EECS 591
Distributed Systems

Manos Kapritsos
Fall 2020
State Machine Replication
MODELING FAULTS

- Mean Time To Failure/Mean Time To Recover
  - used mostly for disks
  - of questionable value in expressing reliability

- Threshold: \( f \) out of \( n \)
  - makes condition for correct operation explicit
  - measures fault-tolerance of the architecture, not of individual components

- Enumerate failure scenarios
A HIERARCHY OF FAILURE MODELS

- Fail-stop
- Crash
- Send omission
- General omission
- Receive omission
- Arbitrary (Byzantine) failures

○ = benign failures
A HIERARCHY OF FAILURE MODELS
FAULT TOLERANCE: THE PROBLEM

Solution: replicate the server
Replication in time

- When a server fails, restart it or replace it
- Failures are detected, not masked
- Lower maintenance, lower availability
- Tolerates only benign failures
Replication in space

- Run multiple copies of a server (replicas)
- Vote on replica output
- Failures are masked
- High availability and can tolerate arbitrary failures
  - but at high cost
The enemy: non-determinism

An event is non-deterministic if its output is not uniquely determined by its input

The problem with non-determinism:

- Replication in time: must reproduce the original outcome of all non-deterministic events
- Replication in space: each replica must handle non-deterministic events identically
The solution: state machines

Design the server as a deterministic state machine
The solution: state machines

State machine example: a switch

![State machine diagram](diagram.png)
**State Machine Replication**

**Ingredients:** a server

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure that all replicas go through the same sequence of state transitions
4. Vote on replica outputs
STATE MACHINE REPLICATION

Ingredients: a server

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure that all replicas go through the same sequence of state transitions
4. Vote on replica outputs

All state machines receive all commands in the same order

non-faulty

\[ x = 1 \]
**State Machine Replication**

*Ingredients: a server*

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure that all replicas go through the same sequence of state transitions
4. Vote on replica outputs
1. Evil Lorenzo Speaks French
2. And was born in Corsica
3. Went to Dartmouth instead of Cornell
4. Rides a Ducati instead of a Moto Guzzi
5. Still listens to opera, but he doesn’t care for Puccini

Evil Lorenzo thinks that 2f+1 is good enough

When in trouble, cheat!

Voter and client share fate!
ADMINISTRIVIA

• Send me your paper preferences by tonight
• Send me your group declaration preferences by Oct 1
• Homework #2 will be sent out later today
  • due Monday, Oct 12, before class
• Implementation project will be out next Monday
  • due Monday October 26, by end of day
• Research project topics due next Thursday, 10/08
Primary-Backup
THE MODEL

Failure model: crash

Network model: synchrony
  • Reliable, FIFO channels
  • All messages are delivered within $\delta$ time

Tolerates $f$ crash failures
The idea

- Clients communicate with a single replica (primary)

- Primary:
  - sequences and processes clients’ requests
  - updates other replicas (backups)

- Backups use timeouts to detect failure of primary

- On primary failure, a backup becomes the new primary
A SIMPLE PRIMARY-BACKUP PROTOCOL

\((f = 1)\)

Active replication: \(\text{sync} = \text{client request(s)}\)
Passive replication: \(\text{sync} = \text{state update}\)
Weakening the model

Failure model: crash

Network model: synchrony
- Unreliable, FIFO channels
- Channels may drop messages
- All messages are delivered within $\delta$ time
  - (looks paradoxical)

Tolerates $f$ crash failures
A slightly different primary-backup protocol $(f = 1)$
Generalizing to more backups

Primary

\( f \) backups
Generalizing to more backups

$f$ backups
GENERALIZING TO MORE BACKUPS

Primary

update

$f$ backups
GENERALIZING TO MORE BACKUPS

(active updates)

Primary

$f$ backups
GENERALIZING TO MORE BACKUPS

(passive updates)

Primary

\( f \) backups
GENERALIZING TO MORE BACKUPS
(passive updates)
GENERALIZING TO MORE BACKUPS

![Diagram showing a primary with multiple ack connections to backups.](image-url)
Generalizing to more backups

Primary

$\text{reply}$

$f$ backups
Handling queries

query

Primary

f backups
Handling queries

Primary

\[ f \text{ backups} \]
HANDLING QUERIES

reply
Primary

However...

\( f \) backups...
HANDLING QUERIES

query

Primary

$f$ backups

1  2  3  4  5  6
Handling queries

The primary cannot respond until it has received all acks for prior updates.