Fault tolerance

Unit-1

Fault → Static defect
Error → incorrect internal state.
Failure → unintended output.

Fault classification

- By Nature of Occurance
  - Permanent
    - always appear
    - Eg. Bug in the code.
  - Intermittent
    - Appears, disappear & reappear
    - Race condition in multithreaded code
  - Transient
    - Occure only once
    - cosmic ray filping a bit in memory.
- By Domain
  - Hardware
    - Physical defect
    - Memory chip failure
  - Software
    - logical error
    - eg div. by 0.

Redundency → Means adding some extra components, even when there is failure it still function (kind of spair tire)

Types (MITSFD)

- Hardware Redundency
  - Passive
    - Extra component runs in the bg.
  - Active
    - multiple component runs in the bg.
- Info. Redundency
  - Extra data is added to detect & correct codes
- Time Redundency
  - Ops. runs at diff times to check if o/p is consistent or not.
- Sfw Redundency
  - different code is used to achieve the same result.
- Functional
  - Extra functionalities are added in code so if ones fail it takes place
- Data Redundency
  - Storing same data in diff places.

Basic means of fault tolerance :-

1) Error Detection
2) Error Correction
3) Redundancy.
4) failover Degradation
5) Recovery & Restart
6) Monitoring & Diagnosis

#

Fault tolerance

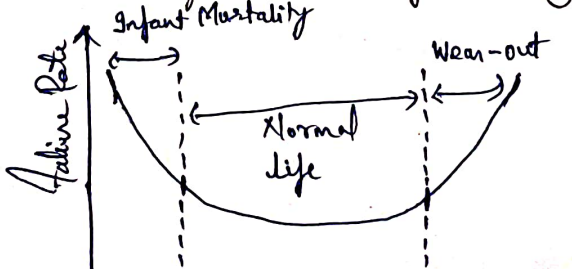
Traditional
↓
Deals with
→ fault occurance in single Machine
↓
Techniques :- (HIT)
- → Hardware Redundancy
- → Info. Redundancy
- → Time Redundancy
- → S/w Redundancy

Network fault tol.
↓
Deals with
Fault occurance in distributed sys.
↓
Techniques :- (CLR)
- → Redundant Path
- → Failover system
- ✓ → Load Balancing
- ✓ → Replication
- ✓ → Consensus Protocols.

Faliure Rate ⇒ $\lambda = \frac{\text{No. of faliure in a time interval}}{\text{Total operating time in that interval}}$ (unit: failure per million hours)

Mean time to faliure = $\frac{1}{\lambda}$

⇒ In practicle system fails by showing → Bathtub Curve of faliure Rate



Canonical Structure:—

→ Basic and Standard design model for fault tolerance sys. (including)

- → Error Detection
- → Damage Confinement
- → Recovery
- → Fault treatment.

Resilient Structure:—

→ Its a practical implementation that goes beyond the canonical model & keep acceptable service.

→ It does not work on detection & Recovery but Survivability & Adaptibility.

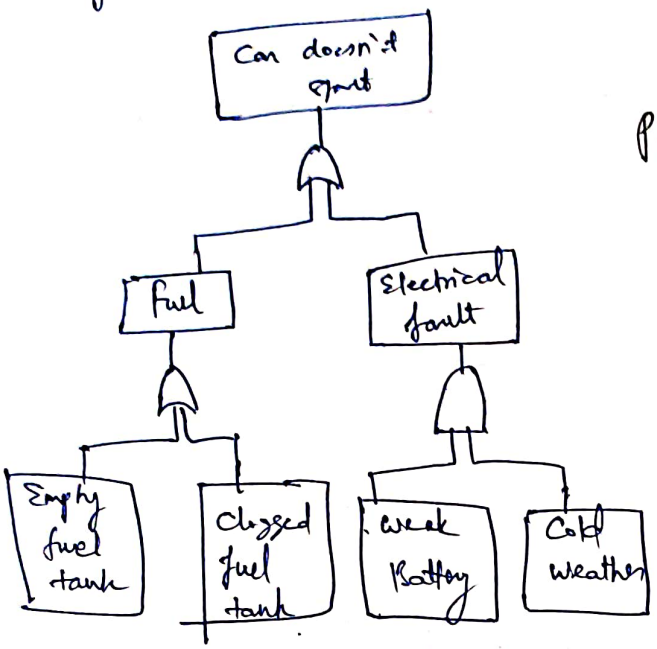
Features →
- → Redundency
- → Adaptibility
- → Graceful degradation
- → Self Healing Mechanism

Reliability Evaluation techniques:—

$$R(t) = e^{-\lambda t}$$

- Analytical Techniques
  - Reliability Block Diagram
    - ↓
    - Sys is represented with a set of blocks → Series Connection = less Reliability
    - ↳ Parallel Connection = More & reliability.
  - Fault tree Analysis
    - ↓
    - Starts with top level failure (Eg next Pg)
  - Markov Model
- Statistical Techniques
  - Life testing
    - ↓
    - Runs Comput Until they fails. → collect data → fix machine
  - Yield data Analysis
    - ↓
    - Real world data as i/p

Fault tree Analysis:



Quantitative Analysis :-

P(car doesn't start) = P(fuel) + P(Electrical fault)
+ P(Empty fuel tank) +
P(clogged fuel tank) +
P(weak Battery) +
P(Cold weather) − overlap terms

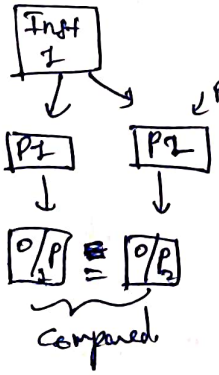
Fault Tolerance Procesor level technique :-

↳ Procesor level fault tolerance techniques are all about making sure a CPU keeps executing correctly even when Hardware fault appears.

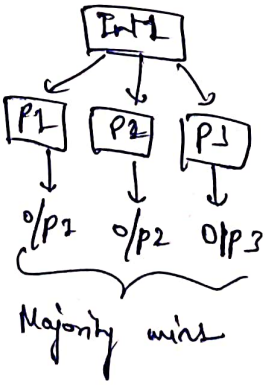
Techniques to Handle it

① Hardware Redundancy techniques

(Dual Modular Redundancy)
~~Lockstep Execution~~



Tripple Modular Redundancy



② Structural Redundancy

Spare Unit in Superscalar Procesor

Multiple → ALU, FPU, Pipelines

Redundant Multithreading

Multiple redundent threads runs

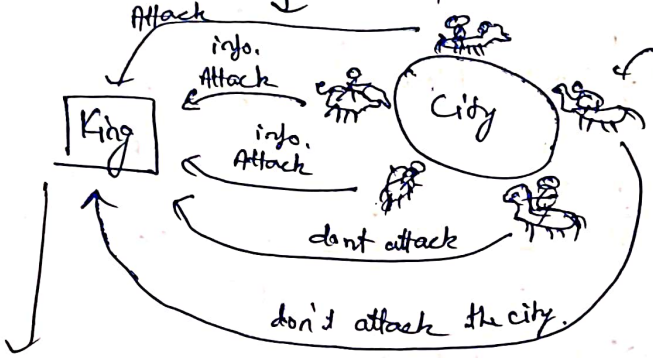
compares result.

Byzantine failure :-

→ Unlike a crash failure → where a node stops working

→ Byzantine failure forward → wrong, inconsistent or even malicious info to different part of the system.

→ Comes from Byzantine General problem



Problem King don't know who is saying the fourth, ?!

↙

Defense technique

↳ Byzantine fault tolerance :-

↳ Consensus. → Majority wins.

gt tells potential failure

FMEA → failure Mode & effective Analysis :-

ways that can cause failure

how effect ↓ Consequences of failure

Severity : (S)
↳ how serious the effect is ?

Occurrence : (O)
↳ Likelihood of failure happening

Detection : (D)
↳ likelihood of detecting failure

Risk Priority Number ↓ RPN = S X O X D

gt tells how severe the problem is.

FMECA → failure Mode & effective & criticality Analysis.

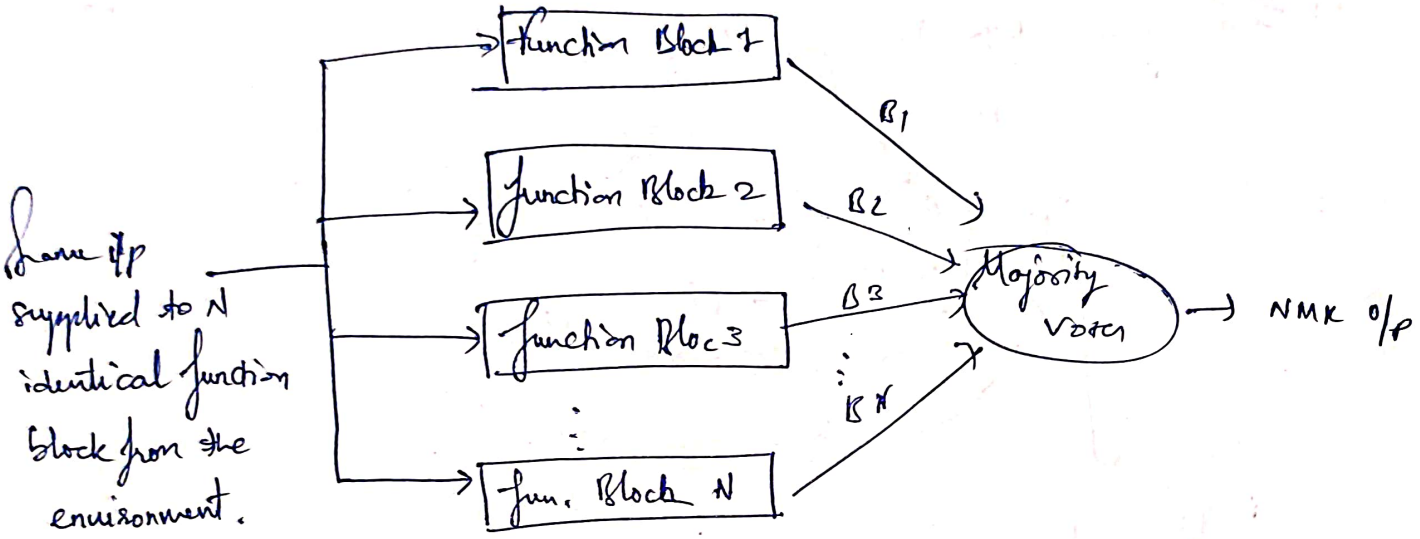
↳ FMEA + formal criticality assessment,
↳ Combines Severity & likelihood

Criticality No. = Severity X likelihood X probability of failure

[Unit : 2]

N-Modular Redundancy :- [NMR]

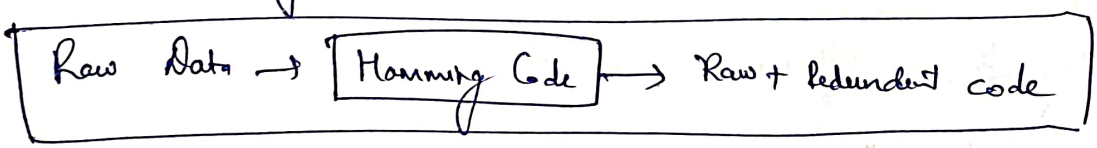
→ It's a classic fault tolerance technique that uses multiple redundant modules to improve reliability.



ECC: Error Correcting code :-

↳ ECC are used to make system fault tolerent, at data levels.

How it does?
→ Don't store/transmit raw bits → add some extra bits so you can't only detect errors but also correct them automatically.

Eg 

Usage :-

→ In Data transmission
→ In Cache & Registers.
→ Reed Solomon Code used in → HDD/SSD, CD/DVD
→ Aerospace electronic systems.

Self Purging Redundancy :—

↳ Instead of (Majority Voting) → it finds faulty modules & fixes them at instant.

Usage :—

↳ Airflight Control → SPR allow sys to keep running with healthy modules while automatically dropping the faulty ones.

↳ Fault tolerant Processors.

→ Communication systems. → Specially in Military communication.

⇒ Hybrid Redundancy technique → Mixture

Shift-out Modular Redundancy :—

↳ Instead of letting faulty module to vote

↓

if module consistently giving wrong o/p (many times)

↓

It shifted out (removed) from voting + Remove from further operation.

⇒ It Reconfiguration :—

↳ Means the system can re-organize itself dynamically after identifying a faulty module. Instead of keeping the failed module in the voting pool → the system reconfigure by :—

↓

Detect faulty module (multiple times)

↓

Purge the faulty unit

↓

Reconfigure the module except that.