

Elections

29 December 2009
Lecture 11

Slide Credits: Maarten van Steen

Topics for Today

- Causally Ordered Multicast
- Global Positioning of Nodes
- Elections

December 29, 2009

ISE 431: Distributed Information Systems

2

Vector and Lamport Clocks

Lamport Clocks

Rule 1: Each process has its own version of the **global clock**

Rule 2: Each process increments its **global clock** version when it performs an internal event or sends a message (which includes a timestamp)

Rule 3: When a process receives a message from another process it updates its **global clock** version if the **received timestamp is larger**.

Vector clocks

Rule 1: Each process has its **own clock** and a version of **every other processes' clock**.

Rule 2: Each process increments its **own clock** when it sends or receives a message.

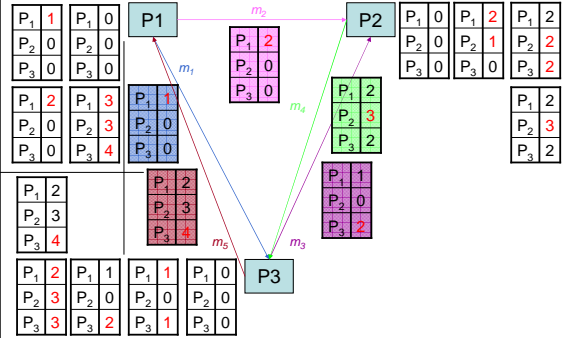
Rule 3: When a process receives a message from another process it updates its version of the **other clocks' timestamps** if the **received timestamp is larger**

December 29, 2009

ISE 431: Distributed Information Systems

3

Vector Clock Example

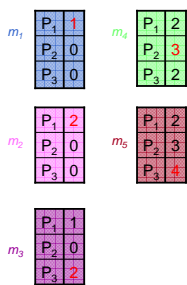


December 29, 2009

ISE 431: Distributed Information Systems

4

Vector Clock Example



1. $m_1 < m_2$
2. $m_1 < m_3$
3. $m_1 < m_4$
4. $m_1 < m_5$
5. $m_2 < m_3$
6. $m_2 < m_4$
7. $m_2 < m_5$
8. $m_3 < m_4$
9. $m_3 < m_5$
10. $m_4 < m_5$

December 29, 2009

ISE 431: Distributed Information Systems

5

Vector Clocks and COM

Vector clocks

Rule 1: Each process has its **own clock** and a version of **every other processes' clock**.

Rule 2: Each process increments its **own clock** when it **sends or receives** a message.

Rule 3: When a process receives a message from another process it updates its version of the **other clocks' timestamps** if the **received timestamp is larger**

Causally Ordered Multicast

Rule 1: Each process has its **own clock** and a version of **every other processes' clock**.

Rule 2: Each process increments its own clock when it **sends** a message.

Rule 3: When a process receives a message from another process it updates its version of the **sender's timestamp**.

- Rule 4:** A message is delivered only if it is "next in line":
1. It's the next expected one for the sender
 2. The message's timestamp is less than or equal to the local clock.

December 29, 2009

ISE 431: Distributed Information Systems

6

Causally Ordered Multicast

December 29, 2009 ISE 431: Distributed Information Systems 7

Causally Ordered Multicasting (1/2)

Observation: We can now ensure that a message is delivered only if all causally preceding messages have already been delivered.

Adjustment: P_i increments $VC_i[i]$ only when sending a message, and P_j "adjusts" $VC_j[i]$ when receiving a message (i.e., effectively does not change $VC_j[i]$).

P_j postpones delivery of m until:

- $ts(m)[i] = VC_j[i] + 1$.
- $ts(m)[k] \leq VC_j[k]$ for $k \neq i$

December 29, 2009 ISE 431: Distributed Information Systems 8

So Far

- Causally Ordered Multicast
- Global Positioning of Nodes
- Elections

December 29, 2009 ISE 431: Distributed Information Systems 9

Global Positioning of Nodes

Problem: How can a single node efficiently estimate the latency between **any two other nodes** in a distributed system?

Solution: construct a **geometric overlay network**, in which the distance $d(P, Q)$ reflects the actual latency between P and Q .

December 29, 2009 ISE 431: Distributed Information Systems 10

Computing Position (1/2)

Observation: a node P needs $k + 1$ landmarks to compute its own position in a k -dimensional space. Consider two-dimensional case:

Solution: P needs to solve three equations in two unknowns (x_p, y_p) :

$$d_i = \sqrt{(x_i - x_p)^2 + (y_i - y_p)^2}$$

December 29, 2009 ISE 431: Distributed Information Systems 11

Computing Position (2/2)

Problems:

- measured latencies to landmarks fluctuate
- computed distances will not even be consistent:

Solution: Let the L landmarks measure their pairwise latencies $d(b_i, b_j)$ and let each node P minimize

$$\sum_{i=1}^L \left[\frac{d(b_i, P) - \hat{d}(b_i, P)}{d(b_i, P)} \right]^2$$

where $\hat{d}(b_i, P)$ denotes the distance to landmark b_i given a computed coordinate for P .

December 29, 2009 ISE 431: Distributed Information Systems 12

Computing Position Example

$d(P_1, P_2) = 12ms$
 $d(P_1, P_3) = 6ms$
 $d(P_3, P_2) = 20ms$
 $\hat{d}(P_1, P_2) = 10ms$
 $\hat{d}(P_1, P_3) = 30ms$
 $\hat{d}(P_3, P_2) = 23ms$

$d(P_1, P_2) = 12ms$
 $d(P_1, P_3) = 6ms$
 $d(P_3, P_2) = 20ms$
 $\hat{d}(P_1, P_2) = ?$
 $\hat{d}(P_1, P_3) = ?$
 $\hat{d}(P_3, P_2) = ?$

December 29, 2009 ISE 431: Distributed Information Systems 13

Computing Position Example

$d(P_1, P_2) = 10ms$
 $d(P_1, P_3) = 6ms$
 $d(P_3, P_2) = 20ms$

December 29, 2009 ISE 431: Distributed Information Systems 14

Computing Position Example

$d(P_1, P_2) = 10ms$
 $d(P_1, P_3) = 6ms$
 $d(P_3, P_2) = 20ms$

$\hat{d}(P_1, P_2) = 10$
 $\hat{d}(P_1, P_3) = 6$
 $\hat{d}(P_2, P_3) = \sqrt{(0-6)^2 + (10-0)^2} = \sqrt{136} = 11.66$

Error calculations:
 $\left[\frac{10-10}{10}\right]^2 + \left[\frac{6-6}{6}\right]^2 + \left[\frac{20-11.66}{20}\right]^2 = 0.174$

December 29, 2009 ISE 431: Distributed Information Systems 15

Computing Position Example

$d(P_1, P_2) = 10ms$
 $d(P_1, P_3) = 6ms$
 $d(P_3, P_2) = 20ms$

$\hat{d}(P_1, P_2) = 10$
 $\hat{d}(P_1, P_3) = 6$
 $\hat{d}(P_2, P_3) = 16$

Error calculations:
 $\left[\frac{10-10}{10}\right]^2 + \left[\frac{6-6}{6}\right]^2 + \left[\frac{20-16}{20}\right]^2 = 0.04$

December 29, 2009 ISE 431: Distributed Information Systems 16

Computing Position Example

$d(P_1, P_2) = 10ms$
 $d(P_1, P_3) = 6ms$
 $d(P_3, P_2) = 20ms$

$\hat{d}(P_1, P_2) = 10$
 $\hat{d}(P_1, P_3) = 10$
 $\hat{d}(P_2, P_3) = 20$

Error calculations:
 $\left[\frac{10-10}{10}\right]^2 + \left[\frac{6-10}{6}\right]^2 + \left[\frac{20-20}{20}\right]^2 = 0.444$

December 29, 2009 ISE 431: Distributed Information Systems 17

So Far

- Causally Ordered Multicast
- Global Positioning of Nodes
- Elections

December 29, 2009 ISE 431: Distributed Information Systems 18

Election Algorithms

Principle: An algorithm requires that some process acts as a coordinator. The question is how to select this special process **dynamically**.

Note: In many systems the coordinator is chosen by hand (e.g. file servers). This leads to centralized solutions → single point of failure.

Question: If a coordinator is chosen dynamically, is it centralized? Distributed?

Question: Is a fully distributed solution, i.e. one without a coordinator, always more robust than any centralized/coordinated solution?

December 29, 2009

ISE 431: Distributed Information Systems

19

Election by Bullying (1/2)

Principle: Each process has an associated priority (weight). The process with the highest priority should always be elected as the coordinator.

Issue: How do we find the heaviest process?

- Any process can just start an election by sending an election message to all other processes (assuming you don't know the weights of the others).
- If a process P_{heavy} receives an election message from a lighter process P_{light} , it sends a take-over message to P_{light} . P_{light} is out of the race.
- If a process doesn't get a take-over message back, it wins, and sends a victory message to all other processes.

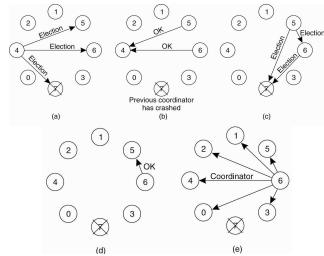
December 29, 2009

ISE 431: Distributed Information Systems

20

Election by Bullying (2/2)

Question: We're assuming something very important here – what?



If you have broadcast – just broadcast to everyone.

December 29, 2009

ISE 431: Distributed Information Systems

21

Election in a Ring

Principle: Process priority is obtained by organizing processes into a (logical) ring. Process with the highest priority should be elected as coordinator.

- Any process can start an election by sending an election message to its successor. If a successor is down, the message is passed on to the next successor.
- If a message is passed on, the sender adds itself to the list. When it gets back to the initiator, everyone had a chance to make its presence known.
- The initiator sends a coordinator message around the ring containing a list of all living processes. The one with the highest priority is elected as coordinator.

Question: Does it matter if two processes initiate an election?

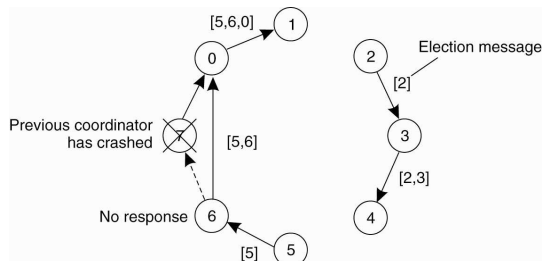
Question: What happens if a process crashes during the election?

December 29, 2009

ISE 431: Distributed Information Systems

22

Election in a Ring



December 29, 2009

ISE 431: Distributed Information Systems

23

Superpeer Election

Issue: How can we select superpeers such that:

- Normal nodes have low-latency access to superpeers
- Superpeers are evenly distributed across the overlay network
- There is a predefined fraction of superpeers
- Each superpeer should not need to serve more than a fixed number of normal nodes

DHT: Reserve a fixed part of the ID space for superpeers. **Example:** if S superpeers are needed for a system that uses m -bit identifiers, simply reserve the $k = \lceil \log_2(S) \rceil$ leftmost bits for superpeers. With N nodes, we'll have, on average, $2^{k-m} \times N$ superpeers.

Routing to superpeer: Send message for key p to node responsible for p AND $\underbrace{11\dots1}_{k}00\dots00$

December 29, 2009

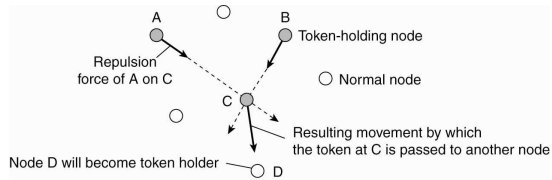
ISE 431: Distributed Information Systems

24

Superpeer Placement

Issue: How can we place **superpeers** so that they are evenly spaced?

- Distribute N tokens to N random nodes
- Each node with a token checks to see if there are nearby tokens (using a gossip protocol)
- If yes, the tokens exert a repulsive force, causing the holder of the token to pass it on to another node further away.



December 29, 2009

ISE 431: Distributed Information Systems

25

Conclusion

- Causally Ordered Multicast
- Global Positioning of Nodes
- Elections

December 29, 2009

ISE 431: Distributed Information Systems

26