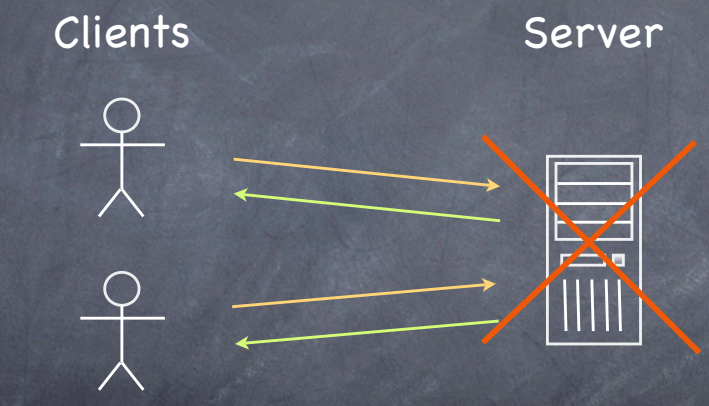
State-Machine Replication

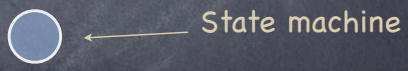
The Problem



Solution: replicate server!

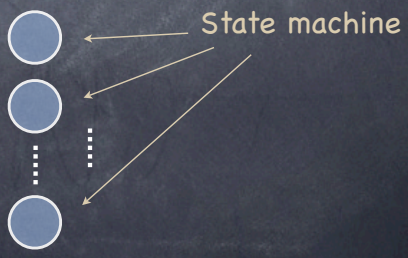
The Solution

1. Make server deterministic (state machine)



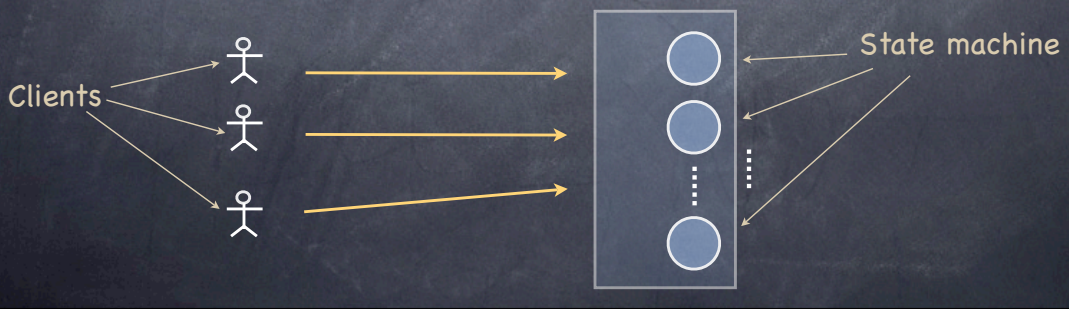
The Solution

1. Make server deterministic (state machine)
2. Replicate server



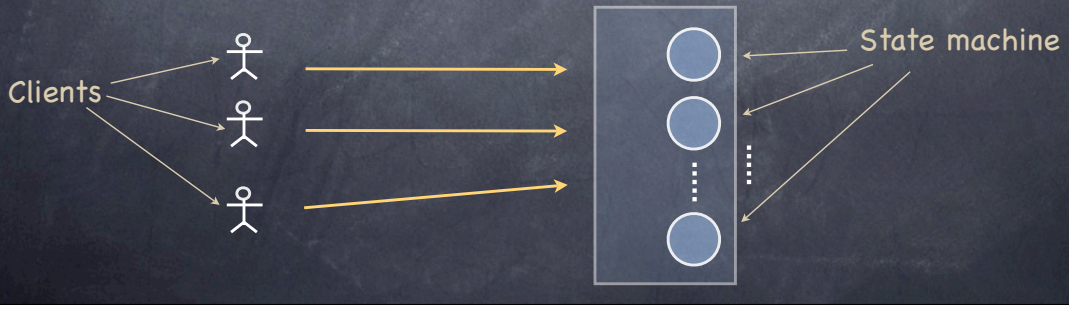
The Solution

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure correct replicas step through the same sequence of state transitions



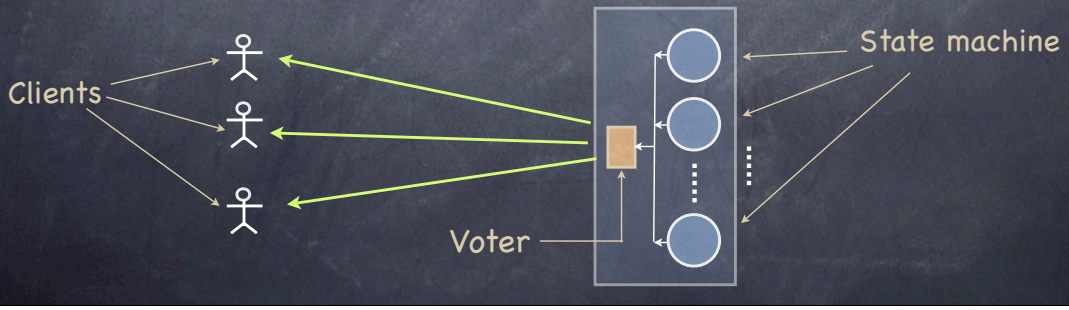
The Solution

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure correct replicas step through the same sequence of state transitions
4. Vote on replica outputs for fault-tolerance

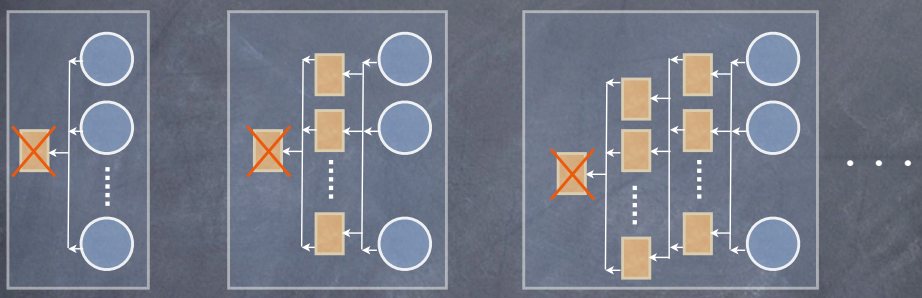


The Solution

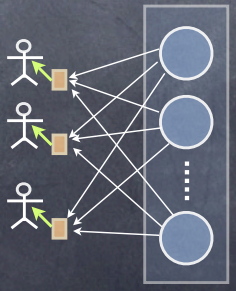
1. Make server deterministic (state machine)
2. Replicate server
3. Ensure correct replicas step through the same sequence of state transitions
4. Vote on replica outputs for fault-tolerance



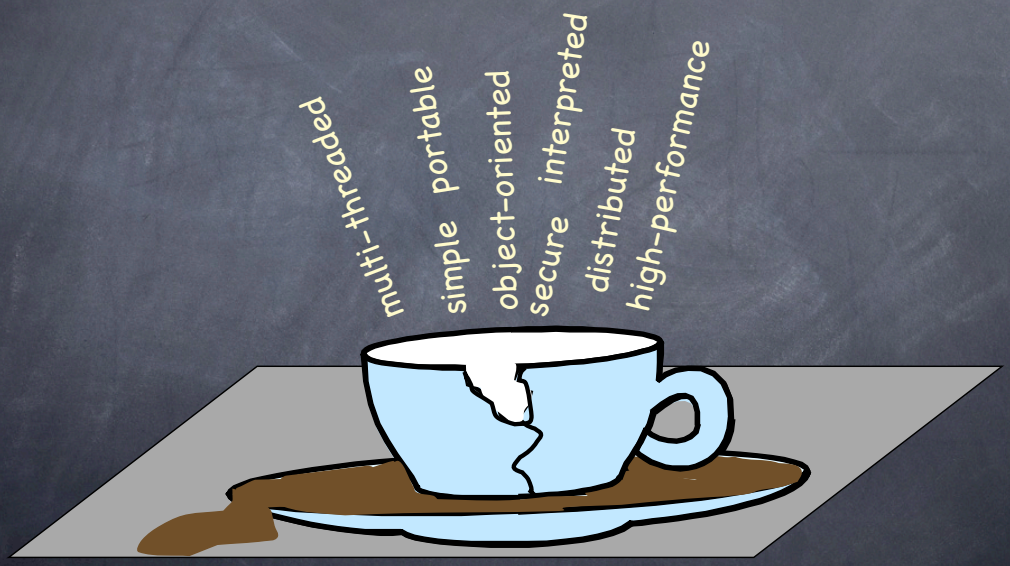
A conundrum



A: voter and client share fate!



Ahhh, Java...



Semantic Characterization of a State Machine

Outputs of a state machine are completely determined by the sequence of requests it processes, independent of time and any other activity in a system

Replica Coordination

All non-faulty state machines receive all requests in the same order

- Agreement: Every non-faulty state machine receives every request
- Order: Every non-faulty state machine processes the requests it receives in the same order

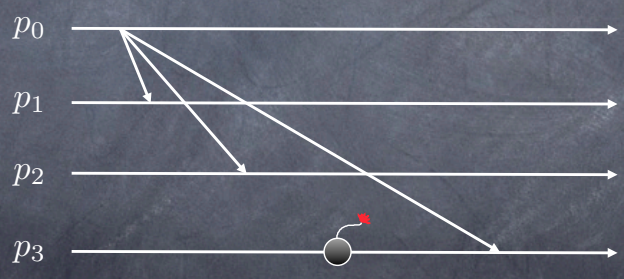
Consensus and Reliable Broadcast

Broadcast

- If a process sends a message m, then every process eventually delivers m

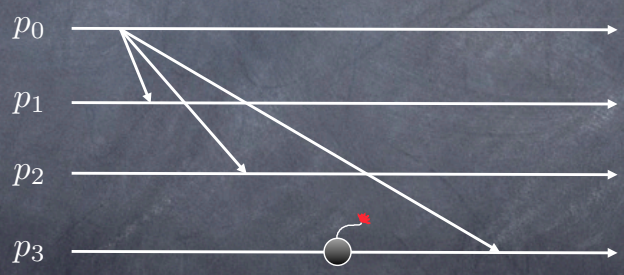
Broadcast

- If a process sends a message m, then every process eventually delivers m



Broadcast

- If a process sends a message m, then every process eventually delivers m

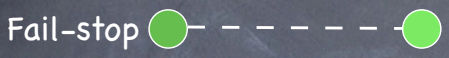


- How can we adapt the spec for an environment where processes can fail? And what does "fail" mean?

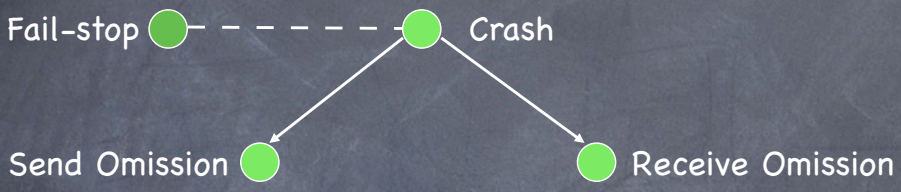
A hierarchy of failure models

- Crash

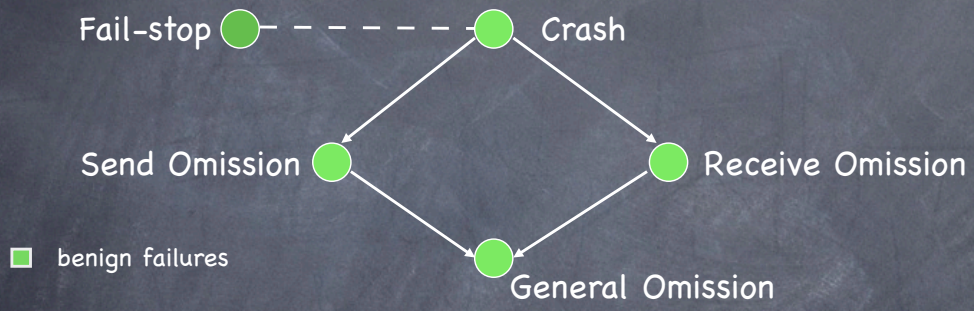
A hierarchy of failure models



A hierarchy of failure models

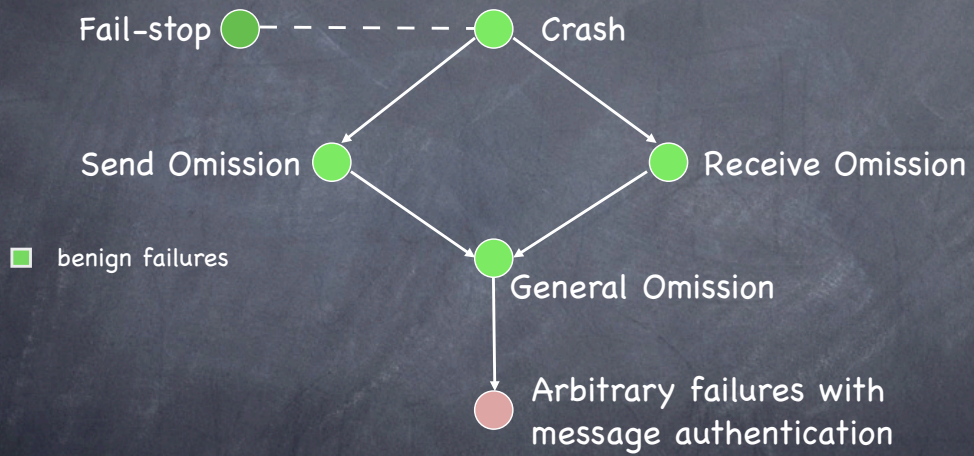


A hierarchy of failure models



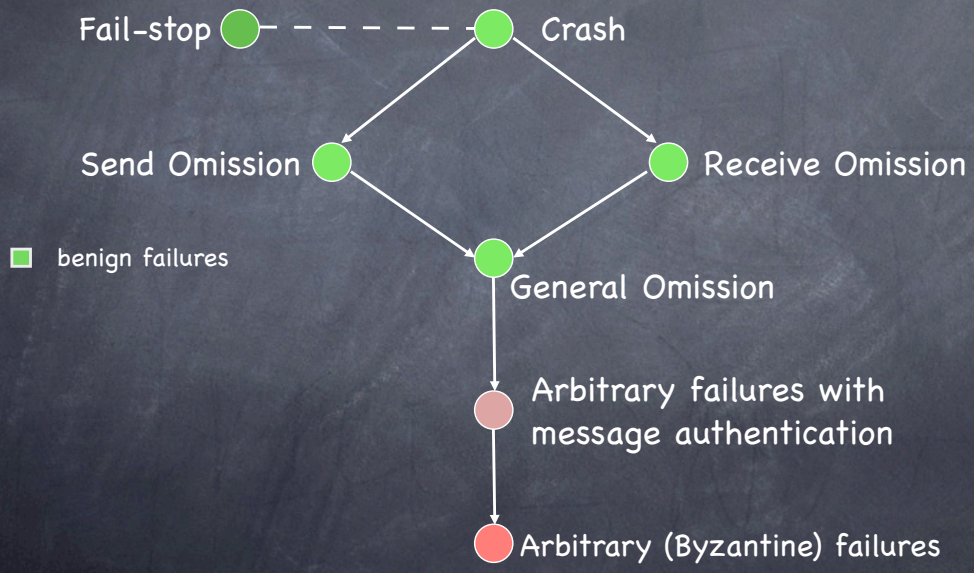
- benign failures

A hierarchy of failure models



- benign failures

A hierarchy of failure models



Reliable Broadcast

Validity If the sender is correct and broadcasts a message m, then all correct processes eventually deliver m

Agreement If a correct process delivers a message m, then all correct processes eventually deliver m

Integrity Every correct process delivers at most one message, and if it delivers m, then some process must have broadcast m

Terminating Reliable Broadcast

Validity	If the sender is correct and broadcasts a message m, then all correct processes eventually deliver m
Agreement	If a correct process delivers a message m, then all correct processes eventually deliver m
Integrity	Every correct process delivers at most one message, and if it delivers $m \neq$ SF, then some process must have broadcast m
Termination	Every correct process eventually delivers some message

Consensus

Validity	If all processes that propose a value propose v, then all correct processes eventually decide v
Agreement	If a correct process decides v, then all correct processes eventually decide v
Integrity	Every correct process decides at most one value, and if it decides v, then some process must have proposed v
Termination	Every correct process eventually decides some value

Properties of send(m) and receive(m)

Benign failures:

Validity If p sends m to q, and p , q, and the link between them are correct, then q eventually receives m

Uniform* Integrity For any message m, q receives m at most once from p, and only if p sent m to q

* A property is uniform if it applies to both correct and faulty processes

Properties of send(m) and receive(m)

Arbitrary failures:

Integrity For any message m , if p and q are correct then q receives m at most once from p, and only if p sent m to q

Questions, Questions...

- Are these problems solvable at all?
- Can they be solved independent of the failure model?
- Does solvability depend on the ratio between faulty and correct processes?
- Does solvability depend on assumptions about the reliability of the network?
- Are the problems solvable in both synchronous and asynchronous systems?
- If a solution exists, how expensive is it?