Fault Tolerance
Some background

Claudio Pinello
(pinello@eecs.berkeley.edu)

Some Terminology

• A fault is the cause of an error;
• an error is the part of the system state which may cause a failure;
• a failure is the deviation of the system from the specification

Example

- Office Desk
  - lamp bulb fails (fault)
  - light level drops (error)
  - I can’t get work done (failure)
  - unless...

One Good Idea:

Redundancy
One Bad Idea: Redundancy

Example (raid?)

Hello World
Failure rate = FR

Hello World
Failure rate = 2*FR – FR^2

Hello World
Failure rate = FR^2

Hello World
Failure rate = ?
More on redundancy

• Space-redundancy
  – hw (e.g. 4 brakes, RAID disks, batteries, …)
  – data structures (e.g. RAID, backups)
  – software (e.g. primary/secondary DNS)
• Time-redundancy
  – compute twice (e.g. tax returns)
  – “reload” in web-browsers

Structure

• System-level fault tolerance
  – avoid single point of failure (e.g. single power supply) Redundancy
  – avoid common-mode failure (e.g. same bug in replicated software, all power supplies fail above 50°C, etc.) Diversity
  – fault isolation (e.g. provide ways to isolate fault subsystems) Modularity
  – cross your fingers!
Statistics?

- Fault rates of components
- MTBF: mean time between failures
- MTBR: mean time before repair
- Availability = MTBF / MTBR
- Some systems are more sensitive to MTBF (e.g. cars, airplanes, etc.)
- Others to availability (e.g. internet services, banking, etc.)

Fault Model

- Temporal Behavior:
  - transient faults (sporadic, periodic, etc.)
  - permanent faults
  - timing faults (loss of synch, delays, etc.)
- Statistical Behavior:
  - non deterministic (noise, etc.)
  - deterministic (bugs, out-of-range behaviors, security breach)
Fault Model

• Value Behavior:
  – Non-silent Faults: result in value errors
  – Silent Faults: result in omission errors

• Examples:
  – Crash Faults (fail-stop): result in crashes i.e. no more data, ever!
  – Byzantine Faults: malicious attacks, non-silent faults, bounded delays, etc…

Fault Tolerance: basic strategy

• Fault detection
• fault isolation
• recovery
Fault Detection

• Typically check for errors
  – Silent Faults: no errors?
    • “omission” errors! Easy for synchronous systems, otherwise use timeouts.
    • Question: You are sick in bed. How do you know if your door bell is broken?

• Typically check for errors
  – Non-silent faults: how do you know if result is wrong?
  – e.g. your calculator computes sin(), how do you know if it is faulty?
  – may use invariants (trigonometry rules), may check against Taylor expansion, against known results (sin(0) =0)
  – In general, may use invariants, properties and models (e.g. sensor data may be matched against predicted plant trajectories, clocks should monotonically increase, etc.)
  – BTW: what time is it?
Fault Detection

• Non-silent faults: try voting
  – you can tolerate up to $n/2 - 1$ faults

Fault Detection

• Typically check for errors
  – Byzantine faults: oh my!
    • you can’t trust people on chatlines…
    • can you ask them the time?
    • the account number of the red cross for a donation?
    • would you ask them what medicine to take?
Byzantine Generals

- question: “attack or retreat?”
- message passing (oral/written)
- there are traitors
- goal: determine consensus among non-traitors

Basic algorithm (by Lamport et al.)
- $n$ rounds of oral message passing
- use majority voting, decide
- Tolerates up to $< 1/3$ traitors

All methods require bounded asynchrony, i.e. bounded delays

Byzantine Generals
(an intuitive interpretation)

General 1: A
1. (AARR)
2. (((AARR), (AARA), (AARA), (ARRR)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (ARRR)),
3. (((AARR), (AARA), (AARA), (ARRR)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (ARRR))

⇒ Attack!

General 2: A
1. (AARA)
2. (((AARR), (AARA), (AARA), (ARRR)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (ARRR)),
3. (((AARR), (AARA), (AARA), (ARRR)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (ARRR))

⇒ Attack!

General 3: R
1. (AARA)
2. (((AARR), (AARA), (AARA), (ARRR)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (ARRR)),
3. (((AARR), (AARA), (AARA), (ARRR)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (ARRR))

⇒ Attack!

General 4: T
1. (AART)
2. (((AARR), (AARA), (AARA), (TTTT)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (TTTT)),
3. (((AARR), (AARA), (AARA), (TTTT)),
   ((AARR), (AARA), (AAA)),
   (AARR), (AARR), (TTTT))

⇒ Curse!

If you can use signed messages, can tolerate n-2 traitors, with fewer messages

What model to use?

- Depends on your application
  - internet transactions?
    • probably Byzantine, to tolerate impostors
  - embedded systems?
    • usually non-silent faults are sufficient, but...
    • more networked applications and more safety critical applications
  - channel transmission?
    • using CRC one “approximates” fail silence
    • cryptographic signatures
Recovery

- You detected a fault, now what?
- Isolate fault to avoid further errors
- Recover from fault
  - backtrack to known good checkpoint
  - start another agent to compute result
  - use another already available result
  - reduce functionality (e.g. slow down)
  - bring system to safe state (e.g. turn off engine)

Conclusions

- Faults do occur, do you care?
- Model them
- Use redundancy right!
- System-level fault tolerance
- Techniques exist, some are complex to get right
Hammurabi code (~1750 BC)

• 229 If a builder build a house for some one, and does not construct it properly, and the house which he built fall in and kill its owner, then that builder shall be put to death.
• 230. If it kill the son of the owner the son of that builder shall be put to death.
• 231. If it kill a slave of the owner, then he shall pay slave for slave to the owner of the house.
• 232. If it ruin goods, he shall make compensation for all that has been ruined, and inasmuch as he did not construct properly this house which he built and it fell, he shall re-erect the house from his own means.

Inspiration: Prof. Patterson; reproduced without permission from http://eawc.evansville.edu/anthology/hammurabi.htm