CSCE 5760: Design For Fault Tolerance

Prerequisites:

- Good programming background (to understand software errors)
- Good understanding of hardware
  - Digital logic – to understand logic errors
  - Computer Organization/Architecture – to understand errors in different parts of a computer
- Strong probability (and statistics) background
  - Probability distributions
  - Joint distributions

Desirable:

- VLSI – to understand how to design error tolerance
- Markov processes
- Multi-threading
- Software testing

Homework Assignment, Term Projects and Exams

I will assign problems from textbook
We will also have reading assignments – from journals and conferences
And students take turns to present these papers

**Term Project**: A technical report on a topic related to the class
- Run reliability models using software tools
- Can be to evaluate a certain technique
- Summarize the current research based on reading several papers

Exams will be open book - open notes format.
- Questions ask you to think how you will apply what you learned in class; nothing to memorize
CSCE 5760: Design For Fault Tolerance

- Textbook

  Textbook is for you to read, not for me to read it to you

- Further Readings

- Papers from journals and conference proceedings

What does fault tolerance mean?

Why do we need fault free operation?

Boeing 737 Max: What happened?

"Imminent Stall" detection initiates Maneuvering Characteristics Augmentation System (MACS)

This relies on speed and angle of attack.
If too slow and nose is up, stalls -- MACS forces nose down.

But if the sensors fail, MACS is making incorrect decision and prevents pilot from controlling the nose (or angle of attack)
Even when pilots tried to override MACS, they were unable to
CSCE 5760: Design For Fault Tolerance

Why do we call computer errors BUGS?

A Trip Back in History

- September 9, 1947
- Mark II computer at Harvard
- Popularly attributed to Grace Hopper
- Actually Bill Burke found the moth
- Diary with the taped moth can be seen at Smithsonian National Museum of American History
- A failure was attributed to a moth stuck to an electronic relay

CSCE 5760: Design For Fault Tolerance

Costly Software BUGS in History

Facebook IPO: May 2012 ➔ what happened

Computer systems used to establish the opening price were overwhelmed by order cancellations and updates during the “biggest IPO cross in the history of mankind,” Nasdaq Chief Executive Officer Robert Greifeld said Sunday. Nasdaq’s systems fell into a “loop” that prevented it from opening the shares on schedule.

Consequences:

- Investors were unsure how much of Facebook they’d bought. There were 12 million postponed share orders suddenly filled between 1:49 p.m. and 1:51 p.m. without being properly marked ‘late sale’, which exaggerated the impression that people were trying to dump Facebook shares.
- NASDAQ earmarked $62 million for reimbursing investors and even more for system upgrades and legal battles. WSJ pegged the total loss to investors at $500M
CSCE 5760: Design For Fault Tolerance

Costly Software BUGS in History


50 million people lost power for up to two days in the biggest blackout in North American history. The event contributed to at least 11 deaths and cost an estimated $6 billion.

Diagnosis

• The task force responsible for investigating the cause of the blackout concluded that a software failure at FirstEnergy Corp. "may have contributed significantly" to the outage.

• FirstEnergy’s Alarm and Event Processing Routine (AEPR), a key software program that gives operators visual and audible indications of events occurring on their portion of the grid, began to malfunction. As a result, "key personnel may not have been aware of the need to take preventive measures at critical times".

• Internet links to Supervisory Control and Data Acquisition (SCADA) software weren’t properly secure and some operators lacked a system to view the status of electric systems outside their immediate control.

Medical Devices

• It has been shown to be possible for a heart defibrillator and pacemaker to reprogram itself to shut down and to deliver jolts of electricity that could be fatal. (For a device called Maximo from industry #1 company called Medtronic).

• Also possible to glean personal patient data by eavesdropping on signals from the tiny wireless radio embedded in the implant as a way to let doctors monitor and adjust it without surgery.

• 1983-1997: There were 2,792 quality problems that resulted in recalls of medical devices, 383 of which were related to computer software (14%), according to a 2001 study analyzing FDA reports of the medical devices that were voluntarily recalled by manufacturers.

• 2014: FDA had 43 recalls so far, categorized as ones where there is "reasonable probability that use of these products will cause serious adverse health consequences or death." At least 6 of the recalls were likely caused by software defects.
Effect of major network outages on large business customers

October 2000 data

- Survey by Meta Group Inc. of 21 industrial sectors in 2000 found the mean loss of revenue due to computer system downtime was $1.01M/hour

Supercomputer applications → execution times exceed hours if not days

If one or more nodes in a supercomputer fails
   → should you restart the computation?
   → may never complete the computation

Does redundancy work?
   should you do every computation in redundant fashion?

What other choices?

Checkpoint successfully completed computations and results periodically
CSCE 5760: Design For Fault Tolerance

Good Sources For discussion on well known failures

1. The Risks Digest. At: http://catless.ncl.ac.uk/Risks

-Weekly to every two weeks; news of risks to the public in computers and related systems. Started by (and moderated by) Peter Neumann.

Homework 1: Describe (1 page) some software/hardware/network failure reported in 2019. Include a description of the event, cause and consequence.

2. D. Siewiorek and Swarz, Chapter 2 “Faults and their Manifestations” Pages 22-49


CSCE 5760: Design For Fault Tolerance

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

Unreachable

A computing system once deployed cannot easily be accessed, replaced or repaired

Sensor devices
CSCE 5760: Design For Fault Tolerance

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

DEPENDABILITY

ATTRIBUTES

RELIABILITY

SAFETY

CONFIDENTIALITY

INTEGRITY

MAINTAINABILITY

FAULT PREVENTION

FAULT TOLERANCE

FAULT REMOVAL

FAULT FORECASTING

FAILURES

ERRORS

IMPAIRMENTS

Should add security to this list
Fault Classes

- Permanent faults, whose presence is continuous and stable.
- Intermittent faults, whose presence is only occasional due to unstable hardware or varying hardware and software states (e.g., as a function of load or activity).
- Transient faults, resulting from temporary environmental conditions.

Fault Cycle & Dependability Measures

- **Reliability:** a measure of the continuous delivery of service, \( R(0) \) is the probability that the system survives (does not fail) throughout \([0, t]\); expected value: \( \text{MTTF} \) (Mean Time To Failure).
- **Maintainability:** a measure of the service interruption, \( M(0) \) is the probability that the system will be repaired within a time less than \( t \); expected value: \( \text{MTTR} \) (Mean Time To Repair).
- **Availability:** a measure of the service delivery with respect to the alternation of the delivery and interruptions. \( A(0) \) is the probability that the system delivers a proper (conforming to specification) service at a given time \( t \); expected value: \( E[A] = ?? \).
- **Safety:** a measure of the time to catastrophic failure, \( S(0) \) is the probability that no catastrophic failures occur during \([0, 1]\); expected value: \( \text{MTTCF} \) (Mean Time To Catastrophic Failure).

Why things fail?

- Physical Fault: Due to physical phenomena (usually hardware)
- Human Fault: Design errors, operational modes Did not anticipate inputs in real world

What is the difference between hardware and software faults
Does software fail? How does it fail?

Hardware Faults
- Design errors
- wear and tear
- heat or thermal (operating range)
- other environmental (radio activity, magnetic fields)
CSCE 5760: Design For Fault Tolerance

Software faults
- Design and logic errors
- Range of input values (use short integer but input was larger)
  can lead to infinite loop (as with Facebook IPO)
  Any other?

Why can’t we find faults during testing
and use only fault free systems?

Complexity of testing

Consider a circuit that takes two 32 bit integers and adds them
How do we test the circuit?

Software testing?
Formal (model) verification?

So, in practice - we can never guarantee the flawless execution of tasks under all circumstances

We need to be able to tolerate faults and design systems to overcome them
How do we do this?
  Ability to detect faults
  Techniques to overcome or mask faults

Graceful degradation
  OK to sacrifice performance for higher reliability
  OK to sacrifice power consumption for higher reliability

We will later define a term called: Performability

Probability that the system achieves some defined level of performance even in the presence of failures.
CSCE 5760: Design For Fault Tolerance

Hardware Faults Classification

- 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

Which category would most software faults belong?
Examples of permanent hardware faults → stuck at faults, short circuits, wear out
Can we think of similar software faults?
Which are more common?

Impact of hardware technology

With maturity, hardware reliability improves
Intermittent and transient are more common, but may be changing
Depends on technology → NVM have write endurance issues
Likewise small feature sizes (e.g., 7nm) are less reliable

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
CSCE 5760: Design For Fault Tolerance

Faults Vs. Errors

- **Fault** - either a hardware defect or a software/programming mistake
- **Error or Failure** - a manifestation of a fault
- **Example** An adder circuit with one output lines stuck at 1
  - This is a fault, but not (yet) an error
  - Becomes an error when the adder is used and the result on that line should be 0

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

CSCE 5760: Design For Fault Tolerance

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
- 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 → too complex)
- Assert pre and post condition
- Except handling

---
CSCE 5760: Design For Fault Tolerance

**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

CSCE5760 August 27, 2019

---

CSCE 5760: Design For Fault Tolerance

**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