universal Coordinate Time. It is used to correct the our concept of time considering the TAI.

To provide UTC to people who need precise time, shortwave radio station broadcast a short pulse at the each UTC

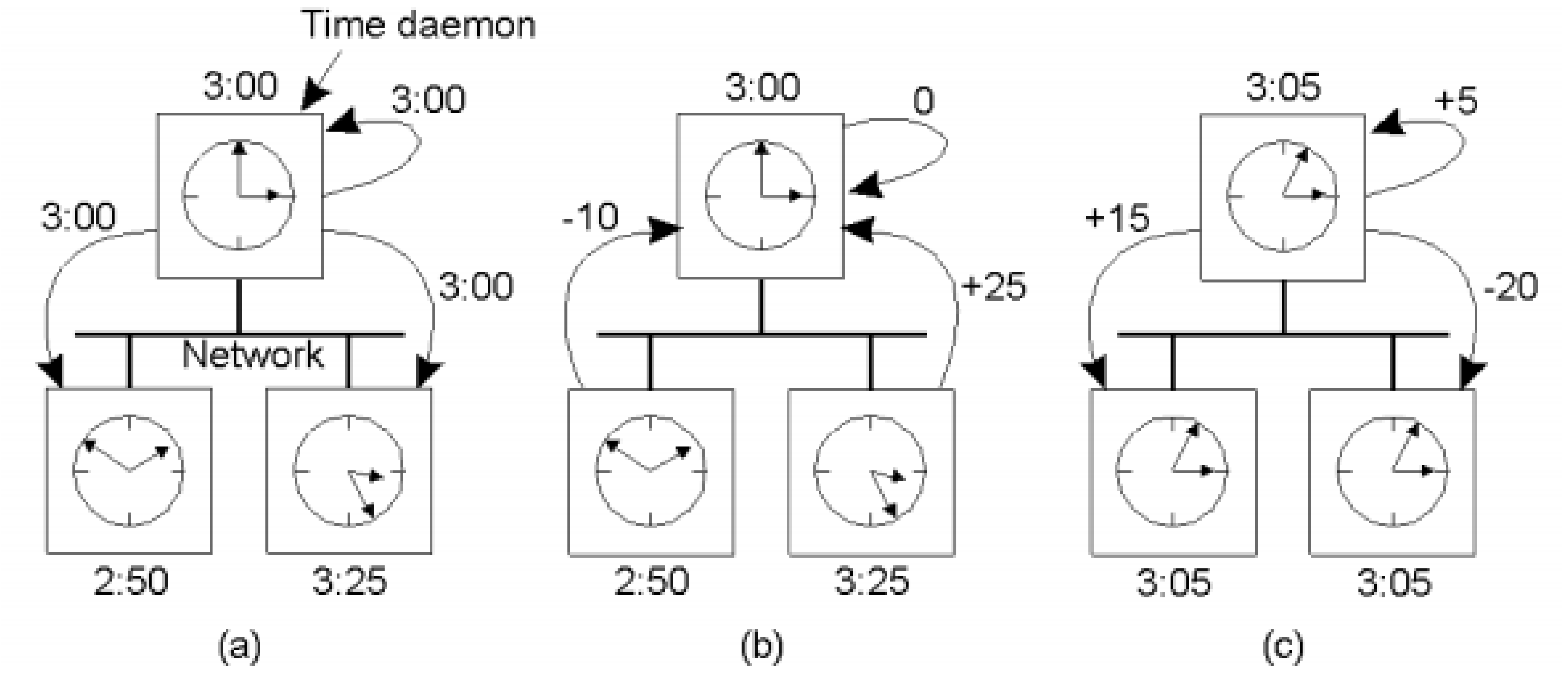
Cristian's Algorithm



Both T_0 and T_1 are measured with the same clock
T_0
T_1
Client
Request
C_{UTC}
Time server
Time
I, Interrupt handling time

Getting the current time from a time server. This algorithm considers that there is one machine synchronized with UTC

The Berkeley Algorithm

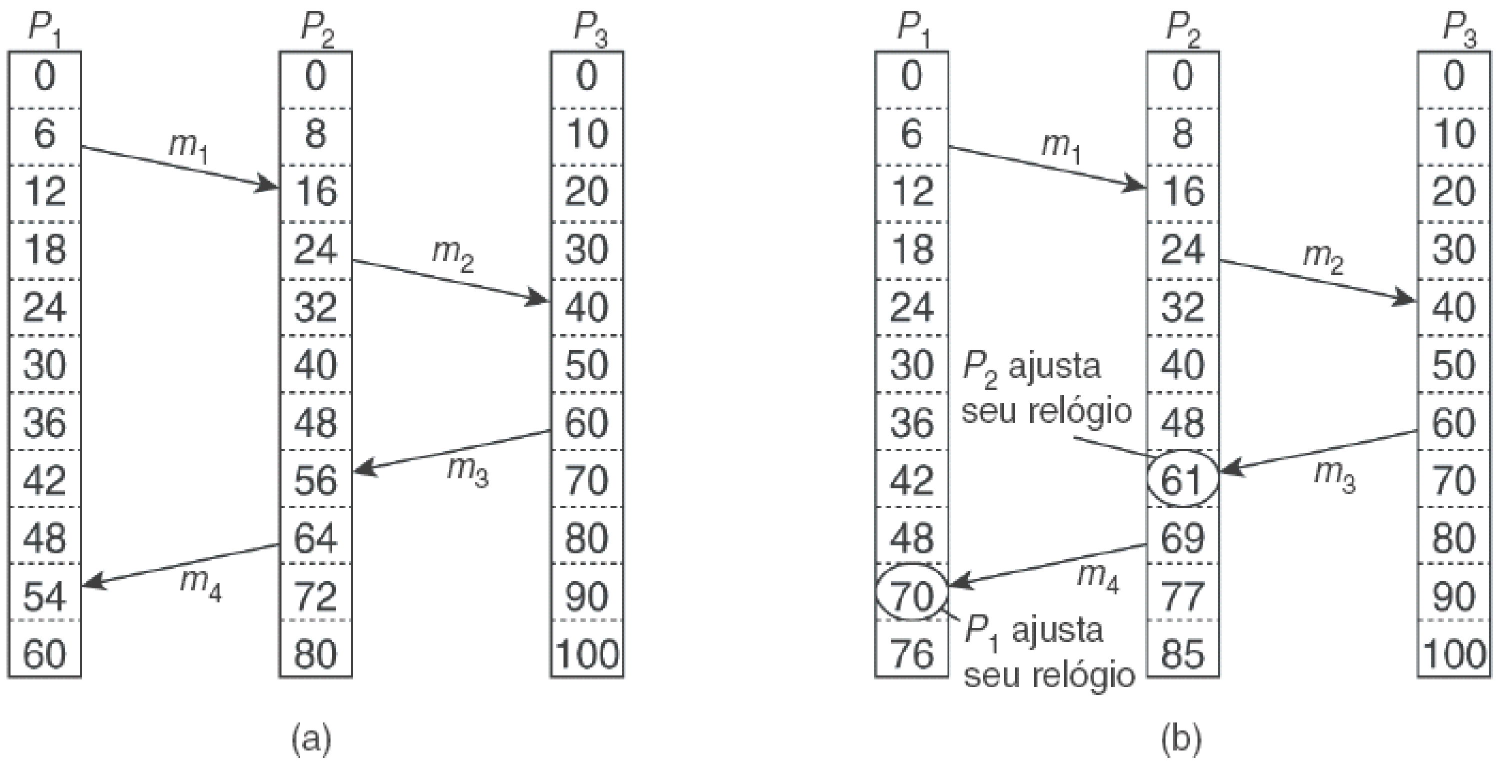


a) The time daemon asks all the other machines for their clock values
b) The machines answer
c) The time daemon tells everyone how to adjust their clock

No machine is synchronized with UTC.

- Logical CLOCKS

•Lamport Timestamps



P_1
P_2
P_3
m_1
m_2
m_3
m_4
P1: 0 6 12 18 24 30 36 42 48 54 60
P2: 0 8 16 24 32 40 48 56 64 72 80
P3: 0 10 20 30 40 50 60 70 80 90 100
(a)
P1: 0 6 12 18 24 30 36 42 48 70 76
P2: 0 8 16 24 32 40 48 61 69 77 85
P3: 0 10 20 30 40 50 60 70 80 90 100
P_2 ajusta seu relógio
P_1 ajusta seu relógio
(b)

Figura 6.9 (a) Três processos, cada um com seu próprio relógio. Os relógios funcionam a taxas diferentes. (b) O algoritmo de Lamport corrige os relógios.

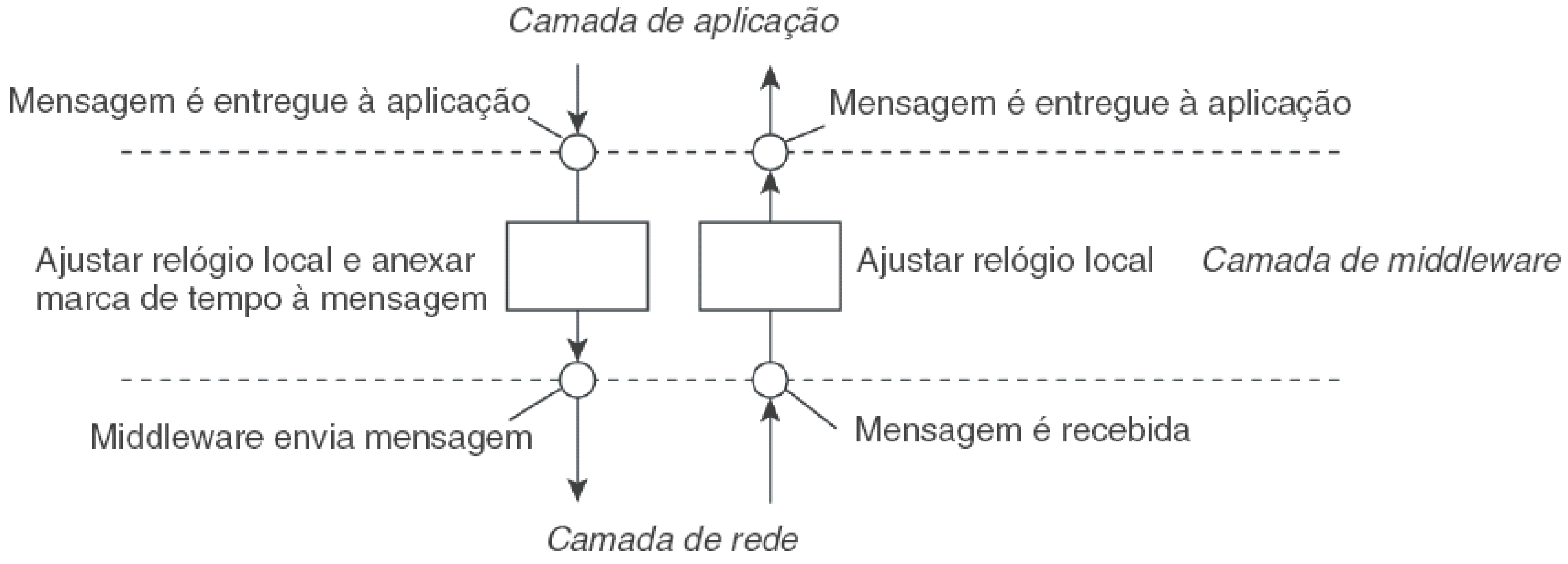

Camada de aplicação
Mensagem é entregue à aplicação
Mensagem é entregue à aplicação
Ajustar relógio local e anexar marca de tempo à mensagem
Ajustar relógio local
Camada de middleware
Middleware envia mensagem
Mensagem é recebida
Camada de rede

Figura 6.10 Posicionamento de relógios lógicos de Lamport em sistemas distribuídos.

RELÓGIOS VETORIAIS (causalidade)

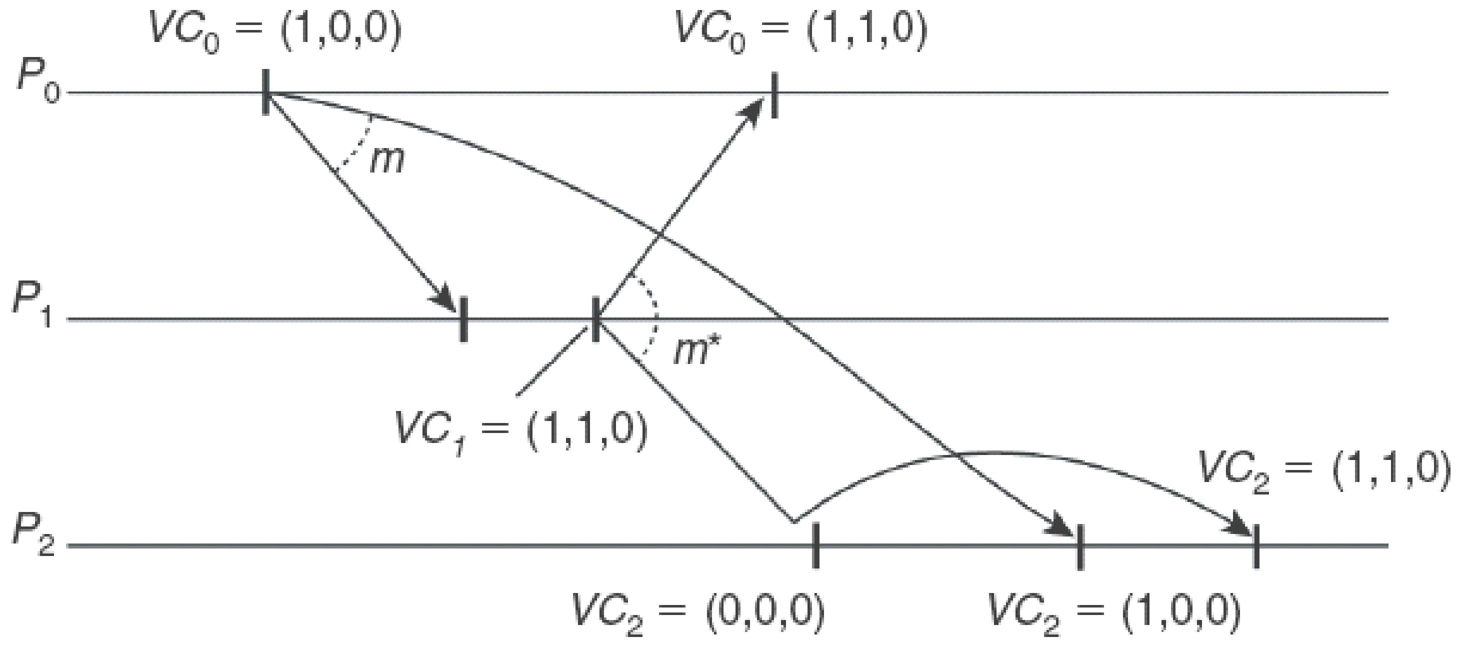
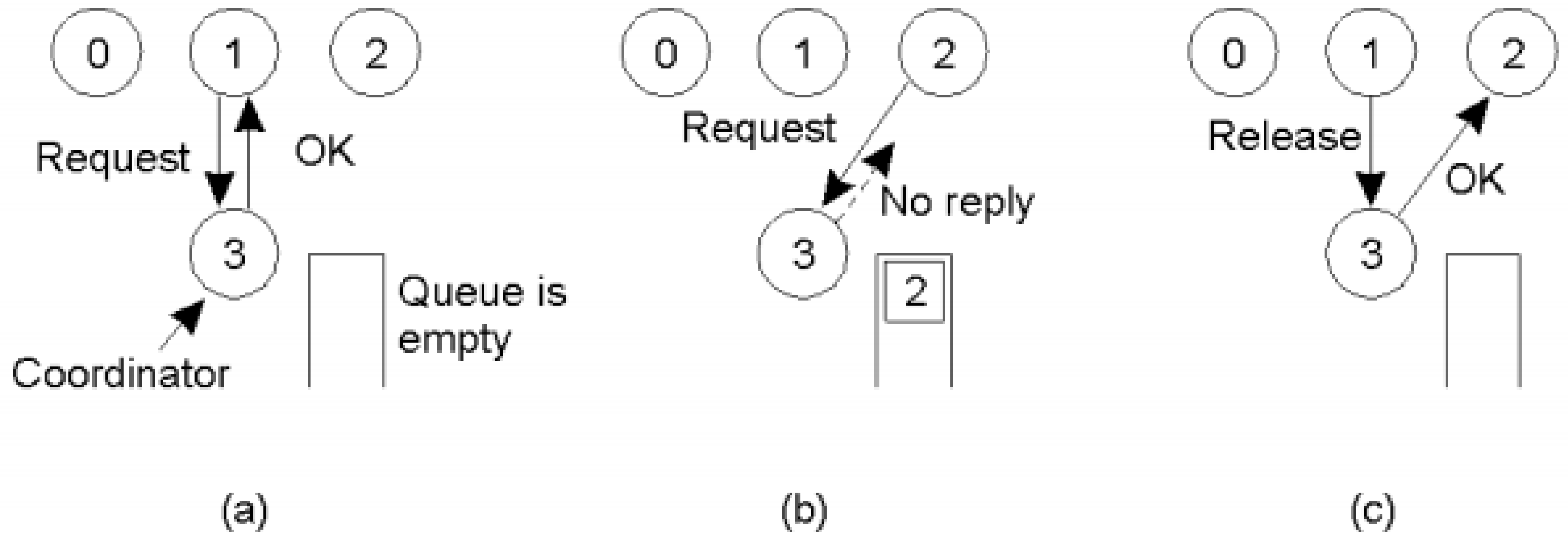


Figura 6.13 Imposição de comunicação causal.

Mutual exclusion

Critical regions remarks

A Centralized Algorithm

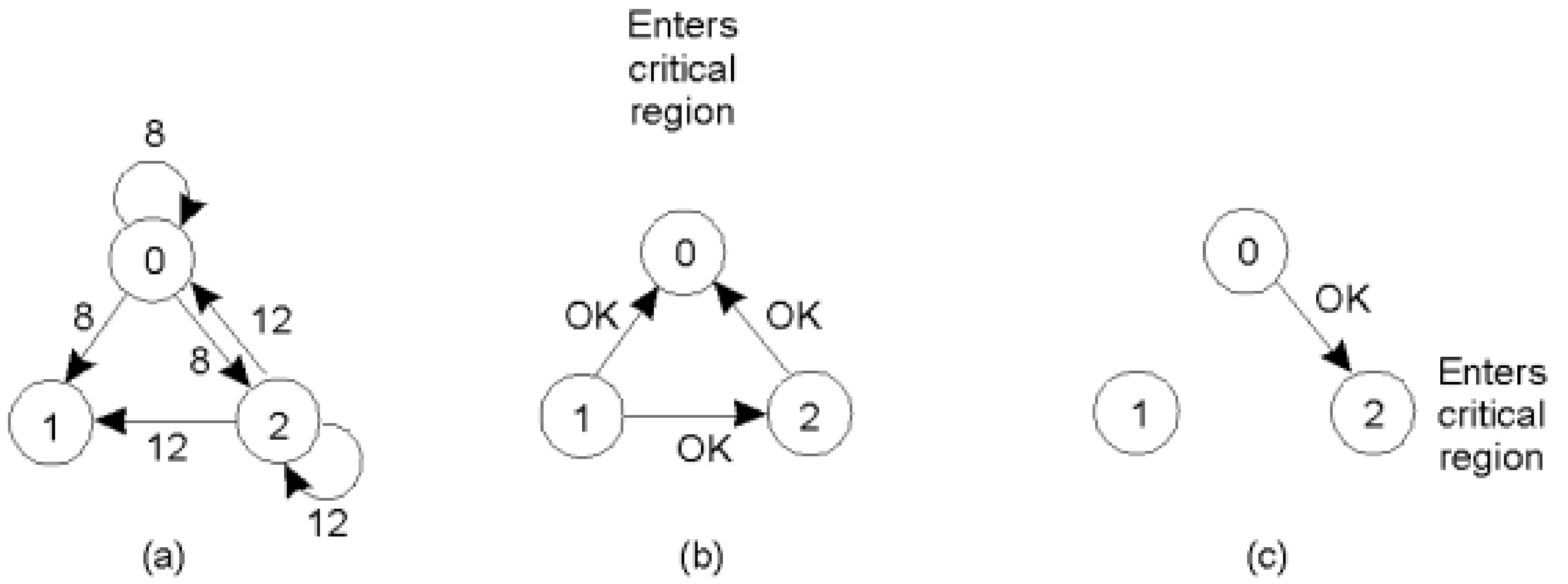


a) Process 1 asks the coordinator for permission to enter a critical region. Permission is granted

b) Process 2 then asks permission to enter the same critical region. The coordinator does not reply.

c) When process 1 exits the critical region, it tells the coordinator, when then replies to 2

A Distributed Algorithm

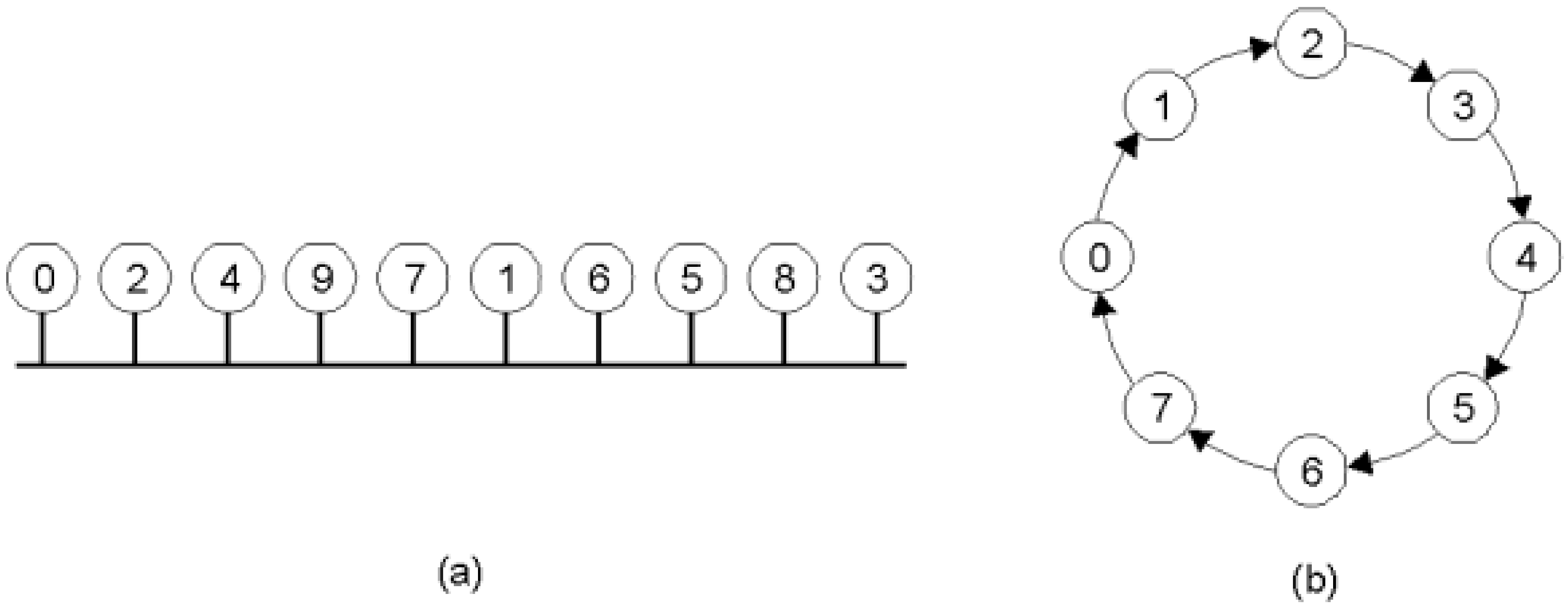


a) Two processes want to enter the same critical region at the same moment.

b) Process 0 has the lowest timestamp, so it wins.

c) When process 0 is done, it sends an OK also, so 2 can now enter the critical region.

A Token Ring Algorithm



a) An unordered group of processes on a network.
b) A logical ring constructed in software.

Algoritmo	Mensagens por entrada/saída	Atraso antes da entrada (em número de tempos de mensagens)	Problemas
Centralizado	3	2	Queda do coordenador
Descentralizado	$3mk$, $k = 1,2,...$	$2\ m$	Inanição, baixa eficiência
Distribuído	$2\ (n - 1)$	$2\ (n - 1)$	Queda de qualquer processo
Token Ring	1 a ∞	0 a $n - 1$	Ficha perdida; processo cai

Tabela 6.1 Comparação entre quatro algoritmos de exclusão mútua.

- Election Algorithms

ALGORITMO DO VALENTÃO

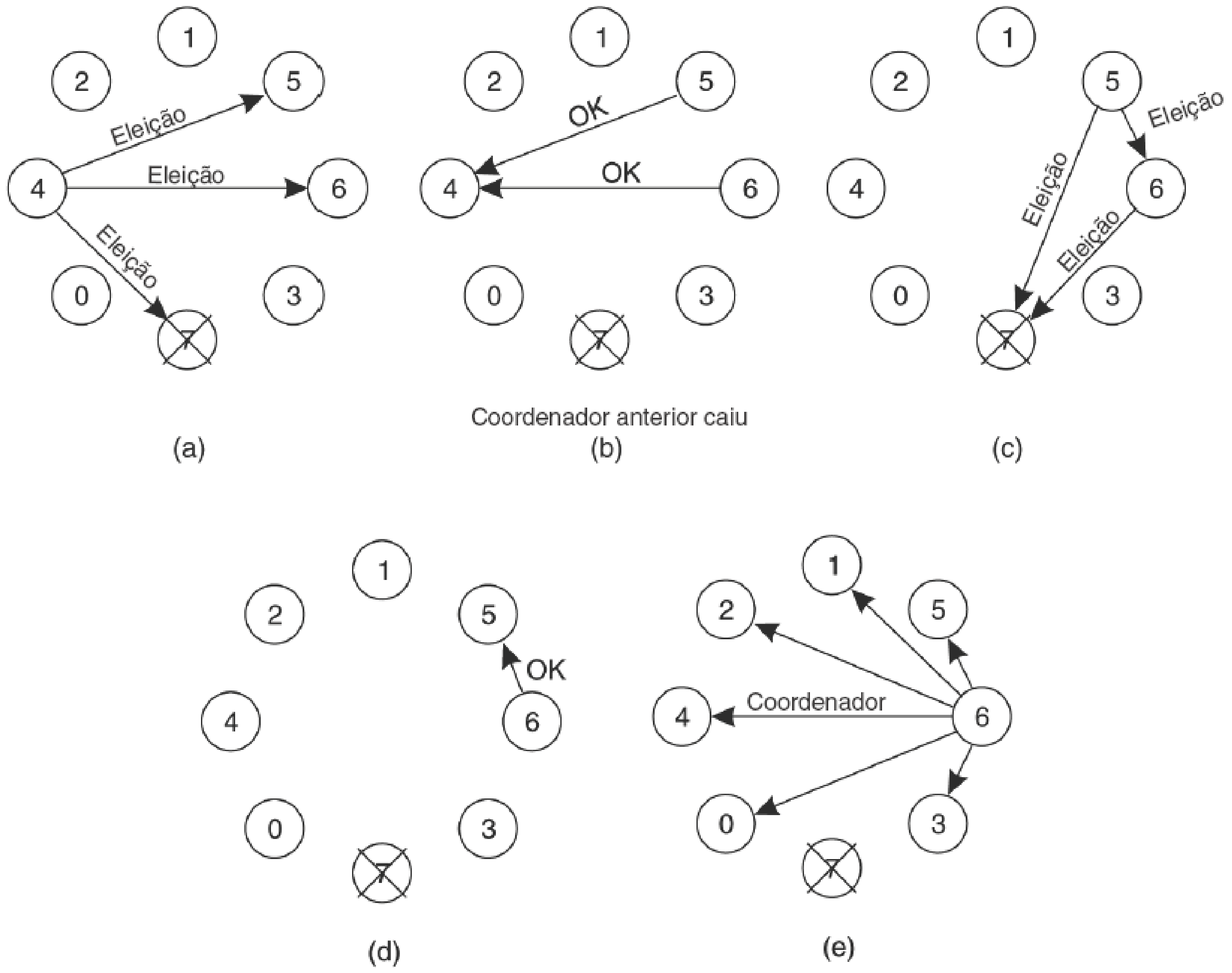


Figura 6.19 Algoritmo de eleição do valentão. (a) O processo 4 convoca uma eleição. (b) Os processos 5 e 6 respondem e mandam 4 parar. (c) Agora, cada um, 5 e 6, convoca uma eleição. (d) O processo 6 manda 5 parar. (e) O processo 6 vence e informa a todos.

ALGORITMO DE ANEL

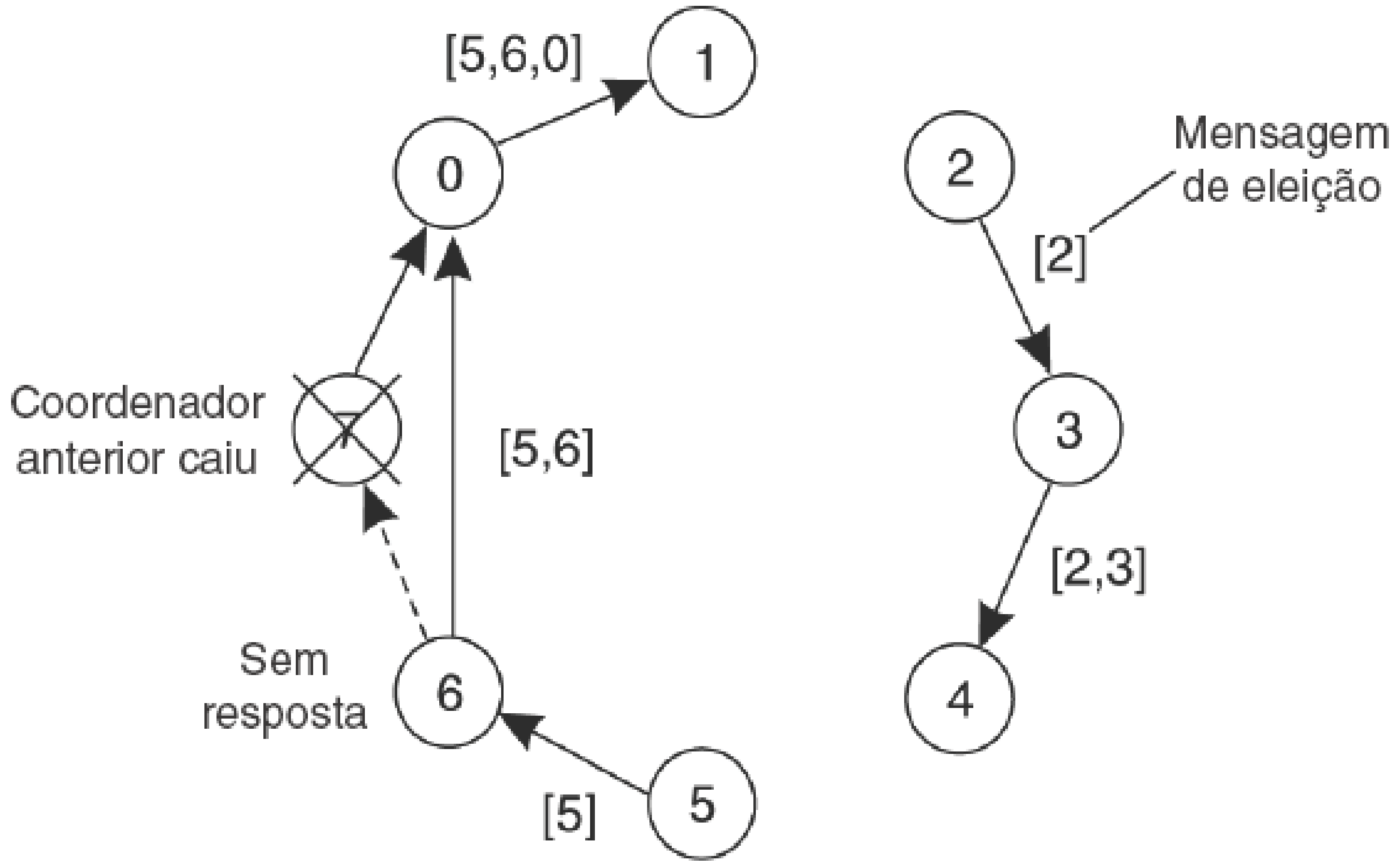


Figura 6.20 Algoritmo de eleição que usa um anel.