Replication

Outline

- Failure Models
- Mirroring
- Quorums

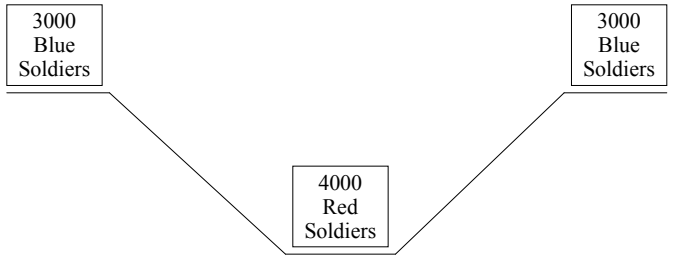
Why Replicate?

- Performance
  - keep copy close to remote users
  - caching is a special case
- Survive Failures
  - availability: provide service during temporary failure
  - fault tolerance: provide service despite catastrophic failure

Fault Models

- Crashed
  - failed device doesn't do anything (i.e., fails silently)
- Fail-Stop
  - failed device tells you that it has failed
- Byzantine
  - failed device can do anything
  - adversary
    - playing a game against an evil opponent
    - opponent knows what you're doing and tries to fool you
    - usually some limit on opponent's actions (e.g. at most k failures)

Byzantine Army Problem



Synchrony

- Assumptions concerning boundedness of component execution or network transmissions
- Synchronous
  - always performs function in a finite & known time bound
- Asynchronous
  - no such bound
- Famous Result: A group of processes cannot agree on a value in an asynchronous system given a single crash failure

Network Partitions

- Can't tell the difference between a crashed process and a process that's inaccessible due to a network failure.
- Network Partition: network failure that cuts processes into two or more groups
  - full communication within each group
  - no communication between groups
  - danger: each group thinks everyone else is dead

Mirroring

- Goal: service up to K failures
- Approach: keep K+1 copies of everything
- Clients do operations on "primary" copy
- Primary makes sure other copies do operations too
- Advantage: simple
- Disadvantages:
  - do every operation K times
  - use K times more storage than necessary

Mirroring Details

- Optimization: contact one replica to read
- What if a replica fails?
  - get up-to-date data from primary after recovering
- What if primary fails?
  - elect a new primary

Election Problem

- When algorithm terminates, all non-failed processes agree on which replica is the primary
- Algorithm works despite arbitrary failures and recoveries during the election
- If there are no more failures and recoveries, the algorithm must eventually terminate

Bully Algorithm

- Use fixed "pecking order" among processes
  - e.g., use network addresses
- Idea: choose the "biggest" non-failed machine as primary
- Correctness proof is difficult

Bully Algorithm Details

- Process starts an election whenever it recovers or whenever primary has failed
  - how know primary has failed?
- To start an election, send *election* messages to all machines bigger than yourself
  - if somebody responds with an ACK, give up
  - if nobody ACKs, declare yourself the primary
- On receiving election message, reply with ACK and start an election yourself (unless in progress)

Quorums

- Quorum: a set of server machines
- Define what constitutes a "read quorum" and a "write quorum"
- To write
  - acquire locks on all members of some write quorum
  - do writes on all locked servers
  - release locks
- To read: similar, but use read quorum

Quorums

- Correctness requirements
  - any two write quorums must share a member
  - any read quorum and any write quorum must share a member (read quorums need not overlap)
- Locking ensures that
  - at most one write happening at a time
  - never have a write and a read happening at the same time

Defining Quorums

- Many alternatives
- Example
  - write quorum must contain all replicas
  - read quorum may contain any one replica
- Consequence
  - writes are slow, reads are fast
  - can write only if all replicas are available
  - can read if any one replica is available

Defining Quorums (cont)

- Example: Majority Quorum
  - write quorum: any set with more than half the replicas
  - read quorum: any set with more than half the replicas
- Consequences
  - modest performance for read and write
  - can proceed as long as more than half the replicas are available

Quorums & Version Numbers

- Write operation writes only a subset of the servers
  - some servers are out-of-date
- Remedy
  - put version number stamp on each item in each replica
  - when acquiring locks, get current version number from each replica
  - quorum overlap rules ensure that one member of your quorum has the latest version

Version Numbers (cont)

- When reading, get the data from the latest version number in your quorum
- When writing, set version number of all replicas you wrote equal to 1 + (max version number in your quorum beforehand)
- Guarantees correctness even if no recovery action is taken when replica recovers from a crash

Quorums and Partitions

- One group has a write quorum (and thus usually a read quorum);
  - that group can do anything
  - other groups are frozen
- No group has a write quorum, but some groups have a read quorum
  - some groups can read
  - no groups can write
- No group contains any quorum
  - everyone is frozen