Today’s Lecture

- Definition, metrics, and terminology

**fault-tolerant**

*adj* : able to function in the absence of a major component
Goals of Fault Tolerant Systems

- How can we deal with problems?
- Option 1: Make problems less likely
  - Tough to do!
  - Testing and design for test (DFT) can help avoid physical defects
  - Careful design reviews can help to avoid design bugs
  - Training and practice can help to avoid operator error
- Option 2: Fail, but don’t corrupt anything
  - Example: ATM should shut down instead of passing out money
- Option 3: Transparently tolerate problems
  - Use hardware and/or software to mask fault effects
  - Key: use redundancy (a.k.a. spares or backups)
  - Example: having a co-pilot on an airplane

Reliable Computing System

- Correct outputs
  - Desired performance, power consumption
- Changing/varying environmental conditions
  - Power supply, radiation, noise
- Manufacturing process conditions
  - Defects, process variation
- Design errors
Reliability approaches

- 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
    - Occurrence of failures minimized
- Fault tolerance: add redundancy to mask effect
  - Failures during system operation
  - Recovery & repair
  - Examples:
    - Error correction coding
    - Backup storage
    - Spare tire

System View of Dependable Computing
How do We Achieve the Objectives?

- Applications
  - Application program interface (API)
  - Middleware
- SIFT
- Reliable communications
- Operating system
- Hardware
  - System network
  - Processing elements
  - Memory
  - Storage system

- Checkpointing and rollback, application replication, software voting (fault masking), process pairs, robust data structures, recovery blocks, N-version programming
- CRC on messages, acknowledgment, watchdogs, heartbeats, consistency protocols
- Memory management, detection of process failures, hooks to support software fault tolerance for application
- Error correcting codes, N_of_M and standby redundancy, voting, watchdog timers, reliable storage (RAID, mirrored disks)

Dependable Computing

- Original definition of dependability (that stresses the need for justification of trust) states that: the dependability is the ability to deliver service that can justifiably be trusted.
- The alternate definition (that provides the criterion for deciding if the service is dependable) states that: the dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable.

- Attributes
  - Availability: Readiness for correct service
  - Reliability: Continuity of correct service
  - Safety: Absence of catastrophic consequences
  - Confidentiality: Absence of unauthorized disclosure of data
  - Integrity: Absence of improper system alteration
  - Maintainability: Ability to undergo modifications and repairs

- Means
  - Fault prevention
  - Fault tolerance
  - Fault removal
  - Fault forecasting

- Threats
  - Faults
  - Errors
  - Failures
Dependable Systems

Intuitive Concepts

- Reliability – continues to work
- Availability – works when I need it
- Safety – does not put me in jeopardy
- Performability - combination of reliability & performance
  - “Graceful degradation”: loss of performance due to minor failures
- Maintainability - ease of repairing a system after failure
- Testability - ease of detecting presence of a fault
- **Survivability** – will the system survive catastrophic events?
Something is wrong…

- Defect
  - Distortion of the physical shape

- Fault
  - Logical model of defects

- Error
  - Incorrect signal values/state/information in computation

- Failure
  - Deviation from designed characteristics
  - Observed malfunction during operation
  - Loss of intended function

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.
Something is wrong…

- **Fault**: abstraction of physical defect or bug to structural level
- **Error**: effect of a physical defect, bug
- **Failure**: malfunction of the system, breakdown

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
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

Fault Cycle & Dependability Measures

- **Reliability**: a measure of the continuous delivery of service; \( R(t) \) is the probability that the system survives (does not fail) throughout \([0, t]\); expected value: \( \text{MTTF} (\text{Mean Time To Failure}) \)
- **Maintainability**: a measure of the service interruption \( M(t) \) is the probability that the system will be repaired within a time less than \( t \); expected value: \( \text{MTTR} (\text{Mean Time To Repair}) \)
- **Availability**: a measure of the service delivery with respect to the alternation of the delivery and interruptions \( A(t) \) is the probability that the system delivers a proper (conforming to specification) service at a given time \( t \); expected value: \( E\{A\} = \text{MTTF} / (\text{MTTF} + \text{MTTR}) \)
- **Safety**: a measure of the time to catastrophic failure \( S(t) \) is the probability that no catastrophic failures occur during \([0, t]\); expected value: \( \text{MTTCF} (\text{Mean Time To Catastrophic Failure}) \)
Typical Recovery Latencies for a Hierarchical Fault Tolerant Design

First some probabilities...

- For each random variable X,
  - cumulative distribution function (CDF): \( F(x) = P(X \leq x) \)
  - Probability \( P \) that event \( X \) is less than or equal to value of \( x \)
  - Probability mass function (PMF): \( F(x) = P(X = x) \)
  - Probability density function (PDF): \( f(x) = \frac{dF}{dx} \)
    - Such that in general \( P(a \leq x \leq b) = \int_a^b f(x) \, dx \)
  - Mean or Expected value: \( E[X] = \int_{-\infty}^{+\infty} x f(x) \, dx \)
  - Variance: \( \sigma_X^2 = E[(x - E[X])^2] \)
Probability of Failure

- Random variable $T$ is time to the next failure
- Lifetime of a module (time until it fails)
- $F(t) = \text{Prob} \{ T \leq t \}$
- Probability that component will fail before or at time $t$
- $f(t) = \frac{dF(t)}{dt}$, $\int_0^\infty f(t)dt = 1$, $f(t) \geq 0$ (for all $t \geq 0$)
- The momentary rate of probability of failure at time $t$
- $F$ and $f$ are related through:
  \[ f(t) = \frac{dF(t)}{dt} \quad F(t) = \int_0^t f(s)ds \]

Reliability $R(t)$

- Probability that the system has been operating correctly and continuously from time 0 until time $t$, given that it was operating correctly at time 0
- $R(t) = \text{Prob} \{ T > t \} = 1 - F(t)$
- MTTF: Mean Time To Failure
- Expected value of the lifetime $T$
  \[ MTTF = E[T] = \int_0^\infty t \cdot f(t)dt \]
- With $\frac{dR(t)}{dt} = -f(t)$, follows:
  \[ MTTF = -\int_0^\infty \frac{dR(t)}{dt} \cdot dt = -tR(t) \bigg|_0^\infty + \int_0^\infty R(t)dt = \int_0^\infty R(t)dt \]
Failure Rate $\lambda$

- Number of failures per time unit w.r.t. number of surviving components
  - Also known as hazard function, $z(t)$
  - $\lambda(t) = z(t) = \frac{dF(t)}{dt} = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{R(t)}$

- A module has a constant failure rate if and only if $T$ has an exponential distribution
  
  $\begin{align*}
  R(t) &= e^{-\lambda t} ; F(t) = 1 - e^{-\lambda t} ; R(0) = 1 \\
  f(t) &= \lambda e^{-\lambda t}
  \end{align*}$

Failure Rate $\lambda(t) = \lambda$

- $MTTF = \int_0^\infty t \cdot e^{-\lambda t} dt = \int_0^\infty e^{-\lambda t} dt = \frac{1}{\lambda}$

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

- **Availability A(t)**
  - Fraction of time system is operational during the interval \([0,t]\)
  - Excludes time for recovery or repair
- **MTTR: Mean Time To Repair**
- **MTBF: Mean Time Between Failures**
  - \(MTBF = MTTF + MTTR\)

\[
A = \frac{E[\text{Uptime}]}{E[\text{Uptime}] + E[\text{Downtime}]} = \frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF}
\]

\(i_0\) \hspace{1cm} \(MTTF\) \hspace{1cm} \(MTBF\) \hspace{1cm} Time

**Other failure distribution models**

- **Weibull distribution**
  - \(\alpha\) : shape parameter
    - \(\alpha < 1\) : failure rate decreasing with time
    - \(\alpha = 1\) : failure rate constant
    - \(\alpha > 1\) : failure rate increasing with time
  - \(\lambda\) : scale parameter
  - PDF = \(f(t) = \alpha \lambda (\lambda t)^{\alpha-1} e^{-(\lambda t)\alpha}\)
  - CDF = \(F(t) = 1 - e^{-(\lambda t)\alpha}\)
  - Reliability = \(R(t) = e^{-(\lambda t)^\alpha}\)
Other failure distribution models

- Geometric distribution
  - Discrete times 0, 1, 2, …
  - Replacing $e^{-\lambda}$ by discrete probability $q$
  - Replacing $t$ by $n$
  - PMF = $f(n) = q^n - q^{n+1} = q^n(1 - q)$
  - CDF = $F(n) = 1 - q^n$
  - Reliability = $R(n) = q^n$
  - $\mu = \frac{1}{1-q}$, $\sigma = \frac{q^{1/2}}{1-q}$
- Discrete Weibull distribution

Maintainability

- MTTR may be subdivided as follows
  - Time needed to detect a fault and isolate the responsible components (diagnosis)
  - Time needed to replace the faulty component
  - Time needed to verify that the fault has been removed and the system is fully operational
- Design for maintainability
  - System design which supports efficient fault detection, isolation and repair
Performability

- Accomplishment levels \(L_1, L_2, \ldots, L_n\) defined in the application context
  - Representing a level of quality of service delivered by the application
  - E.g.: \(L_i\) indicates \(i\) system crashes during mission time
- Performability is a vector \((P(L_1), P(L_2), \ldots, P(L_n))\)
  - \(P(L_i)\): Probability that the system performs well enough that the application reaches level \(L_i\)