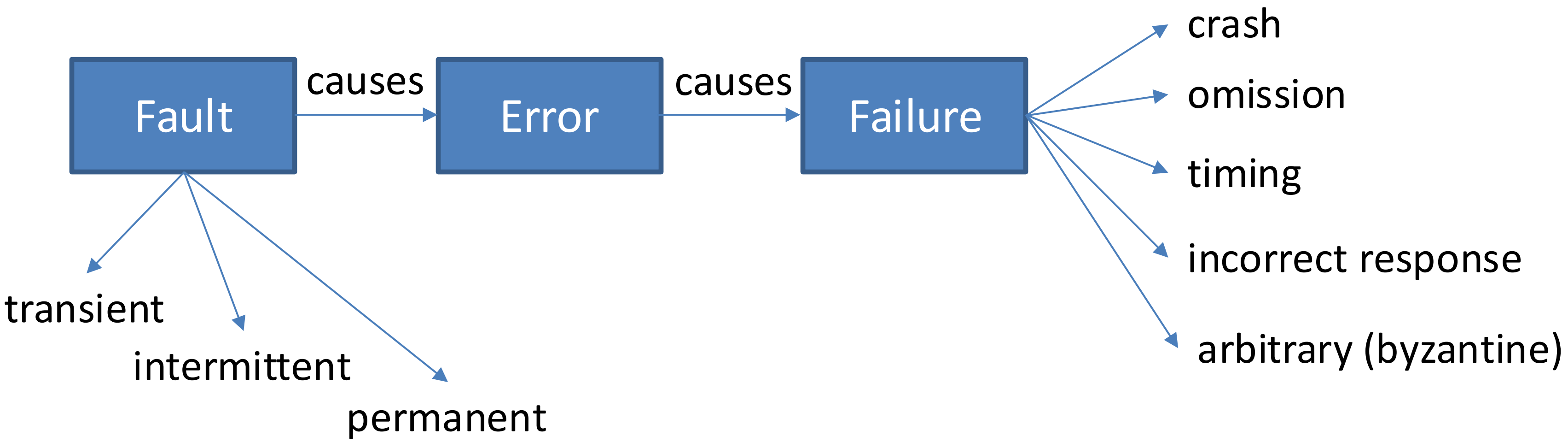
Fault Tolerance

Reference for study:

Van Steen, Tanenbaum, "Distributed Systems", chapter 8

Main Fault Tolerance Concepts



Fault causes Error causes Failure

transient

intermittent

permanent

crash

omission

timing

incorrect response

arbitrary (byzantine)

- Dependability
  - Availability, Reliability, Safety, Security, Maintainability
  - Threatened by faults
  - Can be achieved by a combination of
    - fault **prevention**, fault **tolerance**, fault **removal**, fault **forecasting**

Fault Tolerance of Distributed Systems

- Ability to continue delivering service in the presence of partial failures, possibly with degradation of service, but without seriously affecting the overall performance
  - automatically **recover** from partial failures
    - failures need to be **detected**
    - recovery procedures must be implemented
  - **continue operation** in the presence of partial failures (while waiting for **repair**), rather than stopping

Detection, Recovery and Repair Techniques

Failure Type	Detection Techniques	Recovery/Repair Techniques
crash	timeout	replacement, restart, redundancy
omission	timeout	redundancy (including retransmission)
timing	timeout	redundancy
incorrect response	input validation /comparison	redundancy
arbitrary	input validation /comparison	redundancy

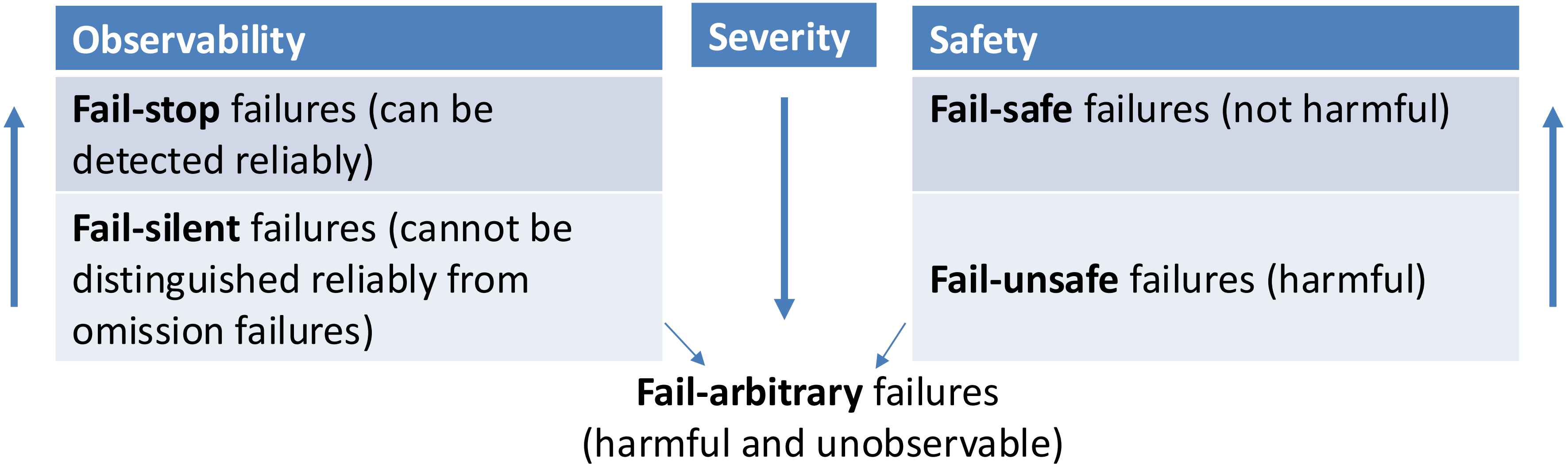
Redundancy Techniques

- Time redundancy
  - retransmit messages
  - redo operations that fail (e.g., retry an aborted transaction)
- "Physical" Redundancy
  - adding more equipment, processes or data copies
  - The most common solution: process replication (process groups)

Detecting Process Crash

- DS are asynchronous with no delay upper bound
  - A process P can detect the crash failure of another process Q by timing out when no data arrive from Q, but detections may be **erroneous** and **delayed**

- Not all crash failures have the same observability/safety/ severity:



Observability

Fail-stop failures (can be detected reliably)

Fail-silent failures (cannot be distinguished reliably from omission failures)

Severity

Safety

Fail-safe failures (not harmful)

Fail-unsafe failures (harmful)

Fail-arbitrary failures (harmful and unobservable)

Example: Detecting Crash of Process Connected via TCP

- The TCP layer itself can **know** that a connection has been closed or reset by the peer
  - FIN or RST segment received
- or **guess** that communication with the peer has been lost
  - Maximum number of retransmissions reached => Reset
- A process is informed about these conditions by the Socket API when reading or writing:

	Read	Write
connection closed	returns -1	successful
connection reset	throws SocketException	throws SocketException

Possible Crash Scenarios

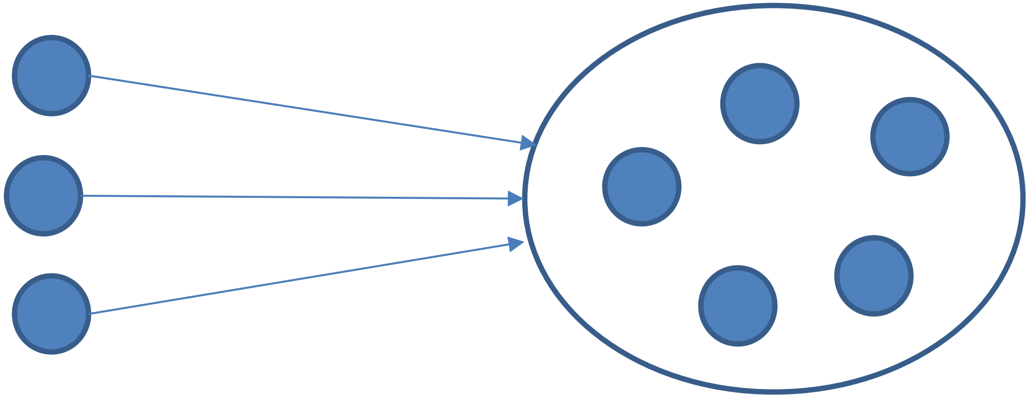
- Process crash
  - OS detects crash, TCP layer sends RST to connected peers
- Host crash (with subsequent restart)
  - when restarted, TCP will respond to segments belonging to pre-crash connections with RST
- Host crash (without subsequent restart) or permanent network disconnection
  - connected peers trying to **send** will eventually timeout (can take several minutes)
  - programmers should use **their own** timeout mechanisms (or rely on the socket timeout mechanism offered by the Socket API)

Exercise

- Introduce mechanisms to avoid deadlocks in the Echo protocol implementations

Dependability through Redundancy: Process Groups

- **Process Group**: a group of processes perceived as a single, dependable process

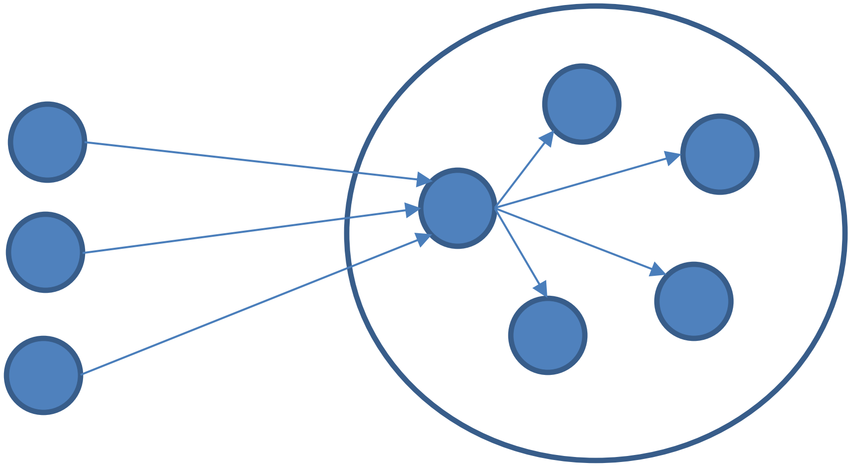


- Implementing a Process Group requires distributed algorithms (protocols)
  - for managing the group (e.g., membership, leadership, data replication and consistency)
  - for making the group fault-tolerant

Process Group Organizations

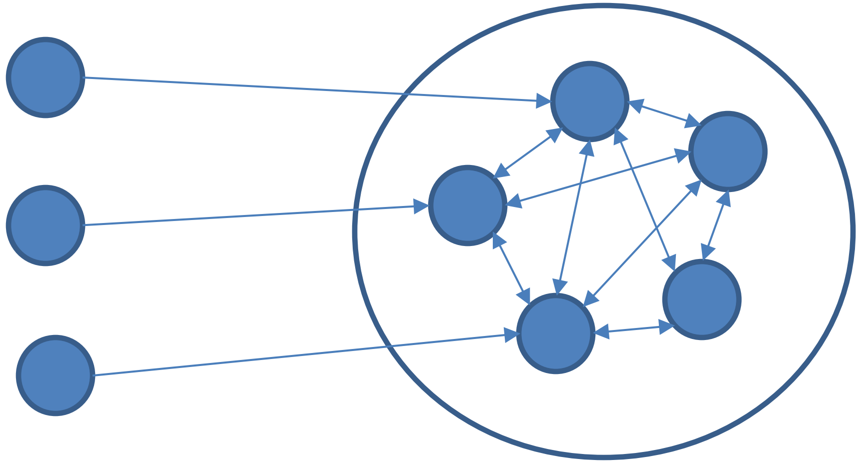
Hierarchical

(primary-based protocols)



Flat

(replicated-write, active repl. or quorum-based protocols)



Obtaining The Desired Fault-Tolerance

- A group is **k-fault tolerant** if it can withstand faults in k members
- The consensus protocols and the total number of processes necessary to achieve k-fault tolerance depend on the types of failures that are possible

Failure type	#Processes needed for k-fault tolerance	Consensus protocol Examples
crash, omission, timing, network	k+1	flooding
+incorrect response	2k+1	voting, Paxos
+arbitrary (byzantine)	3k+1	PBFT, Blockchain

The CAP (or Brewer's) Theorem

- Only 2 out of the following 3 properties can be obtained at the same time in a distributed system:
  - **C**onsistency
  - **A**vailability
  - **P**artition Tolerance
- Discussion
  - C, A, P are not on-off properties. The 2 out of 3 formulation may be misleading. A spectrum of alternatives is available.
- Readings:
  - E. Brewer. CAP Twelve Years Later: How the “Rules” Have Changed, IEEE Computer, Vol. 45, No. 2, Feb. 2012, pp. 23-29.
  - P. Bailis, and K. Kingsbury. The Network is Reliable, Communications of the ACM, Vol. 57, No. 9, Sept. 2014, pp. 48-55.