RELIABILITY

Klaus-Peter Zauner

COMP2215: Computer Systems II
Context
Reliability is one of the oldest problems studied in computer science:

- Early computers were notoriously unreliable
  - Relays contacts corroded
  - Vacuum tubes burned out
- Program length was limited by mean-time to failure
A Classic Problem

Reliability is one of the oldest problems studied in computer science:

• Early computers were notoriously unreliable
  • Relays contacts corroded
  • Vacuum tubes burned out
• Program length was limited by mean-time to failure

It is still a problem: new technologies trade size and energy consumption against reliability.

nano-transistors, organic transistors, etc.
An Interesting Problem

• We would not exist if evolution would not have found a solution
  • Biological systems operate fine with plenty of noise on all scales
• All living systems need to preserve their complex organisation against entropy from the environment
  ⇒ All living systems are very good information processors
An Interesting Problem

• We would not exist if evolution would not have found a solution
  • Biological systems operate fine with plenty of noise on all scales

• All living systems need to preserve their complex organisation against entropy from the environment

⇒ All living systems are very good information processors
What can persist?

- Simple system
- Deteriorates slowly
What can persist?

- Engineered Complex System
- Deteriorates fast
What can persist?

- Self-repair
- Deteriorates slowly
Reliability in Embedded Systems
Recall

- Software has bugs
- Anything can be wrong anyway
  - arbitrary bit flips in any register or memory cell are possible
- Redundancy is required
  - several copies of the same system
    → protects against random hardware faults
  - diversity in replicated systems
    → protects against systematic faults (software bugs)
- Redundant units need to communicate
  - communication can fail
Lightning may pose a danger to patients receiving deep brain stimulation: case report

Neža Prezelj, MD¹, Maja Trošt, MD, PhD¹,², Dejan Georgiev, MD, PhD¹, and Dušan Flisar, MI

MAUDE - Manufacturer and User Facility Device Experience:
https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/index.cfm?smc=1
What can be done?

• Error Detection
• Self-repair?
Integrity Checks

Is anything wrong?

• Parity
  • Guarantee that the number of 0-bits is even (or odd)
  • Requires one extra bit for the bit-stream that is covered
  • Typically on byte level
  • Typically available in hardware

• Check sums
  • Reduce probability of undetected error
Cyclic Redundancy Checksums

- Message integrity
- Memory integrity
- Many techniques
  - e.g. take bitwise XOR of all bytes in a message
  - some techniques have been exhaustively studies (possible for microcontroller size data)
  - make sure you pick a known good CRC
  - make sure it is long enough!
  - Hardware or lookup tables
Which bit is wrong?

If we know which bit is wrong we can flip it back.

- Error Correcting Codes
- With extra redundancy some level of errors can be fully corrected
- Compress and add ECC?
  - transition from smooth degradation to sudden dropoff
  - Makes problem recognition easier
Redundant codes

Hamming distance between code words: how many bits differ?

- 1 bit: any bit flip gives a new code word
  $\Rightarrow$ no detection
- 2 bits: a single bit flip results in an invalid code word
  $\Rightarrow$ error detection
- 3 bits: a single bit flip results in an invalid code word and we know which correct code word is closer
  $\Rightarrow$ can correct single bit errors