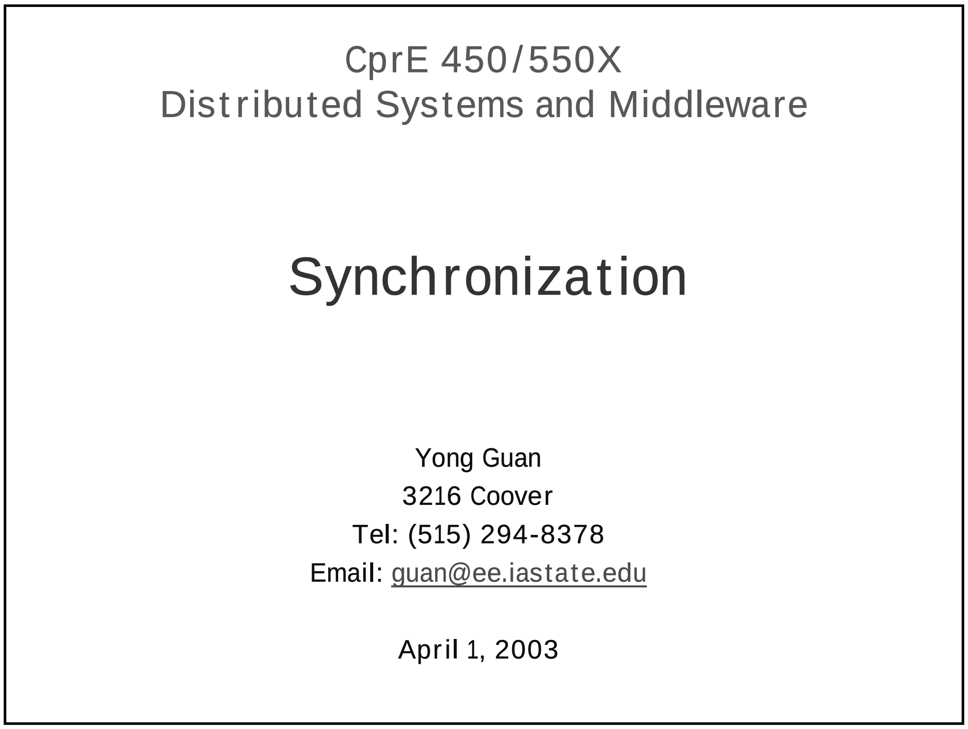
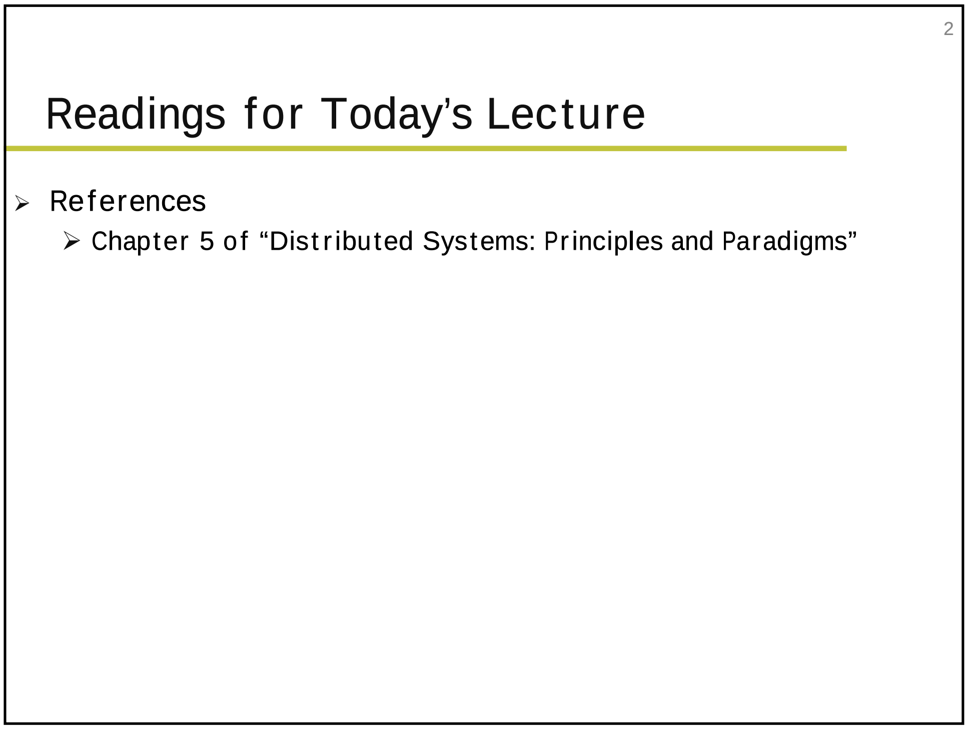


CprE 450/550X
Distributed Systems and Middleware

Synchronization

Yong Guan
3216 Coover
Tel: (515) 294-8378
Email: guan@ee.iastate.edu

April 1, 2003



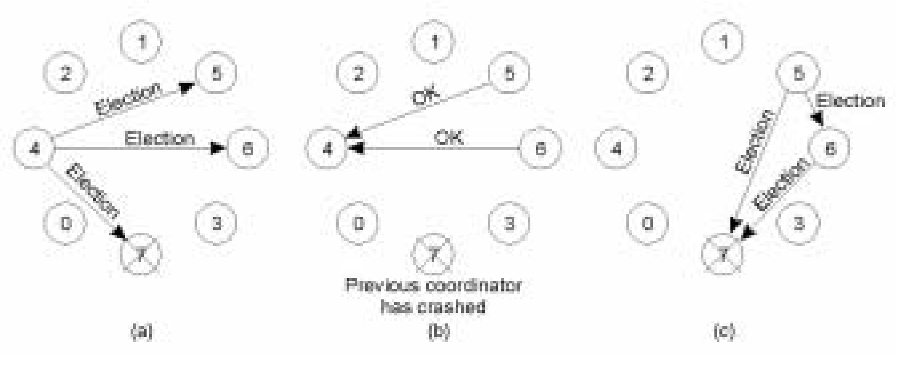
Readings for Today's Lecture

- References
  - Chapter 5 of "Distributed Systems: Principles and Paradigms"

Election Algorithms

- Many distributed algorithms requires one process in the system acts as a leader (coordinator, initiator).
- It does not matter which process it is, but one of them has to do it.
- The goal of election algorithm is to ensure that when an election starts, it concludes with all processes agreeing on who the new coordinator is to be.

The Bully Algorithm (1)



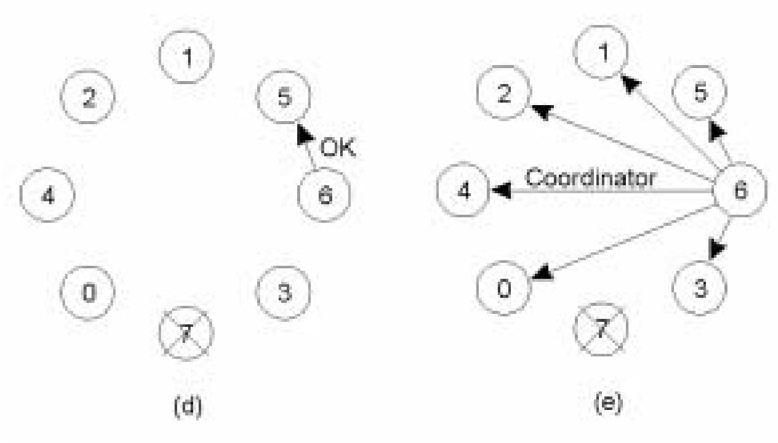
The bully election algorithm

a) Process 4 holds an election

b) Process 5 and 6 respond, telling 4 to stop

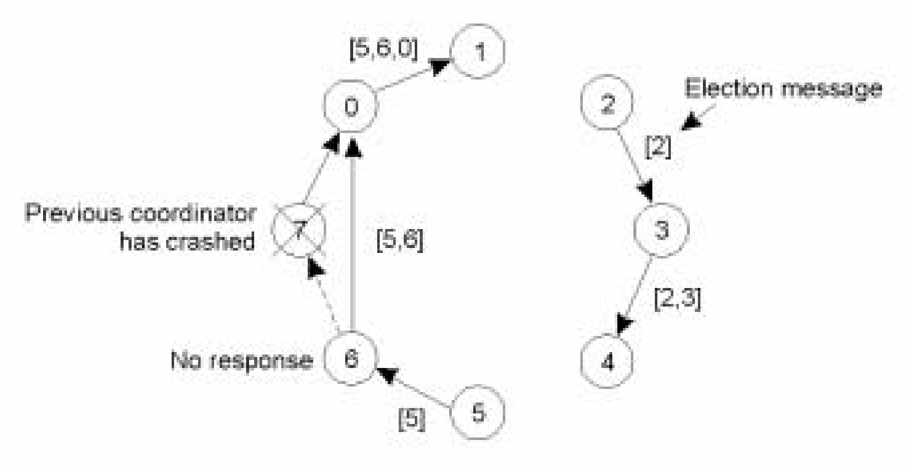
c) Now 5 and 6 each hold an election

The Bully Algorithm (2)



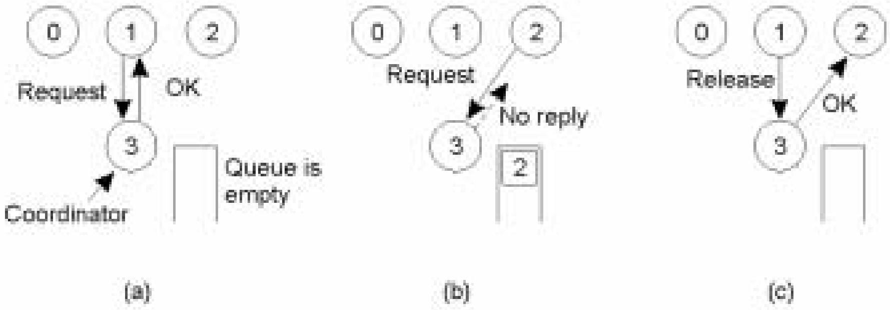
d) Process 6 tells 5 to stop
e) Process 6 wins and tells everyone

A Ring Algorithm



Election algorithm using a ring.

Mutual Exclusion: A Centralized Algorithm

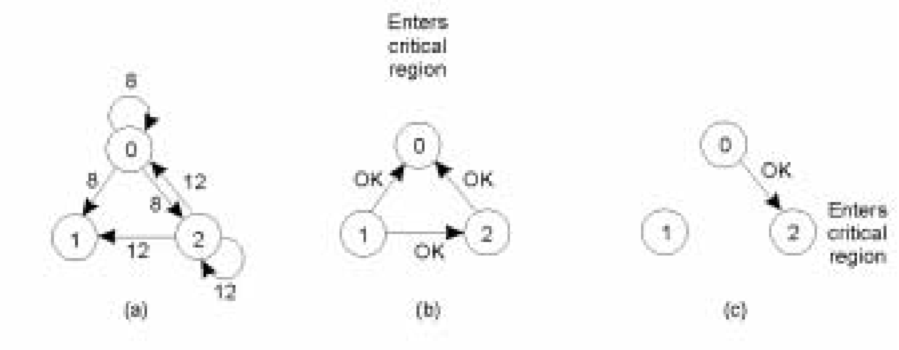


a) Process 1 asks the coordinator for permission to enter a critical region. Permission is granted

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

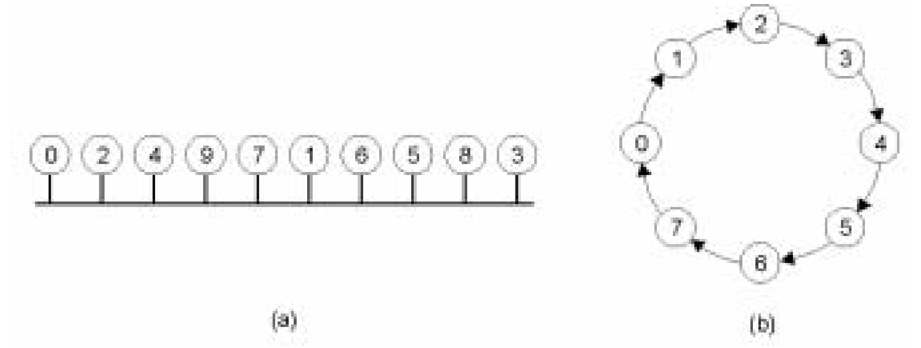


a) Two processes want to enter the same critical region at the same moment.

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

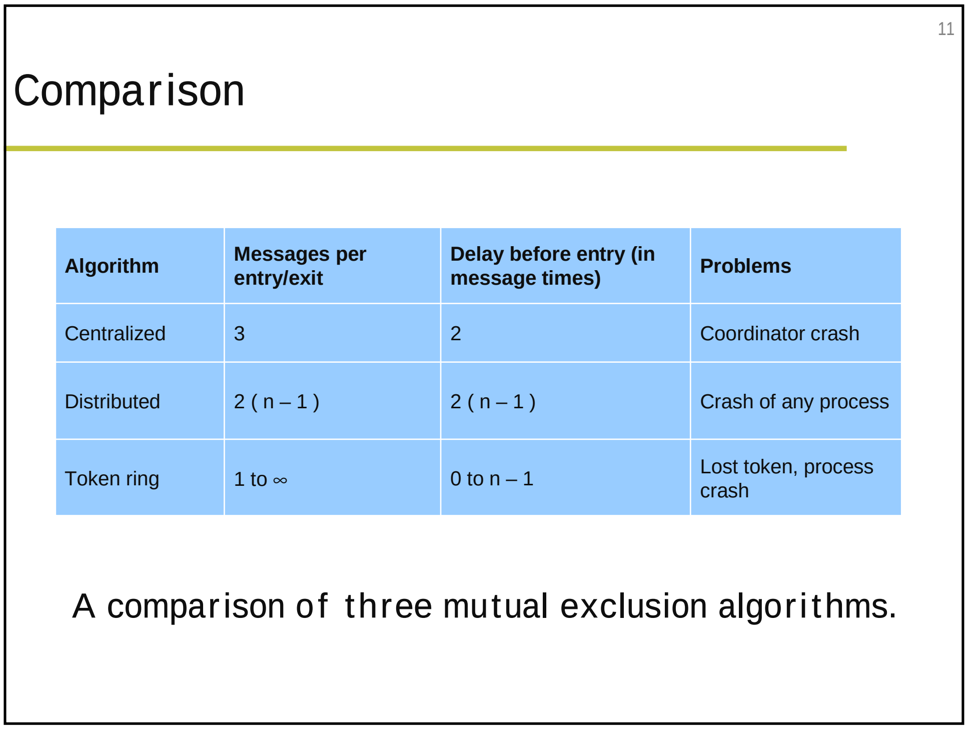
c) When process 0 is done, it sends an OK also, so 2 can now enter the critical region.

A Toke Ring Algorithm



a) An unordered group of processes on a network.

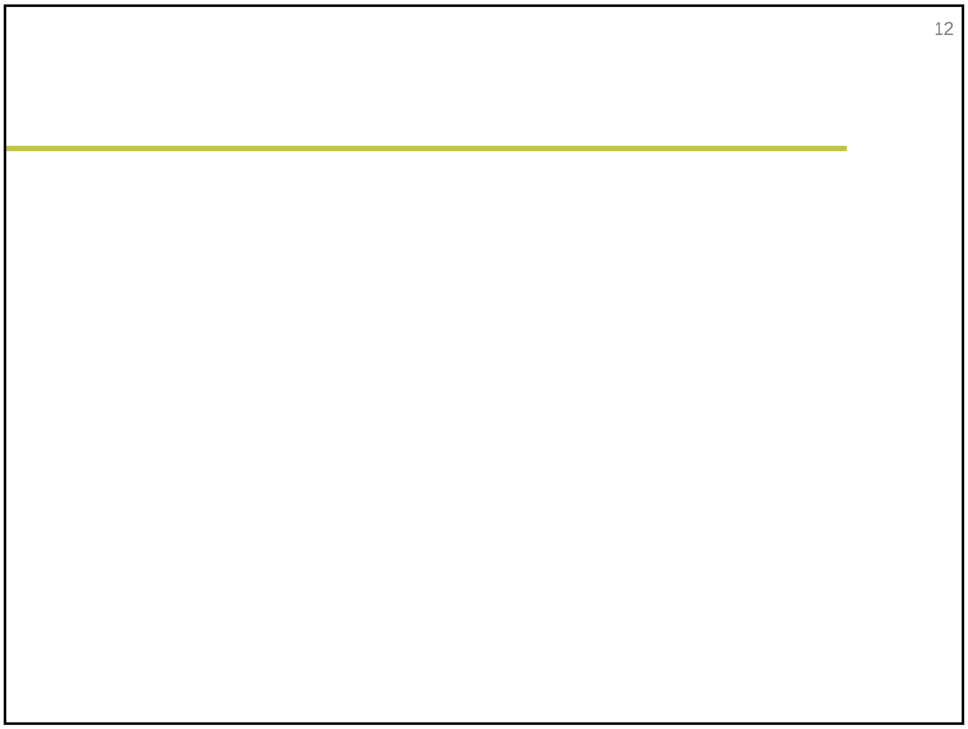
b) A logical ring constructed in software.



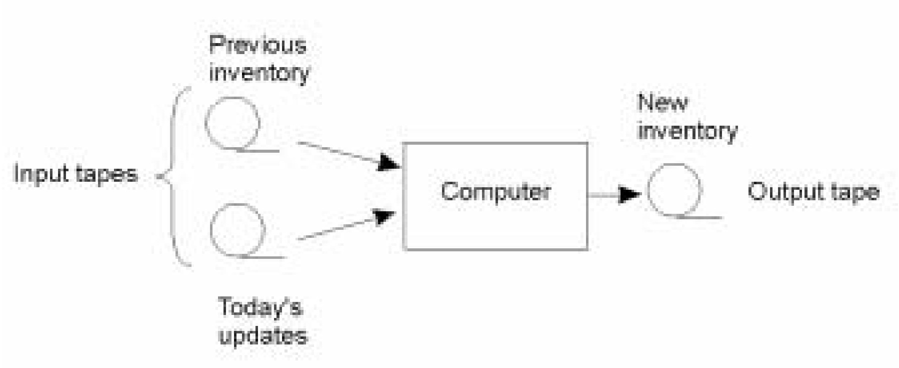
Comparison

Algorithm	Messages per entry/exit	Delay before entry (in message times)	Problems
Centralized	3	2	Coordinator crash
Distributed	2 (n – 1)	2 (n – 1)	Crash of any process
Token ring	1 to ∞	0 to n – 1	Lost token, process crash

A comparison of three mutual exclusion algorithms.



The Transaction Model (1)

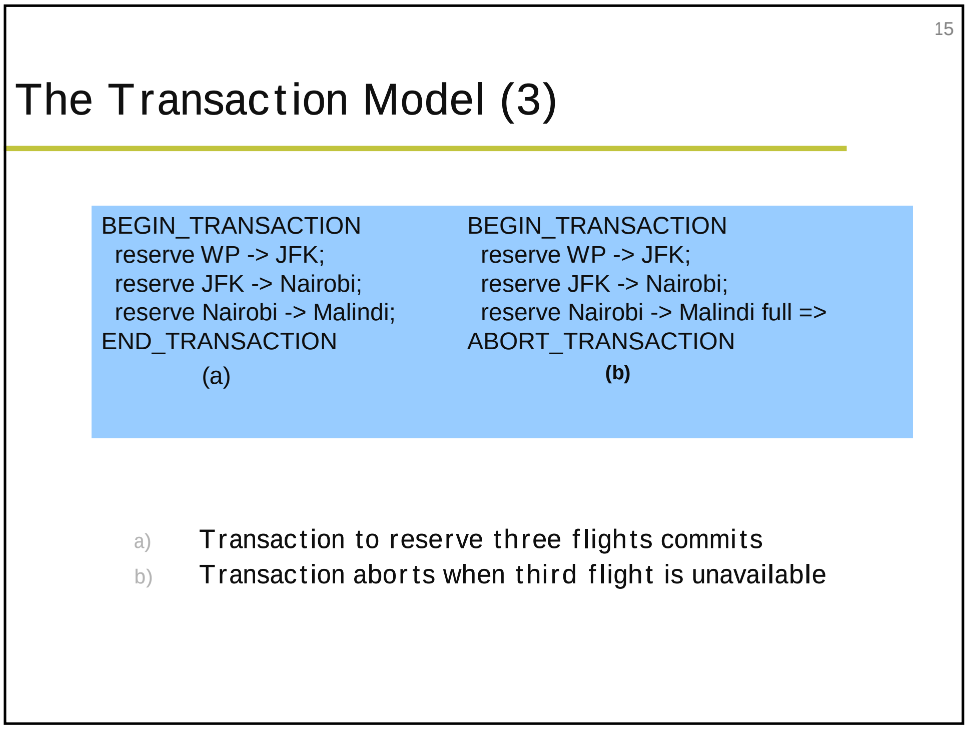


Updating a master tape is fault tolerant.

The Transaction Model (2)

Primitive	**Description**
BEGIN_TRANSACTION	Make the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

Examples of primitives for transactions.

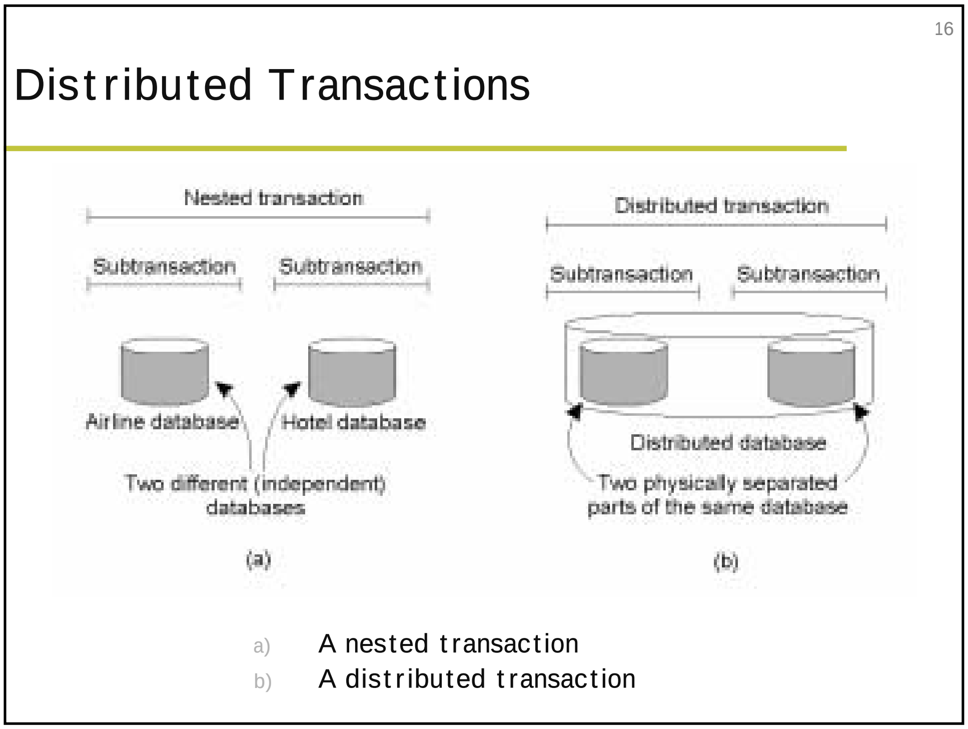


15

The Transaction Model (3)

```
BEGIN_TRANSACTION                    BEGIN_TRANSACTION
  reserve WP -> JFK;                   reserve WP -> JFK;
  reserve JFK -> Nairobi;              reserve JFK -> Nairobi;
  reserve Nairobi -> Malindi;          reserve Nairobi -> Malindi full =>
END_TRANSACTION                      ABORT_TRANSACTION
            (a)                                  (b)
```

a) Transaction to reserve three flights commits
b) Transaction aborts when third flight is unavailable

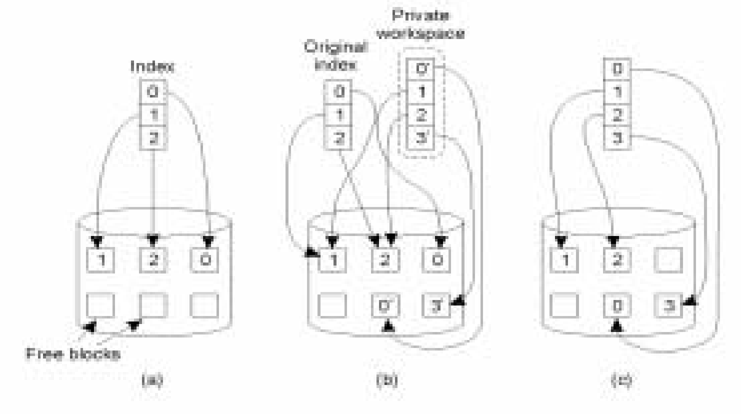


16

Distributed Transactions

a) A nested transaction
b) A distributed transaction

Private Workspace



a) The file index and disk blocks for a three-block file

b) The situation after a transaction has modified block 0 and appended block 3

c) After committing

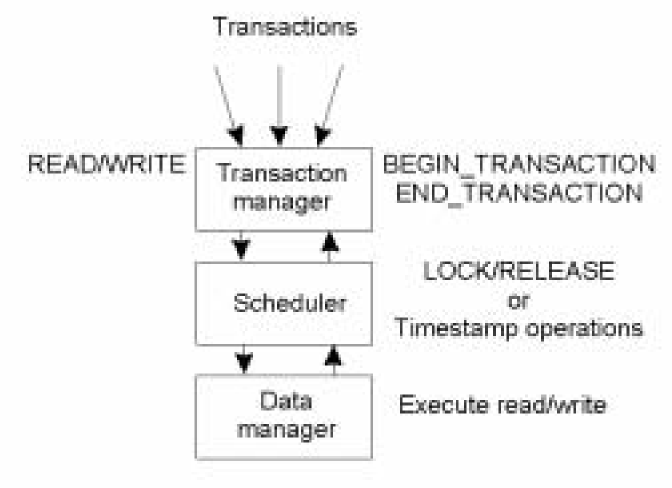
Writeahead Log

x = 0;	Log	Log	Log
y = 0;			
BEGIN_TRANSACTION;			
x = x + 1;	[x = 0 / 1]	[x = 0 / 1]	[x = 0 / 1]
y = y + 2		[y = 0/2]	[y = 0/2]
x = y * y;			[x = 1/4]
END_TRANSACTION;			
(a)	(b)	(c)	(d)

a) A transaction

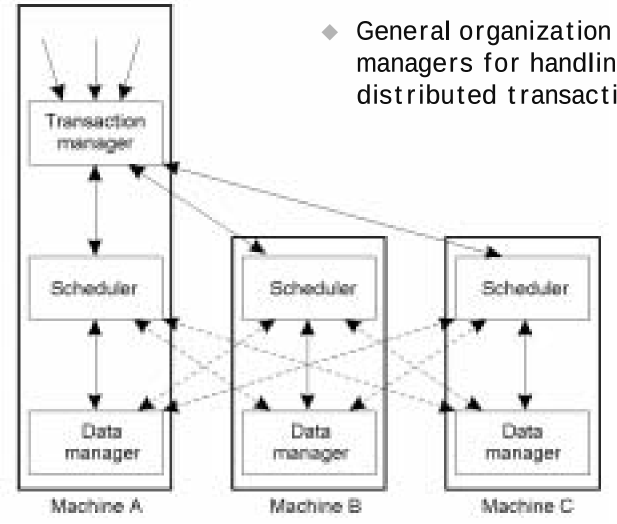
b) – d) The log before each statement is executed

Concurrency Control (1)



- General organization of managers for handling transactions.

Concurrency Control (2)



- General organization of managers for handling distributed transactions.

Serializability

```
BEGIN_TRANSACTION
  x = 0;
  x = x + 1;
END_TRANSACTION
```

(a)

```
BEGIN_TRANSACTION
  x = 0;
  x = x + 2;
END_TRANSACTION
```

(b)

```
BEGIN_TRANSACTION
  x = 0;
  x = x + 3;
END_TRANSACTION
```

(c)

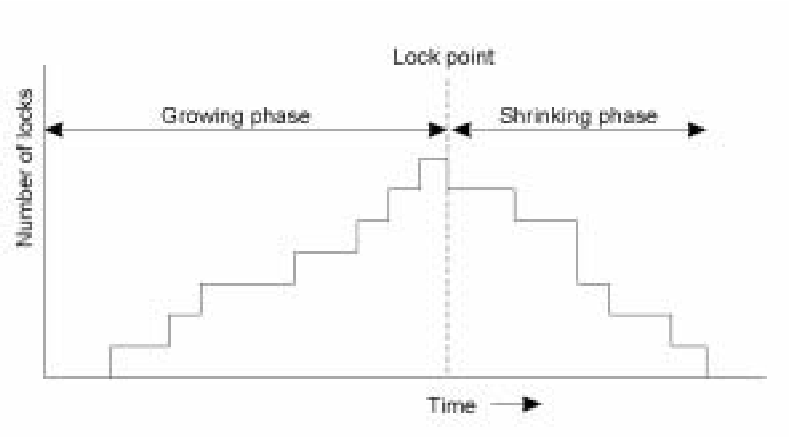
Schedule 1	x = 0; x = x + 1; x = 0; x = x + 2; x = 0; x = x + 3	Legal
Schedule 2	x = 0; x = 0; x = x + 1; x = x + 2; x = 0; x = x + 3;	Legal
Schedule 3	x = 0; x = 0; x = x + 1; x = 0; x = x + 2; x = x + 3;	Illegal

(d)

a) – c) Three transactions T_1, T_2, and T_3

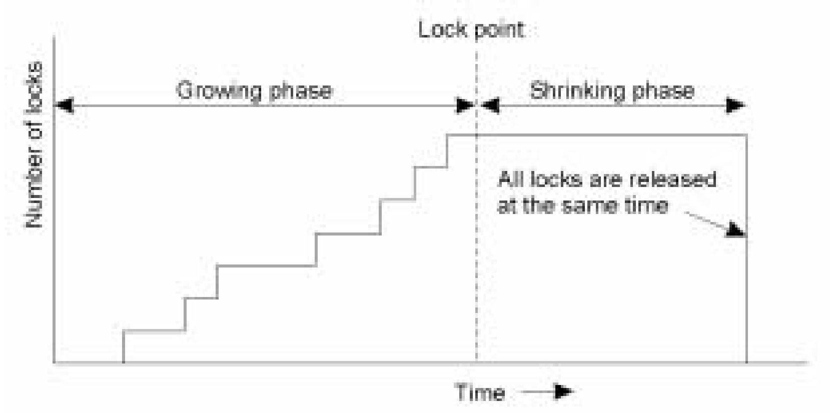
d) Possible schedules

Two-Phase Locking (1)



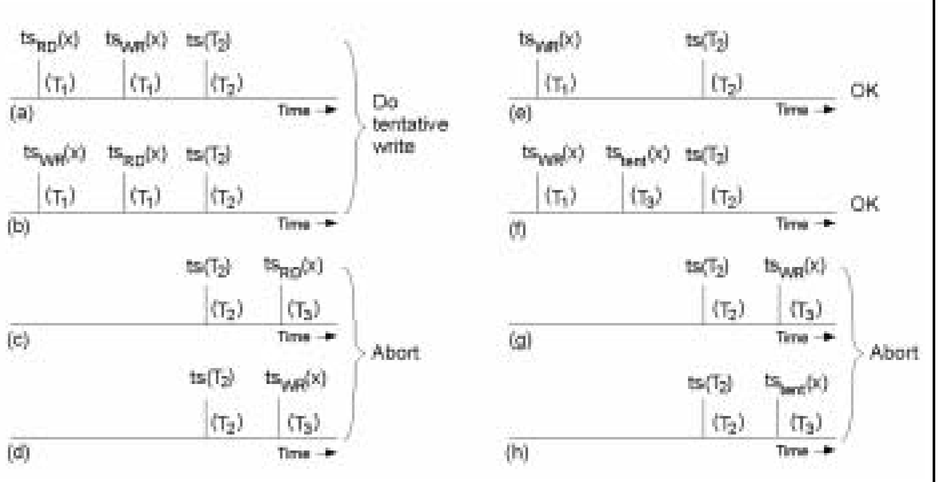
Two-phase locking.

Two-Phase Locking (2)



Strict two-phase locking.

Pessimistic Timestamp Ordering



Concurrency control using timestamps.