

Fault Tolerant Computing

CS 530

Lecture Notes 1

Introduction

Yashwant K. Malaiya
Colorado State University



August 25, 2008

1

What We Will Study

- **Essential terms**
- **How defects arise**
- **Fault taxonomies**
- **Fault handling**
- **Reliability attributes and measures**
- **Redundancy types**
- **Deterministic vs probabilistic approaches**

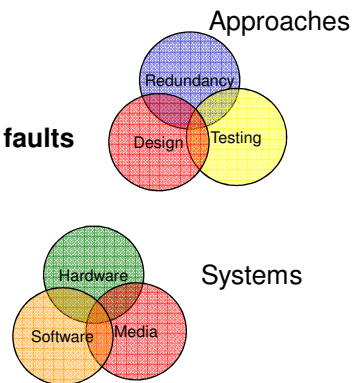


August 25, 2008

2

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
- **In hardware, software & media**
 - Some approaches are common
 - Some not



About this course

- **Texts (and courses) generally focus on**
 - Reliability or Testing or Redundancy
 - Hardware or software
- **This course attempts to address**
 - different aspects of highly reliable computing
 - Relationships among Reliability, Testing and Redundancy
 - Similarities and differences between hardware & software issues
- **No single book used. Study based on**
 - Course notes
 - Articles (including some by instructor)
 - Various sources

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

Reliability: increasing concern

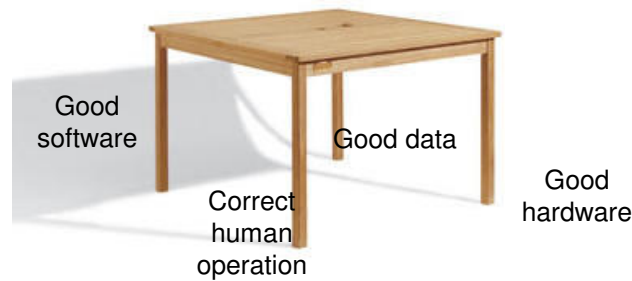
Historical

- High reliability in computers was needed in critical applications: space missions, telephone switching, process control etc.

Contemporary

- Extraordinary dependence on computers: on-line banking, commerce, cars, planes, communications etc.
- Hardware is increasingly more fault-prone
- Software is increasingly more complex
- Things simply will not work without special reliability measures

Correct Operation in Computing



These are the system components. All are needed for proper operation

Reliability approaches: Fault Avoidance Vs. Tolerance

- **Fault avoidance: eliminate problem sources**
 - Remove defects: Testing and debugging
 - Robust design: reduce probability of defects
 - Minimize environmental stress: Radiation shielding etc
 - Impossible to avoid faults completely
- **Fault tolerance: add redundancy to mask effect**
 - Additional resources needed (more later)
 - **Examples:**
 - Error correction coding
 - Backup storage
 - Spare tire etc

Terminology

```

graph LR
    A["Defects  
(physical)"] --> B["Errors  
(information)"]
    B --> C["Failures  
(application)"]
    subgraph Internal_to_system [Internal to system]
        A
        B
    end
    subgraph External [external]
        C
    end
    
```

- **Latent fault: which has not yet produced error**
 - Faulty component will produce error only when used by a process.
- **Latent error: which has not yet produced failure.**
 - An infected person may not show symptoms of a disease.
- **Unfortunately terminology is not standard.**
 - You need to ensure you have understood author's intent.

Colorado State University
August 25, 2008
9

Origin of Defects in Objects

(in hardware or software)

Increasing
human
responsibility

- **Good object wearing out with age**
 - Hardware (software can *age* too)
 - 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**

Colorado State University
August 25, 2008

Object: refers to a piece of hardware or software

10

Fault Taxonomies

- **Cause** (a previous slide)
- **Nature:**
 - Software
 - Hardware
 - Digital: causing a change in binary (logic) behavior
 - Analog: Ex: high supply current
- **Duration of the fault:**
 - Permanent: You have to throw away the unit
 - Temporary
 - Intermittent: marginal system: Ex: a loose connection
 - Transient: environmental: Ex: charged particles causing soft errors
 - Permanent with repair: repair makes the fault go away

Why We Need High Reliability?

- **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 but widespread applications:**
 - CDs: encoding
 - Internet: packet retransmission

What to do about faults

Finding & identifying faults:

- **Fault detection:** is a fault there?
- **Fault location:** where?
- **Fault diagnosis:** which fault it is?

Automatic handling of faults

- **Fault containment:** blocking error flow
 - Fault masking: fault has no effect
- **Fault recovery:** back to correct operation

Reliability Measures: formal defs

- **Failure rate:** fraction of units failing/unit time
 - 1000 units, 3 failed in 2 hours
 - Failure rate = $3/1000 \times 2 = 1.5 \times 10^{-3}$ per hour
- **Mean time to failure (MTTF):** expected time before unit fails
 - Corresponds to inverse of failure rate
- **“Reliability”** = probability system will survive to time t
- **Availability:** probability that system is operational at time t
 - Corresponds to fraction of time system is operational

Common Reliability Attributes 1

- **Dependability**: combination of several measures
- **Safety**: attribute of a system which either operates correctly or fails in a safe manner.
 - “Fail-safe”: ex: traffic light blinks red upon failure
- **Performability**: combination of reliability & performance
 - “Graceful degradation”: loss of performance due to minor failures

Some of the terms are not defined in a way to be quantifiable.



Common Reliability Attributes 2

- **Security**: authentication, confidentiality, integrity etc.
- **Survivability**: combination of dependability and security
- **Testability**: ease of detecting presence of a fault
 - Controllability and observability
- **Maintainability**: ease of repairing a system after failure

Quantitative measures for testability have been proposed, but not widely accepted.

System Response to Faults

- **Error on output:** may be acceptable in non-critical systems if happens only rarely
- **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**

Need for fault tolerance: Universal & Basic

Natural objects:

- **Fat deposits in body:** survival in famines
- **Clotting of blood:** self repair
- **Duplication of eyes:** graceful degradation upon failure

Man-made objects

- **Redundancy in ordinary text**
- **Asking for password twice during initial set-up**
- **Duplicate tires in trucks**
- **Coin op machines: check for bad coins**

Redundancy

- **Spatial (hardware) Redundancy**
 - Replication (higher level)
 - Encoding (low level)
- **Temporal (time) Redundancy**
 - Rollback and retry
 - Encoding
 - Retransmission in networks (ARQ)
- **Procedural Redundancy**
 - Checking (small overhead)
 - Software redundancy: n-version
 - Design verification



Redundancy (Cont.)

- **Analog Redundancy**
 - Use of slack or margin,
 - Ex: allow for extra delays in chips due to temp rise
- **Information (or Data) Redundancy: already included in**
 - Spatial (Ex: bus with 8 bits + 1 bit parity) or
 - Temporal (Ex: packet transmitted serially, with parity bit at the end)
- **Exact classification is sometimes hard**
- **Disadvantages:**
 - Overhead
 - Difficulty of testing
 - Unmanaged/excessive redundancy: increase unreliability

Fault-tolerant Computing

- **Deterministic approaches**
 - Based on simplifying assumptions: “fault model”
 - Obtain methods using the models: test generation
 - Evaluation of effectiveness
 - Used for Testing & combinatorial fault-tolerance
- **Probabilistic approaches**
 - We can't predict exactly when a person will die, but we can get “life expectancy = 77.2 years”, if we have data
 - Used for evaluating, achieving and optimizing reliability
 - Random testing

Course Topics

Testing

- Fault-modeling, test generation
- Testability and black-box testing

Reliability

- Permanent and temporary faults
- Replication and retry
- Pursuit of ultra-reliability

Software reliability

- Defects, factors, reliability growth
- Reliability strategies

Other topics

References

http://hissa.nist.gov/chissa/SEI_Framework/framework_1.html

A Conceptual Framework for System Fault Tolerance

- A detailed introduction to Fault Tolerance

<http://www.eventhelix.com/RealtimeMantra/FaultHandling>

Fault Handling and Fault Tolerance

- Introduction to how fault tolerance is achieved

<http://rodin.cs.ncl.ac.uk/Publications/avizienis.pdf>

Dependability And Its Threats: A Taxonomy" by Algirdas Avizienis, Jean-Claude Laprie, B. Randell

- Advanced intro by distinguished researchers