

DISTRIBUTED SYSTEMS
Principles and Paradigms
Second Edition
ANDREW S. TANENBAUM
MAARTEN VAN STEEN

Chapter 6 Synchronization

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

1

Clock Synchronization

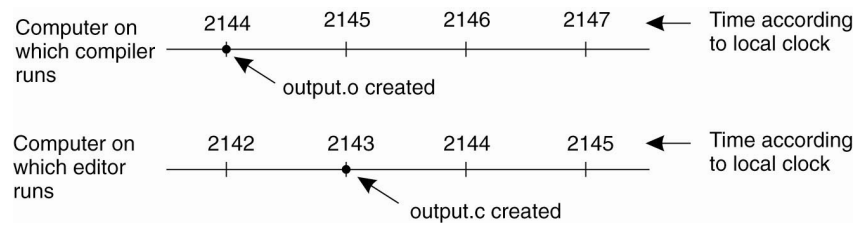


Figure 6-1. When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

2

Physical Clocks (1)

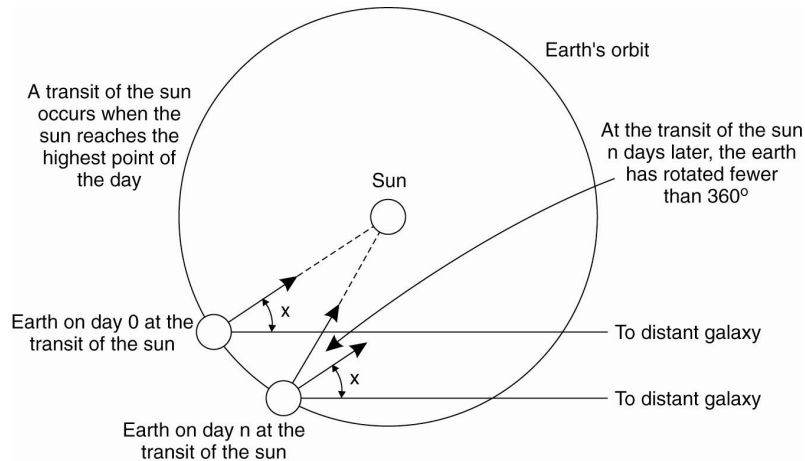


Figure 6-2. Computation of the mean solar day.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

3

Physical Clocks (2)

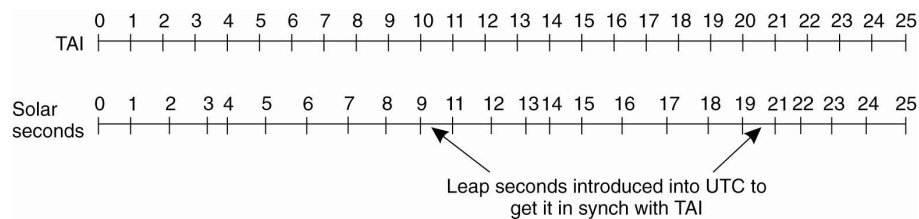


Figure 6-3. TAI seconds are of constant length, unlike solar seconds. Leap seconds are introduced when necessary to keep in phase with the sun.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

4

Global Positioning System (1)

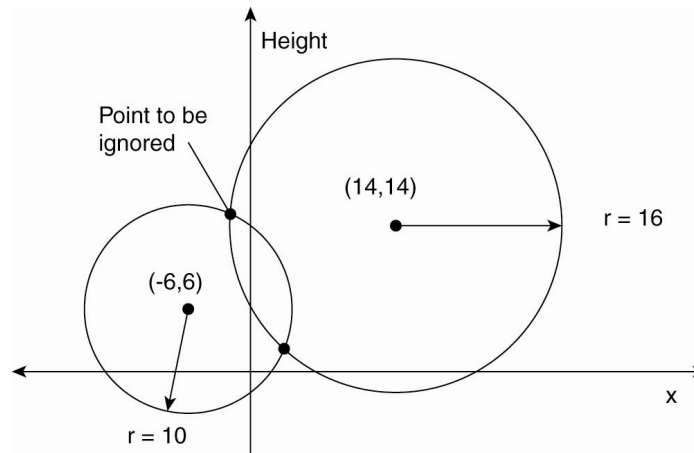


Figure 6-4. Computing a position in a two-dimensional space.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

5

Global Positioning System (2)

Real world facts that complicate GPS

1. It takes a while before data on a satellite's position reaches the receiver.
2. The receiver's clock is generally not in synch with that of a satellite.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

6

Clock Synchronization Algorithms

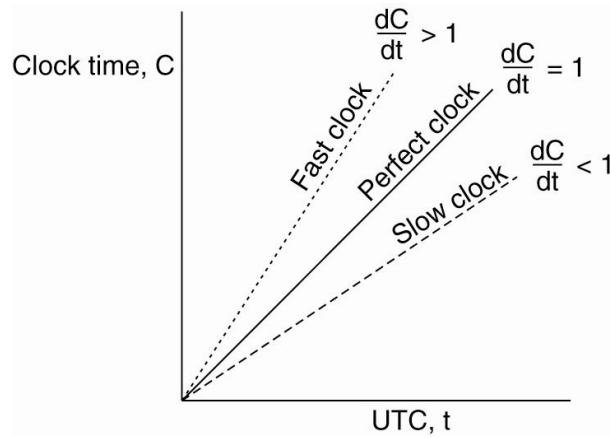


Figure 6-5. The relation between clock time and UTC when clocks tick at different rates.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

7

Network Time Protocol

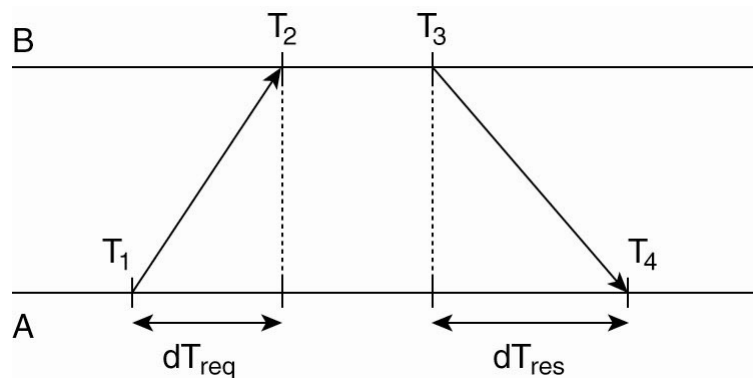


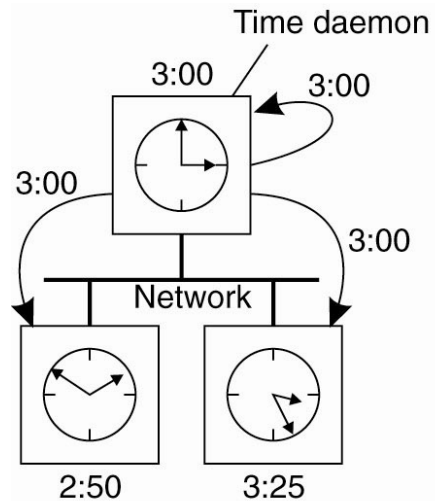
Figure 6-6. Getting the current time from a time server.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

8

The Berkeley Algorithm (1)

Figure 6-7. (a) The time daemon asks all the other machines for their clock values.



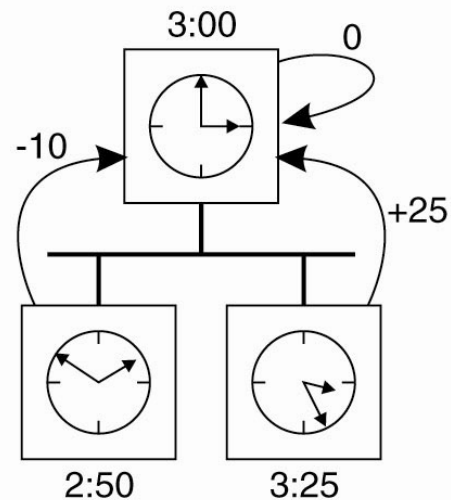
(a)

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

9

The Berkeley Algorithm (2)

Figure 6-7.
(b) The machines answer.



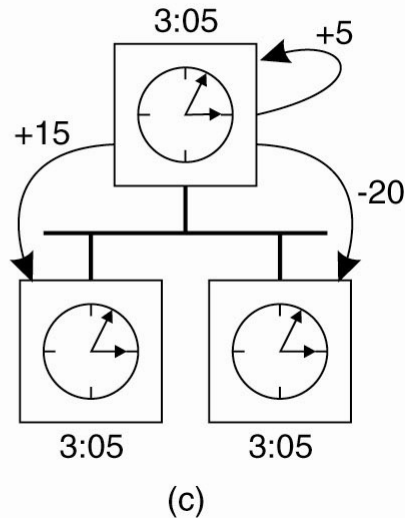
(b)

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

10

The Berkeley Algorithm (3)

Figure 6-7. (c) The time daemon tells everyone how to adjust their clock.

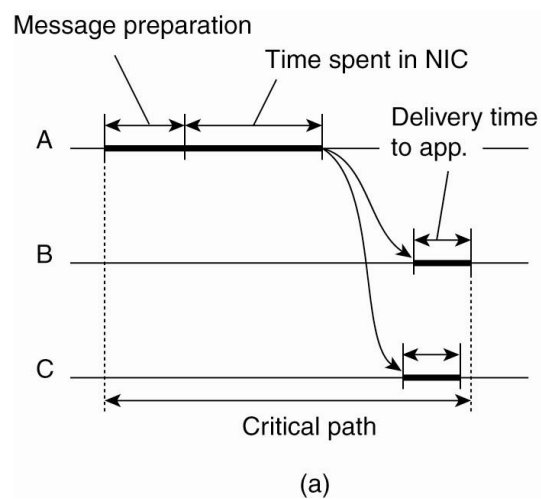


Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

11

Clock Synchronization in Wireless Networks (1)

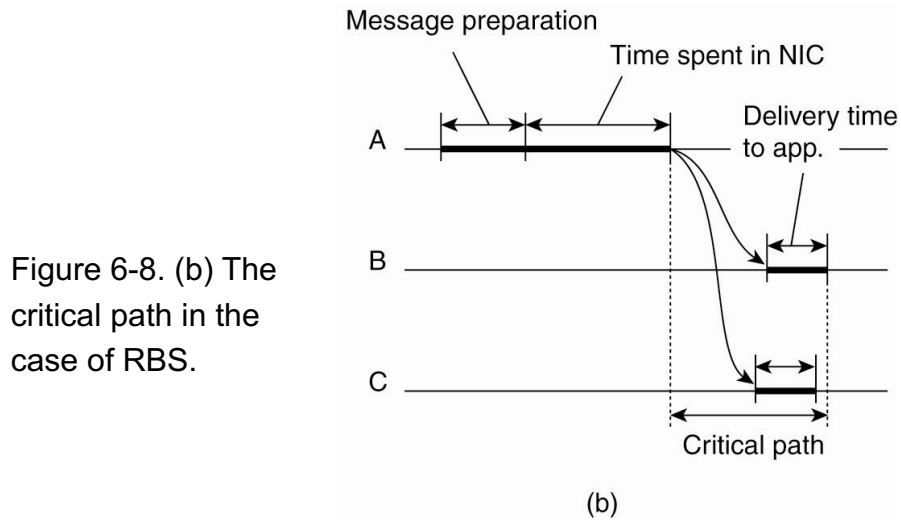
Figure 6-8. (a) The usual critical path in determining network delays.



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

12

Clock Synchronization in Wireless Networks (2)



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

13

Lamport's Logical Clocks (1)

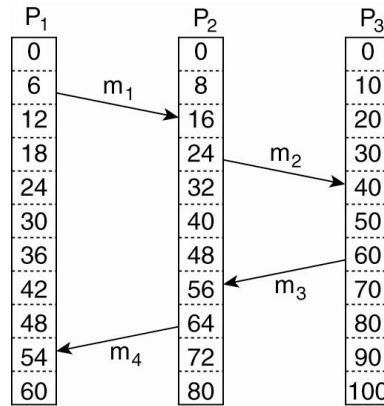
The "happens-before" relation \rightarrow can be observed directly in two situations:

- If a and b are events in the same process, and a occurs before b , then $a \rightarrow b$ is true.
- If a is the event of a message being sent by one process, and b is the event of the message being received by another process, then $a \rightarrow b$

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

14

Lamport's Logical Clocks (2)



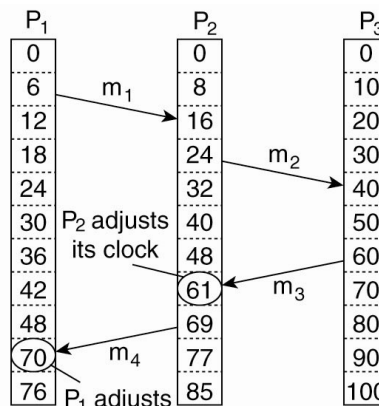
(a)

Figure 6-9. (a) Three processes, each with its own clock. The clocks run at different rates.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

15

Lamport's Logical Clocks (3)



(b)

Figure 6-9. (b) Lamport's algorithm corrects the clocks.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

16

Lamport's Logical Clocks (4)

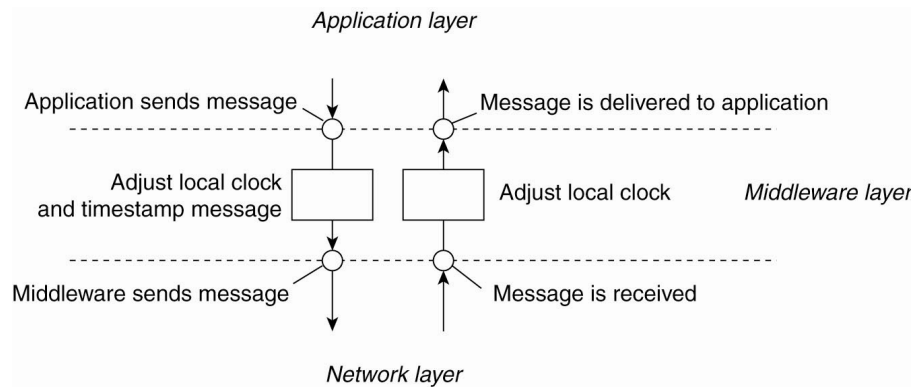


Figure 6-10. The positioning of Lamport's logical clocks in distributed systems.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

17

Lamport's Logical Clocks (5)

Updating counter C_i for process P_i

1. Before executing an event P_i executes $C_i \leftarrow C_i + 1$.
2. When process P_i sends a message m to P_j , it sets m 's timestamp $ts(m)$ equal to C_i after having executed the previous step.
3. Upon the receipt of a message m , process P_j adjusts its own local counter as $C_j \leftarrow \max\{C_j, ts(m)\}$, after which it then executes the first step and delivers the message to the application.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

18

Example: Totally Ordered Multicasting

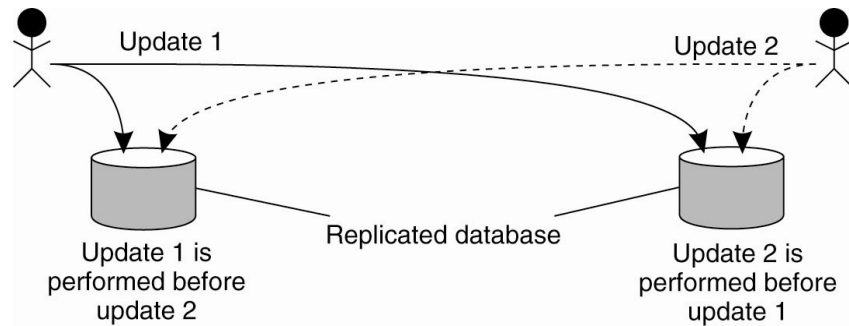


Figure 6-11. Updating a replicated database and leaving it in an inconsistent state.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

19

Totally Ordered Multicast

1. On receiving an update from client, broadcast to others (including yourself)
2. On receiving or processing an update:
 1. Add it to your local queue, if received update
 2. Broadcast an ack to every replica (including yourself) only from head of queue
3. On receiving an acknowledgement:
 1. Mark corresponding update ack'ed in your queue
4. Remove a process updates everyone has ack'ed from head of queue

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

20

Vector Clocks (1)

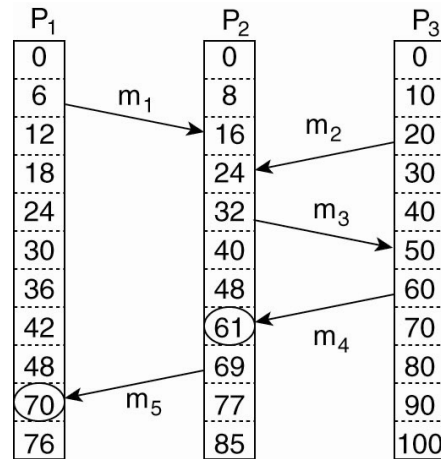


Figure 6-12. Concurrent message transmission using logical clocks.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

21

Vector Clocks (2)

Vector clocks are constructed by letting each process P_i maintain a vector VC_i with the following two properties:

1. $VC_i[i]$ is the number of events that have occurred so far at P_i . In other words, $VC_i[i]$ is the local logical clock at process P_i .
2. If $VC_i[j] = k$ then P_i knows that k events have occurred at P_j . It is thus P_i 's knowledge of the local time at P_j .

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

22

Vector Clocks (3)

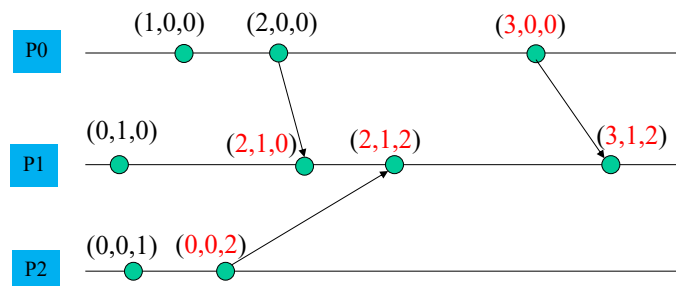
Steps carried out to accomplish property 2 of previous slide:

1. Before executing an event P_i executes $VC_i[i] \leftarrow VC_i[i] + 1$.
2. When process P_i sends a message m to P_j , it sets m 's (vector) timestamp $ts(m)$ equal to VC_i after having executed the previous step.
3. Upon the receipt of a message m , process P_j adjusts its own vector by setting $VC_j[k] \leftarrow \max\{VC_j[k], ts(m)[k]\}$ for each k , after which it executes the first step and delivers the message to the application.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

23

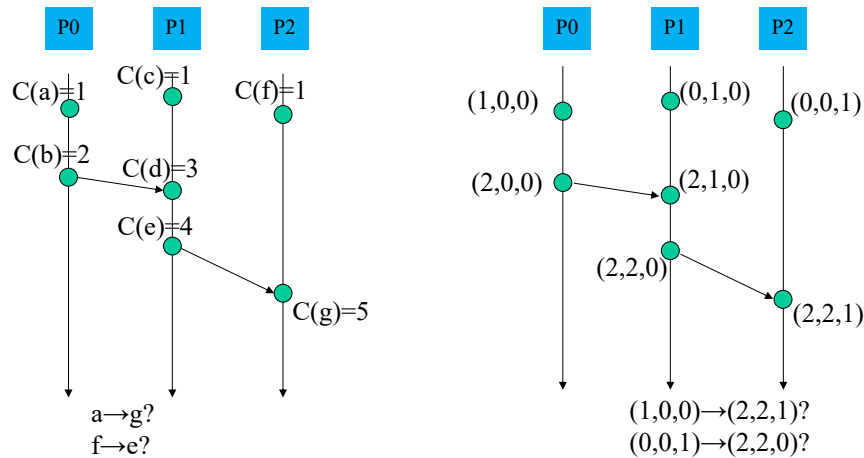
Vector Clock Example



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

24

Lamport Clk vs. Vector Clk



Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

25

Enforcing Causal Communication

- Ensure that a message is delivered only in order
- Suppose clocks are only adjusted when sending and receiving messages
 - P_j receives a message m from P_i with timestamp $ts(m)$. The message delivery conditions are:
 1. $ts(m)[i] = VC_j[i] + 1$
 2. $ts(m)[k] \leq VC_j[k]$ for all $k \neq i$
 - Otherwise, the message is not delivered

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

26

Enforcing Causal Communication

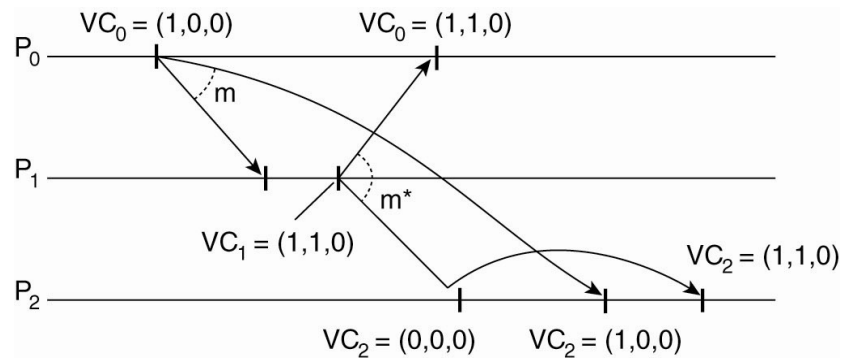


Figure 6-13. Enforcing causal communication.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

27

Mutual Exclusion A Centralized Algorithm (1)

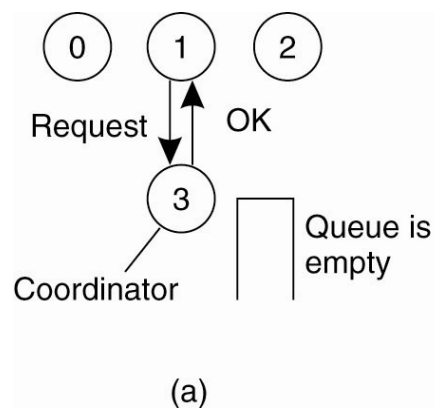


Figure 6-14. (a) Process 1 asks the coordinator for permission to access a shared resource. Permission is granted.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

28

Mutual Exclusion A Centralized Algorithm (2)

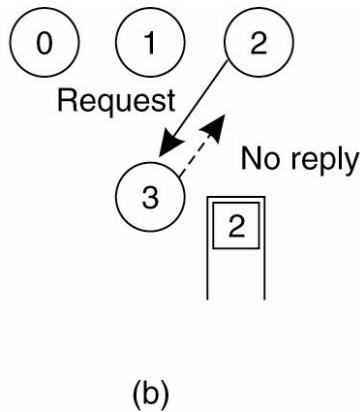


Figure 6-14. (b) Process 2 then asks permission to access the same resource. The coordinator does not reply.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

29

Mutual Exclusion A Centralized Algorithm (3)

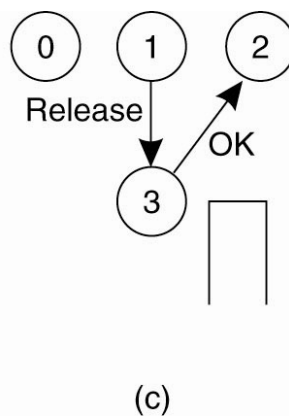


Figure 6-14. (c) When process 1 releases the resource, it tells the coordinator, which then replies to 2.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

30

A Distributed Algorithm (1)

Three different cases:

1. If the receiver is not accessing the resource and does not want to access it, it sends back an OK message to the sender.
2. If the receiver already has access to the resource, it simply does not reply. Instead, it queues the request.
3. If the receiver wants to access the resource as well but has not yet done so, it compares the timestamp of the incoming message with the one contained in the message that it has sent everyone. The lowest one wins.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

31

A Distributed Algorithm (2)

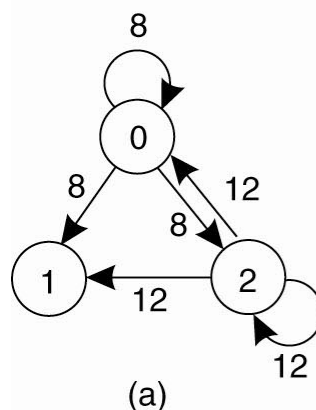


Figure 6-15. (a) Two processes want to access a shared resource at the same moment.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

32

A Distributed Algorithm (3)

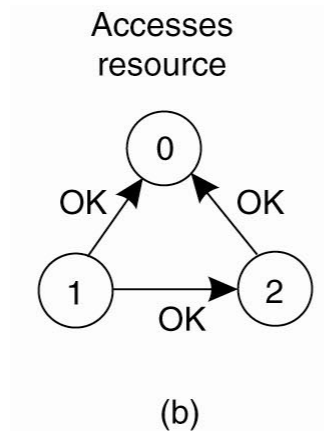


Figure 6-15. (b) Process 0 has the lowest timestamp, so it wins.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

33

A Distributed Algorithm (4)

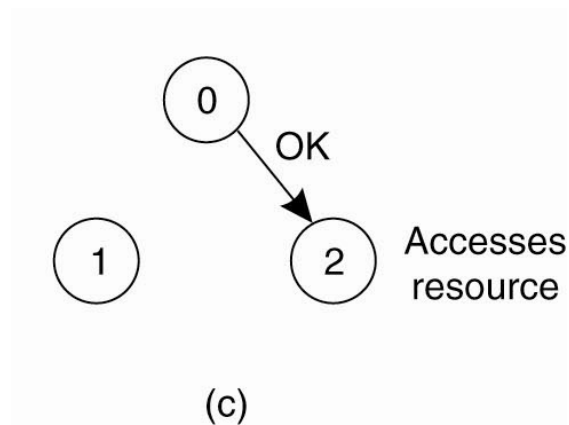


Figure 6-15. (c) When process 0 is done, it sends an OK also, so 2 can now go ahead.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

34

A Token Ring Algorithm

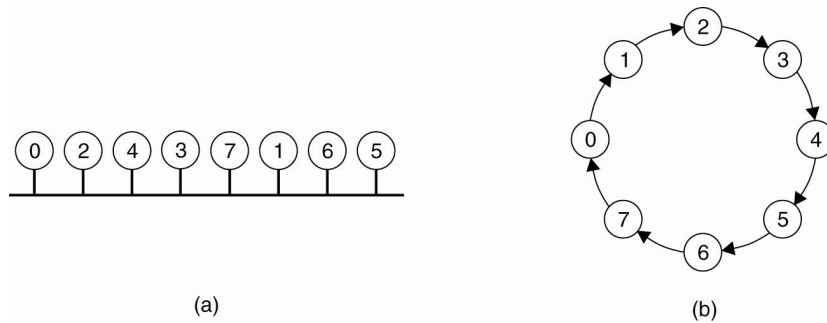


Figure 6-16. (a) An unordered group of processes on a network.
(b) A logical ring constructed in software.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

35

A Comparison of the Four Algorithms

Algorithm	Messages per entry/exit	Delay before entry (in message times)	Problems
Centralized	3	2	Coordinator crash
Decentralized	$3mk$, $k = 1, 2, \dots$	$2m$	Starvation, low efficiency
Distributed	$2(n - 1)$	$2(n - 1)$	Crash of any process
Token ring	1 to ∞	0 to $n - 1$	Lost token, process crash

Figure 6-17. A comparison of three mutual exclusion algorithms.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

36

Election Algorithms

The Bully Algorithm

1. P sends an *ELECTION* message to all processes with higher numbers.
2. If no one responds, P wins the election and becomes coordinator.
3. If one of the higher-ups answers, it takes over. P 's job is done.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

37

The Bully Algorithm (1)

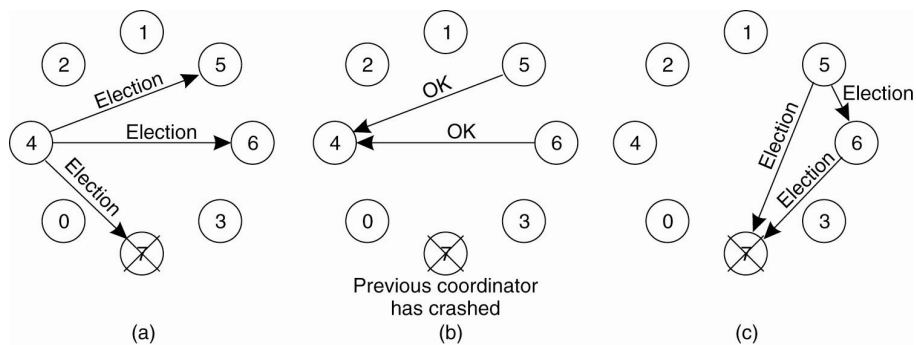


Figure 6-20. The bully election algorithm. (a) Process 4 holds an election. (b) Processes 5 and 6 respond, telling 4 to stop. (c) Now 5 and 6 each hold an election.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

38

The Bully Algorithm (2)

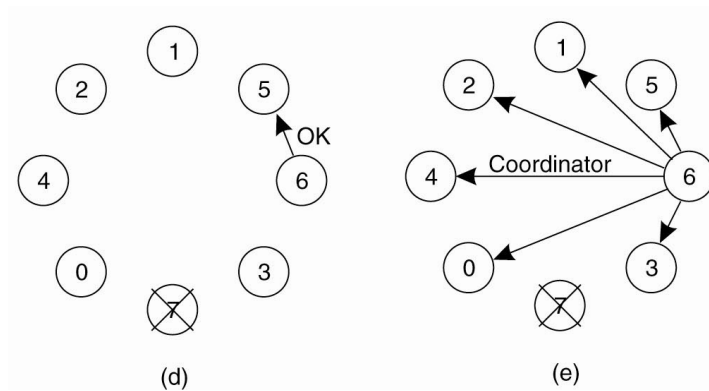


Figure 6-20. The bully election algorithm. (d) Process 6 tells 5 to stop. (e) Process 6 wins and tells everyone.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

39

A Ring Algorithm

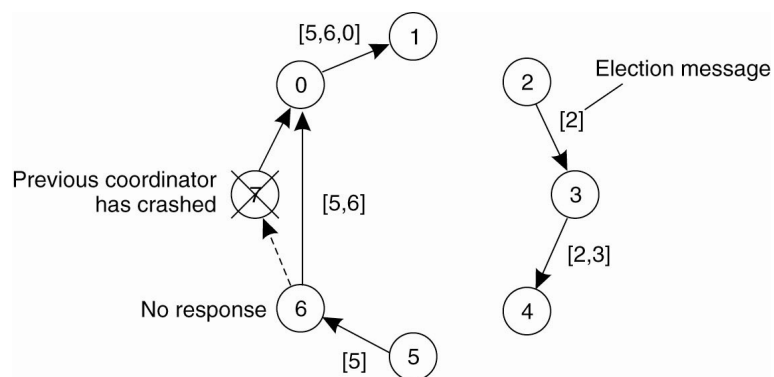


Figure 6-21. Election algorithm using a ring.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

40

Elections in Wireless Environments (1)

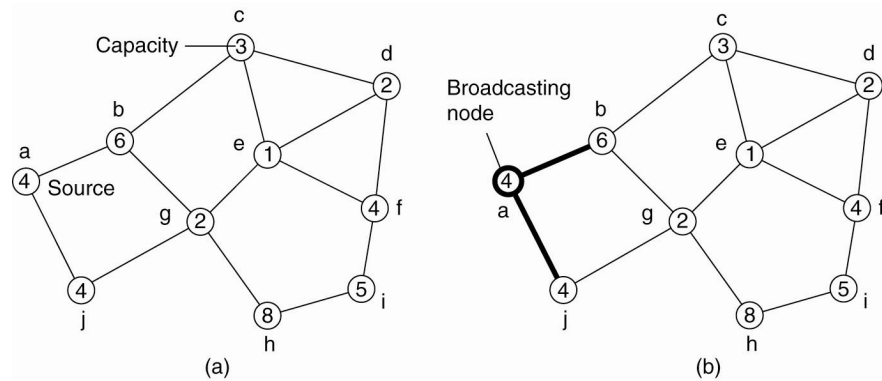


Figure 6-22. Election algorithm in a wireless network, with node a as the source. (a) Initial network. (b)–(e) The build-tree phase

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

41

Elections in Wireless Environments (2)

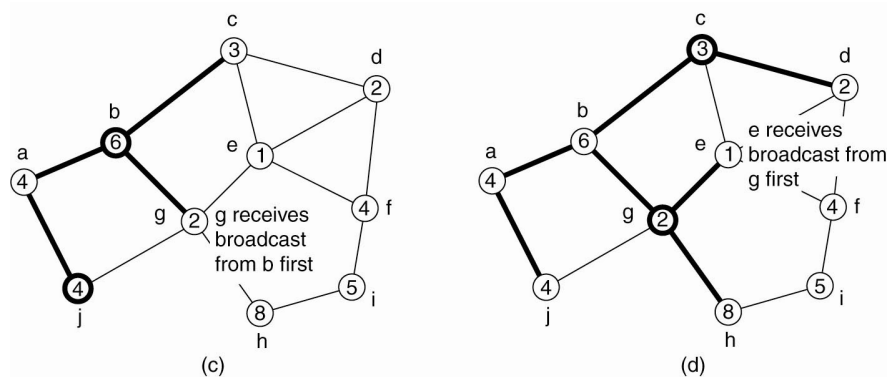


Figure 6-22. Election algorithm in a wireless network, with node a as the source. (a) Initial network. (b)–(e) The build-tree phase

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

42

Elections in Wireless Environments (3)

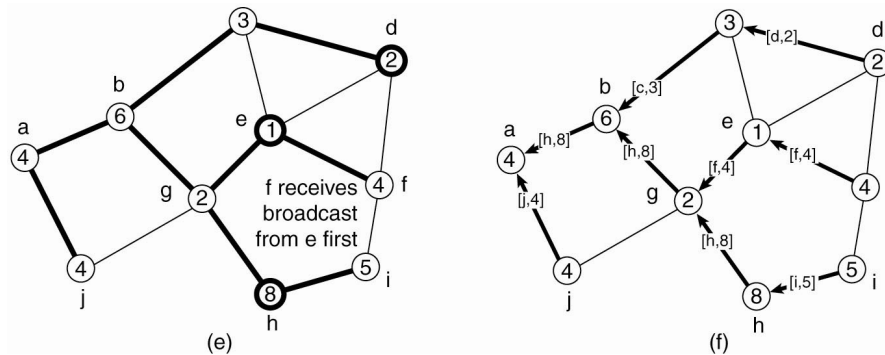


Figure 6-22. (e) The build-tree phase.
(f) Reporting of best node to source.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

43

Elections in Large-Scale Systems (1)

Requirements for superpeer selection:

1. Normal nodes should have low-latency access to superpeers.
2. Superpeers should be evenly distributed across the overlay network.
3. There should be a predefined portion of superpeers relative to the total number of nodes in the overlay network.
4. Each superpeer should not need to serve more than a fixed number of normal nodes.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5

44

Elections in Large-Scale Systems (2)

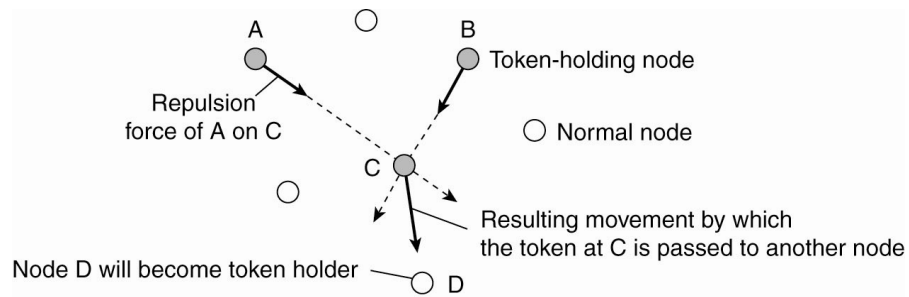


Figure 6-23. Moving tokens in a two-dimensional space using repulsion forces.

Tanenbaum & Van Steen, Distributed Systems: Principles and Paradigms, 2e, (c) 2007 Prentice-Hall, Inc. All rights reserved. 0-13-239227-5