Announcements

- Project 1 update – Thursday
  - Due 2/26
- HW 1 – due Thursday

Last Lecture – Clock Sync
Important Lessons

- Clocks on different systems will always behave differently
  - Skew and drift between clocks
- Time disagreement between machines can result in undesirable behavior
- Two paths to solution: synchronize clocks or ensure consistent clocks
- Clock synchronization
  - Rely on a time-stamped network messages
  - Estimate delay for message transmission
  - Can synchronize to UTC or to local source

Today's Lecture

- Lamport Clocks
- Vector Clocks
- Mutual Exclusion
- Election

15-446 Distributed Systems
Spring 2009

L-9 Logical Time
Example: Totally Ordered Multicasting

Updating a replicated database and leaving it in an inconsistent state.

Logical time and logical clocks (Lamport 1978)

1. Events at three processes
2. If two events occurred at the same process $p_i$ ($i = 1, 2, \ldots, N$) then they occurred in the order observed by $p_i$, that is $a \rightarrow b$.
3. When a message $m$ is sent between two processes, $\text{send}(m)$ happened before $\text{receive}(m)$.
4. The happened before relation is transitive.
5. The happened before relation is the relation of causal ordering.
6. $a \rightarrow b$ (at $p_1$) $c \rightarrow d$ (at $p_2$)
7. $b \rightarrow c$ because of $m_1$
8. also $d \rightarrow f$ because of $m_2$
Not all events are related by an arrow.
Consider a and e (different processes and no chain of messages to relate them).
They are not related by an arrow; they are said to be concurrent.
Written as a || e.

A logical clock is a monotonically increasing software counter.
It need not relate to a physical clock.
Each process $p_i$ has a logical clock, $L_i$, which can be used to apply logical timestamps to events.

- Rule 1: $L_i$ is incremented by 1 before each event at process $p_i$.
- Rule 2: (a) when process $p_i$ sends message $m$, it piggybacks $t = L_i$.
(b) when $p_j$ receives $(m, t)$ it sets $L_j := \max(t, L_j)$ and applies LC1 before timestamping the event receive $(m)$.

e $\rightarrow$ e implies $L(e) < L(e')$
The converse is not true, that is $L(e) < L(e')$ does not imply $e \rightarrow e'$.
e.g. $L(b) > L(e)$ but $b \parallel e$. 
Today's Lecture

- Lamport Clocks
- Vector Clocks
- Mutual Exclusion
- Election

Vector Clocks

- Vector clocks overcome the shortcoming of Lamport logical clocks
  - \( L(e) < L(e') \) does not imply \( e \) happened before \( e' \)
- Vector timestamps are used to timestamp local events
- They are applied in schemes for replication of data

Vector Clocks

- \( V_i[j] \) is the number of events that \( p_i \) has timestamped
- \( V_i[j] (i \neq j) \) is the number of events at \( p_j \) that \( p_i \) has been affected by

Vector clock \( V_i \) at process \( p_i \) is an array of \( N \) integers
1. initially \( V_i[j] = 0 \) for \( i, j = 1, 2, \ldots N \)
2. before \( p_i \) timestamps an event it sets \( V_i[j] := V_i[j] + 1 \)
3. \( p_i \) piggybacks \( t = V_i \) on every message it sends
4. when \( p_i \) receives \((m,t)\) it sets \( V_i[j] := \max(V_i[j], t[j]) \) \( j = 1, 2, \ldots N \) (then before next event adds 1 to own element using rule 2)
Vector Clocks

- Note that \( e \rightarrow e' \) implies \( L(e) < L(e') \). The converse is also true.
- Can you see a pair of parallel events?
  - \( c \parallel e' \) parallel because neither \( \forall(c) \leq \forall(e) \) nor \( \forall(e) \leq \forall(c) \)

Today's Lecture

- Lamport Clocks
- Vector Clocks
- Mutual Exclusion
- Election

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.

Mutual Exclusion

A Centralized Algorithm

1. Process 1 asks the coordinator for permission to access a shared resource → Permission is granted
2. Process 2 then asks permission to access the same resource → The coordinator does not reply
3. When process 1 releases the resource, it tells the coordinator, which then replies to 2
A Distributed Algorithm (2)

- Two processes want to access a shared resource at the same moment.
- Process 0 has the lowest timestamp, so it wins.
- When process 0 is done, it sends an OK also, so 2 can now go ahead.

A Token Ring Algorithm

- An unordered group of processes on a network → A logical ring constructed in software.
- Use ring to pass right to access resource.

A Comparison of the Four Algorithms

<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>Decentralized</td>
<td>(3m_k, k = 1, 2, \ldots)</td>
<td>2 (m)</td>
<td>Starvation, low efficiency</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>

Today’s Lecture

- Lamport Clocks
- Vector Clocks
- Mutual Exclusion
- Election
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.

The Bully Algorithm (1)

- (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.

The Bully Algorithm (2)

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

A Ring Algorithm
Elections in Wireless Environments

- node a as the source
- The build-tree phase

Reporting of best node to source.

Important Lessons

- Lamport & vector clocks both give a logical timestamps
  - Total ordering vs. causal ordering
- Other issues in coordinating node activities
  - Exclusive access to resources
  - Choosing a single leader