Fault Tolerance & Safety

SS04 ATSEfSCS Seminar
Andrew Hines

Why use Fault Tolerance?

- Faults in HW & SW cannot be avoided
- What to do when a fault occurs?
  - Let it go unnoticed & let it sort itself out?
  - Effect an automatic safety shutdown mechanism?
  - Immediately employ entire backup systems?
- What if faults can be somehow anticipated?
- Fault tolerance!
  - Airbus and Boeing aeroplanes
  - COMTRAC System in Japanese Railways
  - NASA space shuttle

Presentation Overview

1. Faults, errors & failures
2. Characteristics of Fault Tolerance (FT)
3. Fault Tolerance techniques
4. Conclusion

Faults, Errors & Failures

- Fault: defect within system [STO96]
  - Fault in component has potential to infect all other components that depend on it
- Error: deviation of required operation of system or subsystem [STO96]
- System failure: occurs when system fails to perform its required function [STO96]
**Fault/Failure Chain**

- Fault $\Rightarrow$ Error $\Rightarrow$ Failure
- Fault $\Rightarrow$ Error $\Rightarrow$ Failure

- Fault $\Rightarrow$ Error
  - dormant vs. active
- Error $\Rightarrow$ Failure
  - latent vs. detected
- Failure $\Rightarrow$ Fault
  - Failure occurs when error passed through and affects the required service

---

**Fault/Failure State Transitions**

- Correct state
- Error state
- Failure/accident

- Service restoration
- Fault activation
- Error processing
- Error passes

---

**Fault Classification**

(1) **Nature**
- Random faults
  - Primary cause is HW component failure
- Systematic faults
  - Caused by fault in design (HW or SW)
  - HW faults due to random failures or design mistakes
  - SW faults always due to design mistakes

(2) **Extent**
- Localised fault
- Global Fault

---

**Fault Classification**

(3) **Duration**
- Permanent faults
  - Design faults (HW & SW) always permanent
  - Most random HW faults permanent
- Transient faults
  - Caused by temporary system malfunction or external interference
- Intermittent faults
  - e.g. poor solder joints
  - Transient & Intermittent faults much harder to detect
  - HW faults may be all 3 types, but design faults always permanent
Fault Tolerance

- Faults are inevitable
- Fault tolerance (FT): Providing a service that is consistent with its specification in spite of faults [GIE03]
- System tolerant against failures of its components (not its own failures)
- Most important attribute of dependability for FT is reliability (lesser extent availability)
- All methods of FT based on form of redundancy
- Redundancy: parts of system not needed for correct functioning of system, if no FT supported [JAL94]

Fault Tolerance & Redundancy

1. Domain of information
   - Redundant information, e.g. error correcting codes, robust data structures

2. Domain of space
   - Replication of components
     - Active vs. Passive redundancy

3. Domain of time
   - Replication of computations
   - Sending messages multiple times

Redundancy & Diversity

- Redundancy with identical components protects against random HW component failures, but not systematic ones
- Diversity is also needed

4 Phases of Fault Tolerance

1. Error detection
2. Damage confinement & assessment
3. State restoration
   - backward/forward recovery
4. Fault treatment & continued service
   - utilising built-in redundancy
Techniques for Fault Tolerance

(1) Systematic fault-tolerance
- Replication of components
- Divergence of components used for fault detection
- Redundant components are used for continued service

(2) Application-specific fault-tolerance
- Reasonableness checks for fault detection (based on model of real world)
- State estimations for continued service

Application-specific Fault Tolerance
- Computer system interacts with physical process
- Laws of physics used by the system to check its state for reasonableness &/or perform state estimations
- Both reasonableness checks and state estimations are based on application knowledge
- Fail-stop behaviour from reasonableness checks
- Fail-operational behaviour from reasonableness checks and state estimations

Systematic Fault Tolerance
- Uses replicated components instead of application knowledge
- Replicas are required to deliver corresponding results in absence of faults
  - The problem of replica determinism
- Fail-stop behaviour from information of divergent results
- Fail-operational behaviour by using redundant components

Recovery Blocks

Discard checkpoint
Evaluate acceptance test
Execute alternate
Establish checkpoint

Restore checkpoint

Exception signals

New alternate exists & deadline not expired?

Failure exception

pass

fail

yes

no

entry

exit
**Recovery Blocks**

- Apply diverse designs (systematic FT) to provide design FT based on acceptance test (application-specific FT)
- Difficult to develop acceptance tests & quality often questionable
- Can also handle transient HW faults
- Can implement graceful degradation
  - Different modules ⇒ different levels of service
- Recovery blocks can be nested
- Can be supported with exception mechanism

---

**N-Version Programming**

- N non-identical replicated SW modules are applied and a voter takes a majority decision
- approach to systematic FT
- Driver program to invoke different modules
- Redundancy
  - Domain of space or time
  - Signal comparison (voting)
- Protection against (N-1)/2 faulty program versions
- High implementation & performance costs
- Common mode faults are not excluded

---

**Exception Handling**

- SW & HW mechanisms to detect exceptional states
- Then decide how to continue
  - Designers may specify recovery (forward or backward)
- Run-time system used to handle faults
- Subdivision of state domain for procedure
  - anticipated vs. unanticipated vs. standard

---

**N-Version Programming**

- Execution environment (EE)
- Consensus results
- Execution support functions
- Decision Algorithm (Voter)
- Version 1
- Version 2
- Version 3
### N Self-checking programming

- $n$ versions executed in parallel
- Self-checking modules
- Acceptance tests used
- Mixture of application specific and systematic FT
- Requires no backward recovery and no voting

---

### Other FT Techniques

#### Data Diversity

- System input
- Execute algorithm
- Re-express data
- Valid Output? No
  - Yes, System output
  - No, Error
- Deadline Expired? No
  - Yes, Deadline

#### Deadline Mechanism

- Service name
- Within response period by primary module else by alternate module
- Slack time
- Max time for alternate

---

### Conclusion

- Faults are inevitable!
- For ultimate safety, system should handle all anticipated faults
- All FT techniques have strengths & weaknesses
  - Guides their use within FT systems
- In practice, a variety of HW and SW fault tolerant techniques used in combination
- Fault tolerance depends largely on redundancy
- Goal: provide continuous, reliable service in presence of faults consistent with its spec

---

### Bibliography

Any Questions?