Fault Tolerant Computing
(CS 530)

Yashwant K. Malaiya
Colorado State University
malaiya@cs.colostate.edu
Fall 2000

Fault-tolerant Computing

• Objective: to achieve very high reliability in computing
• How:
  – Design for high reliability
  – Test to find and remove potential faults
  – Use redundancy to tolerate faults
Fault-tolerant Computing

- **Objective:** to achieve very high reliability in computing
- **How:**
  - Design for high reliability
  - Test to find and remove potential faults
  - Use redundancy to tolerate faults

Murphy’s Law

- Anything that can go wrong, will.
  - (Actually not by Murphy but by Finagle)
- To every law there is an exception.
- Cs530 laws:
  - Anything that can go wrong, it eventually will, but
    - It may not go wrong for a while
    - It may not go wrong the next time
    - Only one thing may go wrong at a time
Correct Operation

- Good software
- Good hardware
- Correct human effort

Network: connectivity, packets
Data-base: bad data

Fault Avoidance Vs. Tolerance

- **Fault avoidance**: eliminate problem sources
  - Radiation shielding
  - Testing and debugging
  - Robust design
- **Fault tolerance**: add redundancy to mask effect
  - Error correction coding
  - Backup storage
  - Spare tire
Terminology

- Latent fault: which has not yet produced error
- Latent error: which has not yet produced failure

Origin of Defects in Objects (hw/sw)

- Good object wearing out with age
  - Hardware only?
- Incorrect maintenance/operation
- Good object, unforeseen hostile environment
  - Environmental fault
- Marginal object: occasionally fails in target environment
  - Tight design/bad inputs
- Implementation mistakes
- Specification mistakes
Need for fault tolerance: Universal & Basic

- Redundancy in ordinary text
- Clotting of blood: self repair
- Duplication of eyes: redundancy?
- Fat deposits in body
- 4 legs of a table
- Coin op machines: check for bad coins

Fault Characteristics

- Cause (a previous slide)
- Nature:
  - Software
  - Hardware
    - Digital
    - Analog
- Duration:
  - Permanent
  - Temporary
    - Intermittent: marginal system
    - Transient: environmental
    - Permanent
Mission Specific Approaches

- High availability systems:
  - Telephone
  - Transaction processing: banks/airlines

- Long life missions:
  - Unscheduled maintenance too costly
  - Long outages, manual reconfiguration OK
  - Critical applications

- Critical applications:
  - Real-time industrial control
  - Flight control

- Ordinary applications:
  - CDs: encoding
  - Internet: packet retransmission

Fault Handling

- Fault detection: is a fault there?
- Fault location: where?
- Fault diagnosis: which fault it is?
- Fault containment: blocking error flow
- Fault recovery: back to correct operation
- Fault masking: fault has no effect
Reliability Measures

• Failure rate: fraction of units failing/unit time
  – 1000 units, 3 failed in 2 hours
  – Failure rate=3/1000x2=1.5x10^{-3} per hour
• Mean time to failure (MTTF): expected time before unit fails
  – Inverse of failure rate
• Availability: probability that system is operational at time t
  – Fraction of time system is operational
• “Reliability”=probability system will survive to time t

Common Terms

• Dependability: combination of several measures
• Safety: probability that system is operating correctly or has failed in a safe manner.
• Performability: combination of reliability & performance
• Testability: ease of detecting presence of a fault
• Maintainability: ease of repairing a system after failure
System Response to Faults

- Error on output: acceptable in non-critical systems
- Fault masking: output correct even when fault from a specific class occurs
  - Critical applications: air/space/manufacturing
- Fault-secure: output correct or error indication
  - Retryable: banking, telephony, payroll
- Fail safe: output correct or in safe state
  - Flashing red traffic light, disabled ATM

Redundancy

- Spatial (hardware)
  - Replication
  - Encoding (low level)
- Temporal (time)
  - Rollback and retry
  - Encoding
  - ARQ
- Procedural:
  - Checking (small overhead)
  - Software redundancy: n-version
  - Design verification
Redundancy (Cont.)

- Analog
  - Use of slack or margin
  - Ex: allow for extra delays due to temp rise
- Information (already included)
  - Spatial
  - Temporal
- Exact classification is sometimes hard
- Disadvantages:
  - Overhead
  - Difficulty of testing
  - Unmanaged/excessive redund.: increase unreliability