Time, Clocks, and the Ordering of Events in a Distributed System

By: Leslie Lamport
Presented by: Konstantin Berlin

http://www.cs.umd.edu/~kberlin/cmsc714/cmsc714pres.ppt

Introduction
- How do you assign order to events happening in a distributed system?
  - No central clock.
  - Theoretically impossible to fully synchronize clocks.
  - Decentralized

Outline
- Introduction
- The Partial Ordering
- Logical Clocks
- Total Ordering of Events
- Physical Clocks
- Applications
- Conclusion

The Partial Ordering
- Definition: "happened before", denoted as \( \rightarrow \)
  1) If \( a \) and \( b \) are events in the same process, and \( a \) comes before \( b \), then \( a \rightarrow b \).
  2) If \( a \) is the sending of a message by one process and \( b \) is the receipt of the same message by another process, then \( a \rightarrow b \).
  3) If \( a \rightarrow b \) and \( b \rightarrow c \) then \( a \rightarrow c \).
  4) \( a \rightarrow a \) for any event \( a \).

Logical Clocks
- Define clock \( C_{i,a} \) on processor \( P_i \) as function that assigns number to event "\( a \)".
  - \( C_i: a \rightarrow N_0 \)
  - Define \( C_{i,a} = C_{i,b} \) if \( a \) is event on \( P_i \).
  - Clock Condition:
    - For all \( a,b \): if \( a \rightarrow b \), then \( C_{i,a} < C_{i,b} \)

Logical Clocks: Satisfying clock condition
- IR1. Each process \( P_i \) increments \( C_i \) between any two successive events.
  - IR2.
    - (a) If event \( a \) is the sending of a message \( m \) by process \( P_i \), then the message \( m \) contains a timestamp \( T_m = C_i(a) \).
    - (b) Upon receiving a message \( m \), process \( P_j \) sets \( C_j \) greater than or equal to its present value and greater than \( T_m \).
Total Ordering of Events

Using logical clocks it is simple to produce a total ordering of events (⇒)

- a⇒b if and only if either
  i. C< a> < C< b>
  ii. C(a) = C(b) and P_i < P_j.
- a→b implies a⇒b

Outline

- Background
- The Partial Ordering
- Logical Clocks
- Total Ordering of Events
- Physical Clocks
- Applications
- Conclusion

Strong Clock Condition

- What happens when some communication is out of band?
  - Remember IR2 (b) condition
  - Upon receiving a message m, process P_i sets C_i greater than or equal to C_j's present value and greater than T_m.
  - Because message was sent out of band, it is possible that a→b, but C< a> > C< b>.
- Strong Clock Condition
  - Let a→b, as a happening before b.
  - For any events a, b in L, if a→b then C< a> < C< b>.

Physical Clock

- Introduce a physical clock
  - Accurate
    - There exists a constant κ << 1, such that for all i: |dC_i(t)/dt - 1| < κ
  - Synchronized
    - For all i, j: |C_i(t) - C_j(t)| < ε
  - Avoid anomalous behavior
    - For any i, j, t: C_i(t+μ) - C_i(t) > 0. Where μ is the transmission speed.
    - ε/(1 - κ) ≤ μ must hold

Physical Clock: Synchronization

- When P_i sends a message m at physical time t to P_j, m contains a timestamp T_m = C_i(t).
- Upon receiving a message m at time t', process P_j sets C_j(t') = max(C_j(t' - 0), T_m + min delay)
- Theorem states that given the bounds on maximum number of hops and if messages are send frequently enough, synchronization condition
  - For all i, j: |C_i(t) - C_j(t)| < ε holds

Applications

- Granting exclusive right to a resource
  - Use logical clocks to assign ordering to requests (done individually at each process)
  - Move on to next task as soon as got confirmation of a release.
- Updates in Peer-to-Peer network
Conclusion

- Questions?