Synchronization

Chapter 5
When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time.
Physical Clocks (1)

Computation of the mean solar day.

A transit of the sun occurs when the sun reaches the highest point of the day.

At the transit of the sun n days later, the earth has rotated fewer than 360°.

Earth on day 0 at the transit of the sun.

Earth on day n at the transit of the sun.

To distant galaxy.
Physical Clocks (2)

TAI seconds are of constant length, unlike solar seconds. Leap seconds are introduced when necessary to keep in phase with the sun.
Clock Synchronization Algorithms

The relation between clock time and UTC when clocks tick at different rates.

The diagram illustrates the following:
- **Perfect clock**: $\frac{dC}{dt} = 1$
- **Fast clock**: $\frac{dC}{dt} > 1$
- **Slow clock**: $\frac{dC}{dt} < 1$
Cristian's Algorithm

Both $T_0$ and $T_1$ are measured with the same clock

Getting the current time from a time server.
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
a) Three processes, each with its own clock. The clocks run at different rates.
b) Lamport's algorithm corrects the clocks.
Example: Totally-Ordered Multicasting

Updating a replicated database and leaving it in an inconsistent state.
Global State (1)

(a) A consistent cut

(b) An inconsistent cut

Sender of m2 cannot be identified with this cut
a) Organization of a process and channels for a distributed snapshot
b) Process Q receives a marker for the first time and records its local state

c) Q records all incoming message

d) Q receives a marker for its incoming channel and finishes recording the state of the incoming channel
The Bully Algorithm (1)

The bully election algorithm

- Process 4 holds an election
- Process 5 and 6 respond, telling 4 to stop
- Now 5 and 6 each hold an election
Global State (3)

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Messages per entry/exit</th>
<th>Delay before entry (in message times)</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>3</td>
<td>2</td>
<td>Coordinator crash</td>
</tr>
<tr>
<td>Distributed</td>
<td>$2(n - 1)$</td>
<td>$2(n - 1)$</td>
<td>Crash of any process</td>
</tr>
<tr>
<td>Token ring</td>
<td>1 to $\infty$</td>
<td>0 to $n - 1$</td>
<td>Lost token, process crash</td>
</tr>
</tbody>
</table>

A comparison of three mutual exclusion algorithms.
The Transaction Model (1)

Updating a master tape is fault tolerant.

Updating a master tape is fault tolerant.
The Transaction Model (2)

Examples of primitives for transactions.

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN_TRANSACTION</td>
<td>Make the start of a transaction</td>
</tr>
<tr>
<td>END_TRANSACTION</td>
<td>Terminate the transaction and try to commit</td>
</tr>
<tr>
<td>ABORT_TRANSACTION</td>
<td>Kill the transaction and restore the old values</td>
</tr>
<tr>
<td>READ</td>
<td>Read data from a file, a table, or otherwise</td>
</tr>
<tr>
<td>WRITE</td>
<td>Write data to a file, a table, or otherwise</td>
</tr>
</tbody>
</table>
The Transaction Model (3)

(a) Transaction to reserve three flights commits

```
BEGIN_TRANSACTION
  reserve WP -> JFK;
  reserve JFK -> Nairobi;
  reserve Nairobi -> Malindi;
END_TRANSACTION
```

(b) Transaction aborts when third flight is unavailable

```
BEGIN_TRANSACTION
  reserve WP -> JFK;
  reserve JFK -> Nairobi;
  reserve Nairobi -> Malindi full =>
ABORT_TRANSACTION
```
Distributed Transactions

(a) A nested transaction

Subtransaction

Airline database

Hotel database

Two different (independent) databases

(b) A distributed transaction

Subtransaction

Distributed database

Two physically separated parts of the same database
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;
y = 0;
BEGIN_TRANSACTION;
x = x + 1;
y = y + 2
x = y * y;
END_TRANSACTION;

(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

(a)  
BEGIN_TRANSACTION  
x = 0; 
x = x + 1;  
END_TRANSACTION  

(b)  
BEGIN_TRANSACTION  
x = 0;  
x = x + 2;  
END_TRANSACTION  

(c)  
BEGIN_TRANSACTION  
x = 0;   
x = x + 2;  
END_TRANSACTION  

(d)  
<table>
<thead>
<tr>
<th>Schedule 1</th>
<th>Schedule 2</th>
<th>Schedule 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 0; x = x + 1; x = 0; x = x + 2; x = 0; x = x + 3</td>
<td>x = 0; x = 0; x = x + 1; x = x + 2; x = 0; x = x + 3</td>
<td>x = 0; x = 0; x = x + 1; x = 0; x = x + 2; x = x + 3</td>
</tr>
</tbody>
</table>

| Legal | Legal | Illegal |

(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.

\[
\begin{align*}
\text{(a)} & \quad \begin{array}{ccc}
\text{ts}_{RD}(x) & \text{ts}_{WR}(x) & \text{ts}(T_2) \\
(T_1) & (T_1) & (T_2)
\end{array} & \text{Do tentative write} \\
\text{(b)} & \quad \begin{array}{ccc}
\text{ts}_{WR}(x) & \text{ts}_{RD}(x) & \text{ts}(T_2) \\
(T_1) & (T_1) & (T_2)
\end{array} \\
\text{(c)} & \quad \begin{array}{cc}
\text{ts}(T_2) & \text{ts}_{RD}(x) \\
(T_2) & (T_3)
\end{array} & \text{Abort} \\
\text{(d)} & \quad \begin{array}{cc}
\text{ts}(T_2) & \text{ts}_{WR}(x) \\
(T_2) & (T_3)
\end{array}
\end{align*}
\]

\[
\begin{align*}
\text{(e)} & \quad \begin{array}{cc}
\text{ts}_{WR}(x) & \text{ts}(T_2) \\
(T_1) & (T_2)
\end{array} & \text{OK} \\
\text{(f)} & \quad \begin{array}{ccc}
\text{ts}_{WR}(x) & \text{ts}_{tent}(x) & \text{ts}(T_2) \\
(T_1) & (T_3) & (T_2)
\end{array} & \text{OK} \\
\text{(g)} & \quad \begin{array}{ccc}
\text{ts}(T_2) & \text{ts}_{WR}(x) \\
(T_2) & (T_3)
\end{array} & \text{Abort} \\
\text{(h)} & \quad \begin{array}{ccc}
\text{ts}(T_2) & \text{ts}_{tent}(x) \\
(T_2) & (T_3)
\end{array}
\end{align*}
\]