Safety-Critical Real-Time Systems
ARTES PhD course
Lecture 6: Architectural principles

Simin Nadjm-Tehrani ©
Real-time Systems Laboratory
simin@ida.liu.se

Article study
• Identified keywords, concepts?
• First, some basic notions and results...

Fail safe – fail operational
• From NASA Shuttle web:

  Fail-operational performance means that, after one failure in a system, redundancy allows the vehicle to continue on its mission.

  Fail-safe means that after a second failure, the vehicle is still capable of safely returning to a landing site.

Fault containment regions
• (FCR) A collection of components that operate correctly regardless of any arbitrary logical or electrical fault outside the region.

  [Lala & Harper 94]

Common mode failures
• Faults that affect more than one fault containment region (typically due to a common cause)

Fail-silent
• A system component exhibits a fail-silent behaviour if upon failure it shuts down so that its failure can be reliably detected by other FCR:s
**The consensus problem**

- Processes $p_1, \ldots, p_n$ take part in a decision
- Each $p_i$ proposes a value $v_i$
- All correct processes decide on a common value $v$ that is equal to one of the proposed values

**Desired properties**

- Every correct process eventually decides a value (Termination)
- No two correct processes decide differently (Agreement)
- If a process decides $v$ then the value $v$ was proposed by some process (Validity)

**Basic impossibility result**

[Fischer, Lynch and Paterson 1985]

There is no deterministic algorithm solving the consensus problem in an asynchronous distributed system with a single crash failure

**A way around it**

Assume Synchrony:

- Distributed computations proceed in rounds initiated by pulses
- Pulses implemented using local physical clocks, synchronised assuming bounded message delays

**Byzantine agreement**

- A difficult problem solved in 1980 by Pease, Shostak and Lamport
- Each process may fail in an arbitrary way
- Theorem: There is an upperbound $f$ for the number of byzantine failures compared to the size of the network, $N \geq 3f+1$, using a $f+1$ round algorithm

**This lecture**

- What are the distinguishing characteristics of a safety-critical system’s architecture?
  - Architectural patterns for safety
- What are the benefits of identifying such patterns as opposed to building in safety application by application?
### Architecture in a nutshell

- If relevant patterns are found, the analysis of the architecture is done once and for all and reused in many applications.
- Two schools:
  - Middleware oriented
  - Application oriented

### Two approaches

- Middleware oriented:
  - Generic architectural properties can be enforced by elements of middleware
  - Applications use middleware through standard interfaces
- Application (sector) oriented:
  - Practices, methods and analysis tools

### Example: Bus architectures

- Automotive:
  - TTA: extensive middleware support
  - Flexray: Minimal support (only timing isolation)
- Aerospace:
  - Honeywell Safebus: extensive support
  - Gripen DFCS study: timing support and application specific algorithms

### Application oriented

- With some support from middleware:
  - Synchronous clocks
- Two approaches:
  - Bit-wise exact consensus
  - Approximate consensus (Inherent replica consistency)

### The price to pay

- Bitwise exact:
  - Replica determinism
  - Extra hardware costs (3f+1)
- Approximate:
  - Domain knowledge dependent

### Architectural support

- Timed-triggerd architecture
  - [Kopetz et.al]
  - Time division multiple access (TDMA)
### Communication protocol

- **Message Description List:** allocates a pre-defined slot within which each node can send its (pre-defined) message

```
Node 1    ...    Node n-1    Node n
```

A TDMA round

### Temporal firewall

- Provides temporally accurate state information
- When the data is no longer valid, it can no longer be exchanged
- Separates broadcast properties from application dependent replicated code

### Replication & failure detection

- **BG:** Buss Guardian

### What is the catch?

- Middleware as a component, needs to be verified by:
  - Formal analysis
  - Fault injection
- Detecting and masking arbitrary (asymmetric) failures can be costly in time and hardware