

Time and Global States

Distributed Systems

A collection of independent (autonomous) computers that appear as a single coherent system

- E.g., world wide web, distributed file systems
- Properties
 - Transparency
 - Scalability
 - Availability
 - Reliability
 - Serviceability (manageability)
 - Safety
- Issues: communication, synchronization, consistency, fault tolerance

Time: Physical Clocks

A precisely machined quartz crystal that oscillates at a well-defined frequency when kept under tension

- Timer: add a counter and a holding register; decrement counter at each oscillation; when 0, generate interrupt and reload counter from holding register
- Each interrupt called one clock tick
- Problem: clock skew

Physical Clock Synchronization Algorithms

- Cristian's algorithm – request time from server
- Berkeley algorithm – server requests time and sends back average
- Averaging algorithm – distributed algorithm, uses broadcast

Logical Clocks

Agreement on ordering of events rather than what time it is is what matters

- Lamport Timestamps: partial order
 - "happened-before" relation " \rightarrow ", causal ordering, or potential causal ordering
 - Transitive relation
 - Assign every event a time value $C(a)$ such that if $a \rightarrow b$ then $C(a) < C(b)$ – can be captured numerically through a monotonically increasing software counter
 - Lamport's solution for message ordering
 - Each message carries sending time
 - Receiver's clock is set to the greater of its own clock or the sender's clock and then incremented by 1
 - Between every two events, the clock must tick at least once

Example Application: Totally Ordered Multicast

- Use Lamport's solution to advance each processor's logical clock
- Send each message to each processor
- Acknowledge each message and send ack to each processor in the multicast group

Vector Timestamps

- Lamport timestamps do not capture causality
- Solution: vector timestamps: captures the notion of causality in addition to concurrency
 - Maintain a vector clock with n integers for n processes

Global State Determination

- Example uses: deadlock or termination detection
- Solution: distributed snapshot (Chandy and Lamport 1985)
 - Important property: consistent global state/cut
 - Termination detection: snapshot in which all channels are empty

Election Algorithms

- Election of a leader or coordinator
 - Bully algorithm
 - Ring algorithm

Mutual Exclusion

- Centralized
 - Distributed
 - Token-based
- Performance measured in terms of
- Bandwidth (proportional to number of messages)
 - Delay at each entry and exit
 - Throughput

Comparison of Mutual Exclusion Algorithms

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

Transactions

- Modify multiple data items potentially at multiple locations/by multiple processes as a single atomic operation
- Transaction properties (ACID) –
 - Atomic – happens indivisibly to the outside world
 - Consistent – does not violate system invariants – must hold before and after but not necessarily during
 - Isolated (or serializable) – refers to multiple simultaneous transactions – the final result must appear as if each transaction occurred in some sequential order
 - Durable – once committed, the results become permanent – no failure can undo the results

Classification of Transactions

- Flat – series of operations satisfying ACID properties
- Nested – transaction logically divided into sub-transactions
- Distributed – data is distributed (transaction could be flat)

Transaction Implementation

- Private workspace
 - Operations performed on private copy of all open files
- Writeahead log
 - Modify in place but write a log of transaction id, old, and new values BEFORE doing so

Concurrency Control

- Synchronize conflicting read and write operations to ensure serializability
 - Two-phase locking
 - Pessimistic and optimistic timestamp ordering

Reliable Multicast

- Scalable reliable multicast
 - Return negative acknowledgements as feedback
 - Schedule a feedback message with some random delay
- Hierarchical feedback control

Reliable Multicast in the Presence of Failures

- Virtual synchrony
 - Split up time into epochs with group membership G
 - A message m is delivered to every member of the group G before there is a view change allowed
 - If the sender crashes, message m can be delivered to all processes in the group or to none

Message Ordering

- Unordered multicast
- FIFO-ordered multicast
- Causally-ordered multicast
- Totally-ordered multicast
- Total order + virtual synchrony = atomic multicast

Distributed Commit

- 2-Phase commit
- 3-Phase commit