# Fault Tolerance

## CS239
## Advanced Operating Systems

## November 5, 1998
## Dr. Richard Guy



# Lecture Source Materials

Singhal, Ch. 13: **Fault Tolerance**
**IEEE Computer,** April, 1997

Sources of Failure in the PSTN, p. 31-6
(also http://hissa.nist.gov/kuhn/pstn.html)
Understanding FT & Reliability, p. 45-50
Beyond Fault Tolerance, p. 47-49
Toward Systematic Design of FT Systems, p. 51-8
Software-based Replication for FT, p. 68-74



# Additional Source Materials

- papers by Jim Gray
  - Why do computers stop and what can be done about it?, August, 1986. (5th SRDSDS)
  - High-Availability Computer Systems, September, 1991. (IEEE Computer)

## Outline

- Introduction
- Atomicity and Commit/Abort
- Commit Protocols
- Voting Protocols
- Failure-resilient Processes
- Reliable Communication
- Summary



## Prologue

- Failure causes, revisited
  - Gray's 1985 paper says people are the problem, so keep them out of the loop...
  - Kuhn's 1997 paper says people are the problem yet human intervention is key to reliability



## Introduction

- Failure recovery takes time, is disruptive.
- Fault tolerant systems either
  - mask failures
  - exhibit well-defined failure behavior
- Redundancy
  - processes
  - hardware
  - data

# Failure implications

- Process deaths
  - reclaim resources
  - unblock waiting clients/release server
- Machine failure
  - sometimes indistinguishable from process death
- Network failure
  - network partitions
  - machine/network failure often indistinguishable



# Atomicity and Commit/Abort

- Atomicity
  - indivisible, instantaneous, non-interruptable

| Process P1 | Process P2 |
|---|---|
| ---- | ---- |
| Lock (x); | Lock (x); |
| x := x + y; | x := x + z; |
| Unlock(x); | Unlock(x); |
| ---- | ---- |
| failure | |



# Atomic action

- extends machine instruction atomicity to
  - sequence of instructions
  - group of processes
- basic building block of FT operations
- what are characteristics of atomic action?

## Action is atomic if...

- process performing action is unaware of other processes, and vice versa
- process performing action doesn't talk to other processes while performing action
- process can't detect other state changes, and hides own state changes until completion
- action is considered by all other processes to be indivisible, instantaneous



## Transaction

- Transaction
  – groups sequence of actions
  – group is treated as atomic action
  – decides at some point to commit or abort
- Commit: unconditional guarantee that transaction will complete (permanent)
- Abort: unconditional guarantee that none of transaction pieces will persist



## Global atomicity

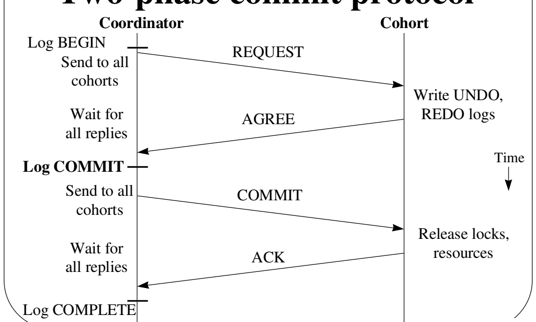
- atomic action spread over multiple players
- enforced by Commit Protocols
  – used with local recovery strategies
  – "well defined failure behavior" FT class

# Commit Protocols

- Blocking
  - Two-phase commit
- Nonblocking
  - Three-phase commit

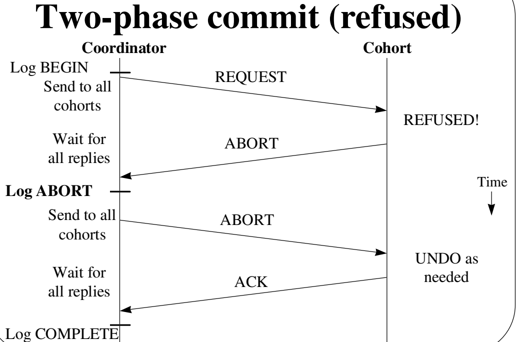


# Two-phase commit protocol

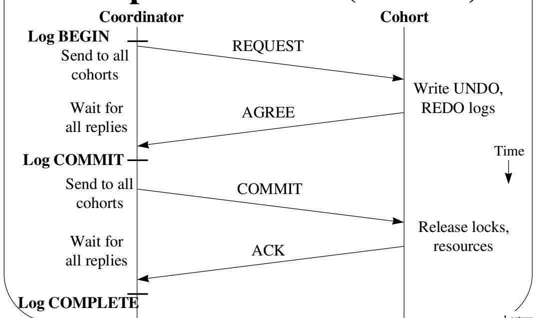





# Two-phase commit (refused)



## Two-phase commit (failures)






## Nonblocking Commit Protocols

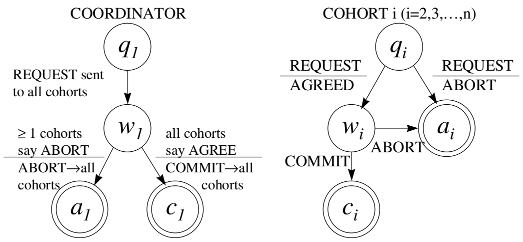
- If 2PC coordinator fails, cohorts blocked until recovery of coordinator
  - cohorts may hold locks for long time
  - 2PC not very resilient to site failures
- Blocking avoidable if sites can self-recover
- Communications assumptions
  - reliable network; point-to-point guaranteed
  - site failure detectable/reportable by network (!)



## Synchronous protocols

- Must find out why 2PC can’t self-recover
- Use finite state machines to model protocol
- Use “synchronous within 1 transition” concept to keep all players “synchronized”

# FSM model of 2PC



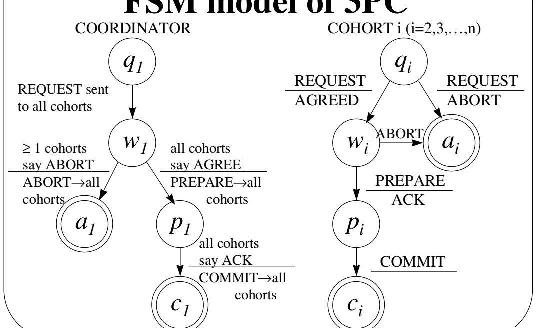



# Synchronous protocols, cont.

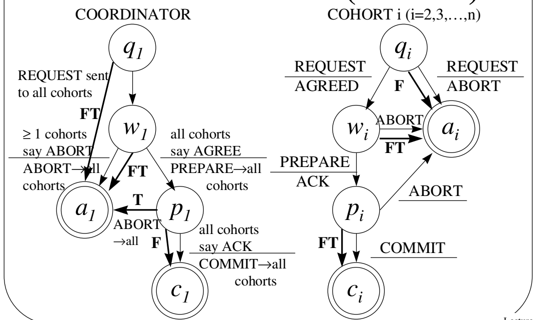
- *Concurrency set* for each site/state
  - 2PC has concurrency sets with commit/abort
  - a protocol which avoids concurrency sets with both commit and abort allows independent recovery of a single failed site



# FSM model of 3PC



## FSM model of 3PC (failures)






## Independent Recovery notes

- 3PC protocol is for single-site failure IR
- Multi-site failure IR is proven impossible
- Nonblocking protocol impossible when
  - network partition with lost messages
  - multiple network partitions
- Conclusions
  - 3PC good for common case (single failure)
  - expensive in rounds, messages vs. 2PC



## Voting Protocols

- Mask failures in a system, especially >1
- Voting tolerates
  - network partitions, site failures, lost messages
- Static
- Dynamic
  - Majority-based
  - Vote reassignment

# Static Voting

- System model
  - one replica per site
  - file access requires lock grant
    - single writer, no readers
    - multiple readers, no writers
  - each replica has version number
    - monotonically increasing
    - atomically incremented with replica update
    - kept on stable storage



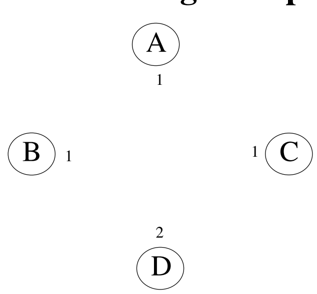
# Static Voting, cont.

- Basic idea
  - each replica is assigned a set number of votes
  - read or write permitted if requesting process collects proper number of votes
    - total votes $v$
    - read quorum $r$
    - write quorum $w$
    - $r + w > v$
    - $w > v/2$



# Static voting example

$v = 5$
$w = 3$
$r = 3$



## Static voting, cont.

- Vote assignment properties
  - write quorum prevents simultaneous writes in distinct partitions
  - nonnull intersection of all read/write quorums → current copy available in read quorum
  - no obsolete copy chosen for write update
  - always current subset whose votes sum to *w*



## Static voting protocol

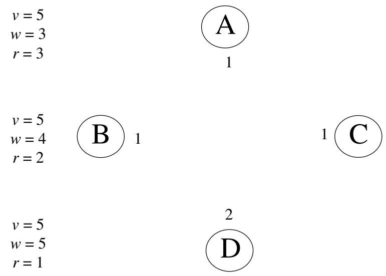
- Initiator asks local lock manager for lock
  - on grant, send Vote_Req to all replicas
- Receiver asks local lock manager for lock
  - on grant, send <version number,votes> back
- Initiator decides if quorum achieved
  - if read, and sum of votes $\geq r$, have quorum
  - if write, and sum of votes $\geq w$, have quorum



## Static voting protocol, cont.

- If initiator fails to achieve quorum
  - send Release_lock to all paticipants (+ self)
- If initiator achieved quorum
  - ensure local replica is current
  - use local copy for read/write request
  - if write
    - update version number
    - transmit new version number, version to all
  - release all locks, both local and remote

## Static voting flexibility






## Dynamic Voting

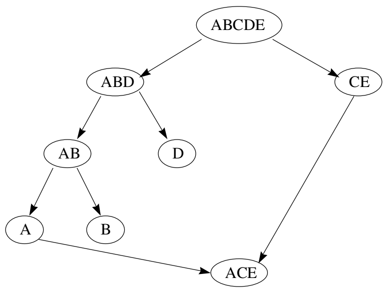
- Motivation
  - multiple failures may prevent getting quorums
  - dynamic protocols adapt to system state
    - adjust quorum size, vote assignment
- Majority-based
- Vote reassignment



## Majority-based dynamic voting

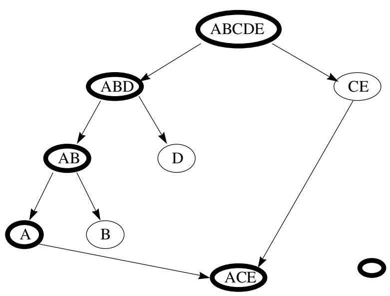
- Adjust set of sites forming majority
  - from operable communicating current replicas
- Partition graph represents network history
- Use history to determine new quorum
  - based on previous quorum

# Partition graph






# Majority concept



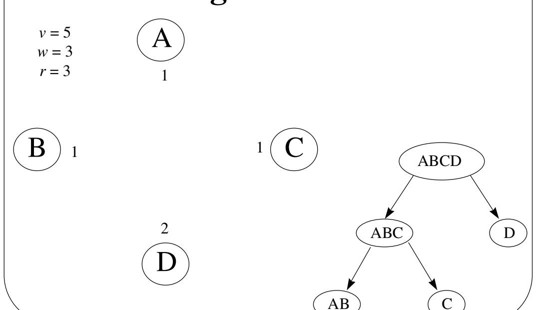

Assume 1 vote per replica

Distinguished partition

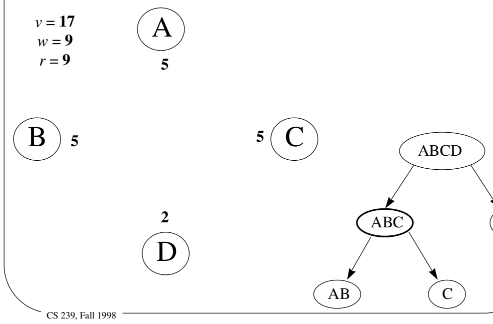


# Vote Reassignment Motivation

$v = 5$
$w = 3$
$r = 3$



## Dynamic Vote Reassignment




CS 239, Fall 1998 — Lecture 10 Page 37

## Dynamic Vote Reassignment

- Group consensus
  - majority group sites run agreement protocol
  - yields high quality vote assignment
- Autonomous reassignment
  - majority group members individually select new votes for self
  - then, get approval from majority
  - complex version vector-based protocol
  - possibly quicker than group consensus

CS 239, Fall 1998 — Lecture 10 Page 38

## Failure-resilient Processes

- Minimal disruption of service provided by process in event of system failure
  - resilient process masks failures and guarantees progress despite set number of system failures
- Backup processes
- Replicated execution

CS 239, Fall 1998 — Lecture 10 Page 39

## Backup processes

- Primary, backup processes
- Checkpoint from primary to backup
- Few system resources for inactive backup
- Delays during checkpointing
- Delay after failure
  - detect failure
  - backup recomputation from checkpoint onward
- Multiple backups must select primary



## Replicated Execution

- Concurrent execution
- Can increase reliability, availability
  - reliability: majority consensus among results
  - availibility: use any result
- Main cost is multiple CPU investment
- Nonidempotent operations (random, I/O)
- Message ordering problems among copies
- Communications with "outside world"



## Reliable Communication

- Assume replicated data, per site manager
- To update, process sends identical messages
- Suppose process fails after sending
  - Manager gets updates, then learns of failure
  - Manager learns of failure, then gets update
  - Manager neither gets updates nor learns failure
- How to get identical manager behavior?

## Reliable communication, cont.

- If two conditions met, have identical view
  - messages received are identically ordered
  - all/none message delivery
- Atomic broadcast
  - Two-phase protocol



## Atomic Broadcast

- Phase one
  - sender multicasts message
  - receiver buffers message; replies with priority
- Phase two
  - sender chooses highest priority from receivers
  - sender multicasts priority
  - receiver assigns priority; adds message to delivery queue in priority order
- Receiver-initiated recovery protocol



## Summary

- Fault tolerance prevents disrupted service
  - well-defined failure behavior
  - mask failure
- Commit protocols
  - blocking/nonblocking
- Voting protocols
  - static/dynamic
- Resilient processes, communication