CSCE 5760: Design For Fault Tolerance

Will meet once a week: Tuesdays 4-6:30pm in D208A

URL to Class Webpage: csrl.cse.unt.edu/kavi/CSCE5760

Review:
Why do we need fault free operation—A summary

Critical Applications

- A malfunction of a computer in such applications can lead to catastrophe
- Their probability of failure must be extremely low, possibly one in a billion per hour of operation (10^{-9} probability of failure sometimes called 9 nines)

Harsh Environment

A computing system operating in a harsh environment where it is subjected to electromagnetic disturbances
particle hits and alike
Space crafts

Why is difficult to design “fault-free” systems

Complexity of systems

- Complex systems consist of millions of devices
- Every physical device has a certain probability of failure
- A very large number of devices implies that the likelihood of failures is high
- The system will experience faults at such a frequency which renders it useless

WE CANNOT GUARANTEE FAULT FREE DESIGNS OR OPERATIONS
NEED TO TOLERATE FAILURES (operate even in the presence of faults)

Fault-tolerance is the ability of a system to continue performing its function in spite of faults
CSCE 5760: Design For Fault Tolerance

A related concept, often used as a synonym for Fault Tolerance

**Dependable Computing**

- Dependability is property of computer system that allows reliance to be placed justifiably on service it delivers. The service delivered by a system is its behavior as it is perceptible by its user

**Attributes**

- Availability
- Reliability
- Safety
- Confidentiality
- Integrity
- Maintainability

**Means**

- Fault Prevention
- Fault Tolerance
- Fault Removal
- Fault Forecasting

**Impairments**

- Faults
- Errors
- Failures

Should add security to this list

---

CSCE 5760: Design For Fault Tolerance

Classification of faults

Physical Fault: Due to physical phenomena (usually hardware)

Human Fault: Design errors, operational modes

Did not anticipate inputs in real world

Hardware and Software Faults

Three types of faults:

**Transient Faults** - disappear after a relatively short time
- Example - a memory cell whose contents are changed spuriously due to some electromagnetic interference
- Overwriting the memory cell with the right content will make the fault go away

**Permanent Faults** - never go away, component has to be repaired or replaced

**Intermittent Faults** - cycle between active and benign states
- Example - a loose connection
Another classification: Benign vs malicious
- What does a benign fault mean

If a fault makes a component or system "dead" – it is benign!
  That means, fault is easily detectable
  They are easier to deal with

Malicious faults are difficult to detect – results may look normal

Propagation of faults and errors
- Faults and errors can spread throughout the system
  - If a chip shorts out power to ground, it may cause nearby chips to fail as well
- Errors can spread - output of one unit is frequently used as input by other units
  - Adder example: erroneous result of faulty adder can be fed into further calculations, thus propagating the error

Redundancy

Redundancy is at the heart of fault tolerance

Redundancy - incorporation of extra components in the design of a system so that its function is not impaired in the event of a failure

Can you think of some forms of redundancy?

We will look at different types of redundancies

- Hardware redundancy
- Software redundancy
- Information redundancy
- Time redundancy
CSCE 5760: Design For Fault Tolerance

**Hardware redundancy**
Extra hardware to help us mask or overcome failed components

**Static redundancy**: Fixed redundancy configuration
- **Example**: Use two identical units and compare results (Boeing 737 Max)
  - No match means an error (but can’t do anything about it)
- **Or** Use three processors and vote on the result. The wrong output of a single faulty processor is masked

**Dynamic hardware redundancy**: Extra hardware is used as spare components
- use spares only when an active component fails
- Standby redundancy
- Hot Spare

**Hybrid hardware redundancy**: Combination of static and dynamic

---

CSCE 5760: Design For Fault Tolerance

**Software redundancy**: How to create software redundancy?

Use multiple copies of the software (N-version programming)
- but each copy is developed by a different team of programmers
- The diversity hopefully eliminates logic or coding errors
- Should we have diversity even at specification level?

Using diversity to address security

There are other ways of improving software reliability and fault tolerance
- without relying purely on redundancy
- Can you think of some?

Proving that your code is correct (formal verification \(\rightarrow\) too complex)
- Assert pre and post condition
- Except handling
Information redundancy: Extra information to aid in tolerating failures

Could be context information
   If we state University of North Texas, Denton
   Denton can help in overcoming any errors in the name of the university

More common form of information redundancy is the use of parity
Simple parity adds one extra bit – to make the number of “1” even (or odd)
   can only detect errors – not correct

More complex parity techniques can actually detect and correct errors
   Hamming code is commonly used with memory (SEC-DED)
   Polynomial (or CRC) codes are used for message communication

Time Redundancy: Extra time to complete a task

Most common approach is timeout and retry
   Does this help?
If failures are transient, second time around, the faulty may have disappeared

Other forms of time redundancy
   Rollback and recover
   Roll forward and recover
   Timeout rebroadcast

We need to decide which form of redundancy (and other fault tolerance methods)
are suitable to achieve desired goals
How to measure fault tolerance?

If we use fault tolerance, we expect error free operation or at least increased reliability. But we may want to also use "cost" and "energy" of achieving the fault tolerance in measuring effectiveness of fault tolerance.

Let us start with Reliability as a measure.

Basic assumption: A system is either in Up or Down state.

Reliability $R(t)$: probability that the system is in UP state during the interval $[0, t]$ given that the system was UP at time 0.

What probability distribution should we use?

Will reliability distribution linear? 

$\rightarrow$ is $R(t)$ the same as $R(2t)$ if we know that the system is UP at $t$?

---

Read up on some probability distributions.

Quick review of concepts

What is a random variable?

What is a probability distribution?

Informal: Random variable is a variable whose values depend on the outcome of a random phenomenon.

The random phenomenon can be an "experiment" like tossing a coin or a natural phenomenon such as the avg high temperature in August in Denton.

So we need to define the different possible value the random variable can take:

RV(coin toss): {H, T} 

RV(Denton avg high temp): {0-115}

Now we can associate probabilities that the random variable obtain a specific value:

$\text{prob}(\text{RV (Denton avg high Temp) = 100}) = 0.75$
Probability distribution associates the probabilities of the random variable assuming different values.

If the random variable takes discrete values (coin toss), then the probabilities for these discrete values are described by discrete distributions.

Examples: Bernoulli, Binomial, Geometric, Hypergeometric, Poisson, Multinomial

Refresh yourself about some of these and understand what types of random variables have these distributions.

If the random variable takes continuous values, then we use continuous probability distributions.

Examples: Normal, Exponential, Weibul.

Probability mass function (for discrete): Probability of RV taking a specific value based on repeated experiments/observations.

Probability density function (for continuous): Probability of RV taking a value in a small range, for example,

\[ \text{Prob[avg temp in Denton between 100 and 100.01]} \]

To be more formal, the range should be very small

\[ [100, 100+\delta] \]

Stochastic Variable?

A random variable that depends on some other dimension

time: the value at a specific time

area: the value on 2/3 dimensional space

(temp at a specific pixel)
CSCE 5760: Design For Fault Tolerance

Now we can extend the probability (mass) density function usually using time dimension.

What is the probability that a device failed at time \( t \) actually between \( t \) and \( t + \delta \)? We label this with the probability density function \( f(t) \).

What is the probability it did not fail before \( t \)?

\[
= 1 - \int_{0}^{t} f(t) dt = 1 - F(t)
\]

CSCE 5760: Design For Fault Tolerance

Mean, variance and expected values

Mean or average also expected value

Discrete random variable

\[
E[x] = \sum x \cdot f(x)
\]

Continuous distribution:

\[
E[x] = \int x \cdot f(x) dt
\]

Variance

\[
V(x) = \sum (x - E(x))^2 f(x) \quad V(x) = \int (x - E(x))^2 f(x) dt
\]
Now let us relate these to Chapter 2 of Textbook

Failure Rate
- Rate at which a component suffers faults
- Depends on age, ambient temperature, voltage or physical shocks that it suffers, and technology
- Dependence on age is usually captured by the bathtub curve:

![Bathtub Curve Image]

- Young component – high failure rate
  - Good chance that some defective units slipped through manufacturing quality control and were released
- Later - bad units weeded out – remaining units have a fairly constant failure rate
- As component becomes very old, aging effects cause the failure rate to rise again

![Bathtub Curve Image]
CSCE 5760: Design For Fault Tolerance

Empirical Formula for $\lambda$ - Failure Rate

- $\lambda = \pi L \cdot \pi Q \cdot (C1 \cdot \pi T \cdot \pi V + C2 \cdot \pi E)$
  - $\pi L$: Learning factor, (how mature the technology is)
  - $\pi Q$: Manufacturing process Quality factor (0.25 to 20.00)
  - $\pi T$: Temperature factor, (from 0.1 to 1000), proportional to $\exp(-Ea/kT)$ where $Ea$ is the activation energy in electron-volts associated with the technology, $k$ is the Boltzmann constant and $T$ is the temperature in Kelvin
  - $\pi V$: Voltage stress factor for CMOS devices (from 1 to 10 depending on the supply voltage and the temperature); does not apply to other technologies (set to 1)
  - $\pi E$: Environment shock factor: from about 0.4 (air-conditioned environment), to 13.0 (harsh environment - e.g., space, cars)
  - $C1$, $C2$: Complexity factors; functions of number of gates on the chip and number of pins in the package
  - Further details: MIL-HDBK-217E handbook

How to measure fault tolerance?

- $F(t)$ - probability that the component will fail at or before time $t$
  $$F(t) = \text{Prob} \{T \leq t\}$$
- $f(t)$ - not a probability, but the momentary rate of probability of failure at time $t$
  $$f(t)dt = \text{Prob} \{t \leq T \leq t+dt\}$$
- Like any density function (defined for $t \geq 0$)
  $$f(t) > 0 \text{ (for all } t \geq 0) \text{ and } \int_{0}^{\infty} f(t)dt = 1$$
- The functions $F$ and $f$ are related through
  $$f(t) = dF(t) / dt \quad F(t) = \int_{0}^{t} f(s)ds$$
CSCE 5760: Design For Fault Tolerance

Reliability and Failure (Hazard) Rate

- The reliability of a single module - $R(t)$
  
  $R(t) = \text{Prob} \{T>t\} = 1 - F(t)$

- The conditional probability that the module will fail at time $t$, given it has not failed before, is
  
  $\text{Prob} \{t \leq T \leq t+dt \mid T \geq t\} = \frac{\text{Prob} \{t \leq T \leq t+dt\}}{\text{Prob} \{T \geq t\}} = \frac{f(t)dt}{1 - F(t)}$

- The failure rate (or hazard rate) of a component at time $t$, $\lambda(t)$, is defined as
  
  $\lambda(t) = \frac{f(t)}{1 - F(t)}$

- Since $\frac{dR(t)}{dt} = -f(t)$, we get $\lambda(t) = -\frac{1}{R(t)} \cdot \frac{dR(t)}{dt}$

Constant Failure rate and Exponential Distribution

- If the module has a failure rate which is constant over time -
  
  $\lambda(t) = \lambda$

  $\frac{dR(t)}{dt} = -\lambda R(t) ; R(0)=1$

- The solution of this differential equation is
  
  $R(t) = e^{-\lambda t}$

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

  $F(t) = 1 - e^{-\lambda t}$

- A module has a constant failure rate if and only if $T$, the lifetime of the module, has an exponential distribution
CSCE 5760: Design For Fault Tolerance

Mean Time to Failure (MTTF)

MTTF - expected value of the lifetime $T$

Two ways of calculating MTTF

First way: $MTTF = E[T] = \int_0^\infty t \cdot f(t)dt$

Second way: $dR(t)/dt = -f(t)$

$MTTF = -\int t \cdot dR(t)/dt \cdot dt = -tR(t)|_0^\infty + \int_0^\infty R(t)dt = \int_0^\infty R(t)dt$

$tR(t) = 0$ if $t = 0$, and $t = \infty$

If the failure rate is a constant $\lambda$

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

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

Notice how MTTF is related to failure rate

Weibull Distribution

Introduction

- Most calculations of reliability assume that a module has a constant failure rate $\lambda$ (or equivalently - an exponential distribution for the module lifetime $T$)
- There are cases in which this simplifying assumption is inappropriate
- Example - during the “infant mortality” and “wear-out” phases of the bathtub curve
- Weibull distribution for the lifetime $T$ can be used instead
CSCE 5760: Design For Fault Tolerance

Weibull distribution - Equation

The Weibull distribution has two parameters, $\lambda$ and $\beta$

The density function of the component lifetime $T$:

$$f(t) = \lambda t^{\beta-1} e^{-\lambda t^{\beta}}$$

The failure rate for the Weibull distribution is

$$\lambda(t) = \lambda t^{\beta-1}$$

$\lambda(t)$ is decreasing with time for $\beta < 1$, increasing with time for $\beta > 1$, constant for $\beta = 1$, appropriate for infant mortality, wearout and middle phases, respectively.

Reliability for Weibull distribution is

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

MTTF for Weibull distribution is

$$MTTF = \Gamma(1/\beta) / (\beta \lambda^{1/\beta})$$

($\Gamma(x)$ is the Gamma function)

The special case $\beta = 1$ is the exponential distribution with a constant failure rate $\lambda$. 