Fault-Tolerant Architectures

- Applications
  - General Purpose Computing
  - High-Availability Systems
    - rapid error detection and correction => minimize downtime
    - unacceptable downtime for software installation/updates
    - examples AT&T switching systems, Tandem: software-intensive approach, Stratus: hardware approach.
  - Long-Life Systems
    - mobile systems: airplanes, mass transit systems etc.
    - the concept of deferred maintenance
    - special considerations: highly redundant spacecraft systems
      - automatic reconfiguration vs. remote access
    - down-times might not be of great concern

- Critical Computations
  - real-time control systems and their timing sensitivity
  - heavy computational workloads => multiple processors
  - hard real-time environment
    - tasks have hard/soft deadlines
    - failure to meet deadlines => catastrophic results
  - need for provably correct algorithms
    - formal verification methods
    - no unexpected side effects
  - classic systems
Fault-Tolerant Architectures

- Brief discussion of some systems
  - AT&T (highly available switching systems)
    - goal: 2 hours downtime in 40 years (3 min/year :-)
    - Pra96 table 2.7, pg 104: Probability of operational outage due to various sources.
    - User implements part of redundancy, i.e. redial
    - Pra96 table 2.8, pg 105: Levels of recovery in a switching system.
    - system features include
      - hardware lock-step duplication
      - online processors write to both stores
      - byte parity on data paths
      - modified hamming code on main memory
      - maintenance channel for observability/controllability of processors
      - extensive self checking hardware (30% +)

Table 2.7: Probability of Operational Outage Due To Various Sources$^a$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>0.20</td>
<td>0.26$^d$</td>
<td>*%</td>
<td>0.18</td>
<td>0.19</td>
<td>.19</td>
<td>.45</td>
</tr>
<tr>
<td>Software</td>
<td>0.15</td>
<td>0.30$^e$</td>
<td>0.75$^f$</td>
<td>0.26</td>
<td>0.43</td>
<td>.19</td>
<td>.20</td>
</tr>
<tr>
<td>Maintenance</td>
<td>---</td>
<td>---</td>
<td>*%</td>
<td>0.25</td>
<td>0.13</td>
<td>---</td>
<td>.05</td>
</tr>
<tr>
<td>Operations</td>
<td>0.65$^g$</td>
<td>0.44$^h$</td>
<td>0.11</td>
<td>0.17</td>
<td>0.13</td>
<td>.33</td>
<td>.15</td>
</tr>
<tr>
<td>Environment</td>
<td>---</td>
<td>---</td>
<td>0.13</td>
<td>0.14</td>
<td>0.12</td>
<td>.28$^i$</td>
<td>.15</td>
</tr>
</tbody>
</table>

$^a$Dashes indicate that no separate value was reported for that category in the cited study.

$^b$Fraction of downtime attributed to each source. Downtime is defined as any service disruption that exceeds 30 seconds duration. The Bellcore data represented a 3.5-minute downtime per year per system.

$^c$Split between procedural errors (0.3) and recovery deficiencies (0.35).

$^d$42% of the hardware failures occurred due to the second unit failing before the first unit could be replaced.

$^e$Recovery software.

$^f$Split between procedural errors (0.42) and operational software (0.02).

$^g$Study only reported probability of vendor-related outage (i.e., 0.75 is split between vendor hardware, software, and maintenance).

$^h,(15)$attributed to power.
Fault-Tolerant Architectures

- **Tandem**
  - High-availability systems for transaction processing.
  - NonStop1 -- first commercial OS designed for high availability.
  - Design objectives
    - nonstop operation: non-intrusive fault detection, reconfiguration and repair.
    - data integrity: no single hardware failure can compromise data integrity.
    - modular system expansion: software application not affected by adding expansion hardware.
  - No single point of failure: dual paths to all system components, including disks, I/O controllers, processor replication, power supplies, RAID 1 disks, and message based OS.
Fault-Tolerant Architectures

> Pra96 fig 2.4, pg 112
  - loosely shared-memory architecture
  - duplication of all components
> Hardware/Software modules designed to behave like a FSP
> Retries on I/O devices
  1) hardware retry, assuming transient fault
  2) software retry
  3) alternate path retry
  4) alternate device retry
> Check point recovery mechanism
> Maintenance and diagnosis system analyzes the event log and automatically calls for field replaceable units.

Figure 2.4: Tandem’s system organization
Fault-Tolerant Architectures

- Stratus
  - Continuous checking of duplexed components
  - Pair and Spare Architecture Pra96, fig 2.7, pg 117
    - 2 processor boards with 2 microprocessors each
    - each board operates independently
    - bus halves are wired-ORed with their counterparts
  - One module consists of replicated power, backplane buses
  - Modules can be interconnected => communicate via message passing SIB (Stratus Intermodule Bus).
  - Boards compare their halves and remove themselves upon disagreement between A and B halves, indicating maintenance interrupt => FSP behavior
  - Board is diagnosed for transient fault and possibly returned to service. Permanent failure is reported by phone to customer assistance center.

Figure 2.7: The Stratus pair-and-spare architecture.
Fault-Tolerant Architectures

- Spacecraft Systems: Long period of unattended operation
  - Design considerations include effects of environment, power, temperature, stability, vibration etc.
  - Systems range from weather, and communication satellites in varies orbits to deep-space probes.
  - Propulsion: controlling fuels and stabilization.
  - Power: regulating and storing power from different sources, e.g. solar panels, batteries.
    - Table 2.13, pg 125, Typical Power Subsystem
  - Data Communication: communication with earth using uplinks, data stream from craft using redundant downlinks
  - Attitude Control: redundant sensors, gyros, momentum wheels
  - Command and Control: hardware testing of parity, illegal instructions, mem. addresses, sanity checks, timing mechanisms

<table>
<thead>
<tr>
<th>Element</th>
<th>Tracking Solar Array</th>
<th>Solar Array Drive</th>
<th>Slip-Ring Assembly</th>
<th>Charge Controller</th>
<th>Batteries</th>
<th>Power Regulation</th>
<th>Power Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redundancy</td>
<td>Extra capacity series/parallel connections of individual solar cells allows for graceful degradation</td>
<td>Redundant drive elements and motors</td>
<td>Parallel rings for power transfer</td>
<td>Automatic monitoring and control of battery charge state</td>
<td>Series/parallel connections; diode protection</td>
<td>Redundant spares</td>
<td>Automatic load shedding</td>
</tr>
</tbody>
</table>

Table 2.14: Attributes of the Voyager Spacecraft

<table>
<thead>
<tr>
<th>Systems Characteristics</th>
<th>Propulsion</th>
<th>Power</th>
<th>Data Communications</th>
<th>Attitude Control</th>
<th>Command and Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planetary probe</td>
<td>Hydrazine thrusters</td>
<td>Three radioactive thermal generation; 430 W at Jupiter</td>
<td>Downlink, 2: uplink, 1; two antennas (high gain and low gain)</td>
<td>Redundant sun sensors and Canopus (star) trackers</td>
<td>Command rate 16 bpi</td>
</tr>
<tr>
<td>Three-axis stabilized</td>
<td>Mission life: 7 years</td>
<td></td>
<td></td>
<td></td>
<td>Redundant computer, 4K words each; data storage on board</td>
</tr>
</tbody>
</table>

© 2007 A.W. Krings
Fault-Tolerant Architectures

- SIFT (software implemented fault tolerance) (70s)
  - intended for real-time aircraft control
  - assumption that future airplanes would be designed to be unstable
  - loss of computer for even milliseconds could lead to catastrophe
  - how does one verify systems when fail rates are $10^{-10}$?
  - approach: mathematically prove correctness of system software
  - hardware is assumed to use independent computers using fully connected graph topology, implementing unidirectional series links.
  - software divided into tasks, results from redundant tasks are voted upon.
    (Actually it is the inputs to tasks that is voted on).
  - 3 processor example Pra96, fig 2.11, pg 130
    - input to A is output of voter with 3 inputs

Figure 2.11: Arrangement of application tasks within SIFT configuration. (Adapted from Wersley 1978.)