# Facult Tolerance via Replication

Hui Chen [a]

[a]CUNY Brooklyn College

September 17, 2025

# Computers can fail!

Let's consider a large computational cluster of 5,000 computers, what is the average number of computers failing daily?

# Computers can fail!

Let's consider a large computational cluster of 5,000 computers, what is the average number of computers failing daily?

Assuming 3% of the computers fail annually, we have:

$$5000 \times 0.03 = 150$$

So, on average, $150/365 \approx 0.41$ computers fail daily.

For real-life failure rate, check out:

PC Reliability Study – Workstation Overview (2019)

# Problem: Providing high availability despite failures

Complete the computational task even if some computers fail.

Solution: Replication . . . (meaning?)

# Any failures?

Replication may not be a solution for all failures.

# A classification of failures (Failure Model)

- Crash failure
- Omission failure
- Transient failure
- Software failure
- Security failure
- Byzantine failure
- Temporal failure

# Crash failure

A component experiences a sudden and complete loss of functionality. The failure is irreversible.

- A computer stops working, and does not respond to any request.
- A network link goes down, and no messages can be sent or received.
- A disk drive fails, and no data can be read from or written to it.

Fail-stop failure is a simple abstraction that mimics crash failure

- Implementations of fail-stop behavior help detect which component has failed.

Replication can help fail-stop failure

# Omission failure

Often seen in networks, e.g., message lost in transit, which can be the result of various causes:

- Transmitter malfunction
- Network buffer overflow
- Packet collisions (link layer)
- Wireless receiver out of range

# Transient failure

Failures occur temporarily and may resolve on their own.

- They are often caused by transient environmental conditions (arbitrary perturbation of the global state)
- Can be induced by power surge, weak batteries, lightning, radio frequency interfaces, cosmic rays etc.
- Heisenbug is a type of transient failure in software.

Transient failures can be challenging to reproduce and diagnose.

# Software failure

Software bugs are the most common cause of system failures.

- Software bugs can be introduced during development, testing, or maintenance.
- They can be caused by human error, miscommunication, or lack of understanding of the system requirements.
- Software failures can lead to system crashes, data corruption, security vulnerabilities, and other issues.

# Security failure

Security failures occur when a system is compromised by an attacker.

- They can be caused by vulnerabilities in the system, such as weak passwords, unpatched software, or misconfigured settings.
- Security failures can lead to data breaches, identity theft, financial loss, and other issues.
- They can be difficult to detect and prevent, as attackers are constantly evolving their tactics and techniques.

# Byzantine failure

Arbitrary failures that can produce any kind of erroneous behavior, including malicious behavior.

- A component may fail and then later recover, but it may not remember its previous state.
- A component may send conflicting or incorrect information to different parts of the system.
- A component may behave in a way that is inconsistent with the system's specifications or requirements.

Byzantine failures are particularly challenging to handle because they can be difficult to detect and diagnose.

# Temporal failure

A component may be correct, but it may not respond within the required time frame.

- A real-time system may fail to meet its deadlines, leading to missed opportunities or lost data.
- A distributed system may experience delays in communication or processing, leading to inconsistencies or errors.
- A user interface may become unresponsive or slow, leading to frustration or confusion.

Temporal failures can be caused by a variety of factors, including hardware limitations, software bugs, network congestion, and user behavior.

# Replication: When does it help?

Creating copies of data or services on different nodes.

Replication can help with: fail-stop failures

Replication may not help with: Byzantine Failures

*Is there a formal proof for this claim?*

# How to realize replication?

- Replicated state machine
- State transfer

# Replicated state machine

Primary/backup: clients send operations to the *primary*, the primary produces sequences of inputs/steps and sends to backups

- Each replica is a deterministic state machine.
- All replicas start in the same initial state.
- All replicas process the same sequence of inputs in the same order.
- All replicas produce the same sequence of outputs.

# State transfer

Replicate the state of a component to another component.

- Periodically transfer the entire state of a component to another component.
- Transfer only the changes made to the state since the last transfer.
- Use a combination of periodic and incremental transfers to keep the replicas up-to-date.

# Case Studies

Study the papers: Scales, Nelson, and Venkitachalam, "The design of a practical system for fault-tolerant virtual machines"; Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial"


- Scales, Daniel J, Mike Nelson, and Ganesh Venkitachalam. "The design of a practical system for fault-tolerant virtual machines". In: *ACM SIGOPS Operating Systems Review* 44.4 (2010), pp. 30–39.
- Schneider, Fred B. "Implementing fault-tolerant services using the state machine approach: A tutorial". In: *Acm Computing Surveys (CSUR)* 22.4 (1990), pp. 299–319.