**EDA122/DIT061 Fault-Tolerant Computer Systems**

Welcome to Lecture 1

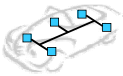
Johan Karlsson

## Why fault tolerance?

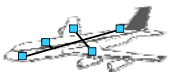
**We *depend* on computer systems!**

Bank services

Data centers



Active safety systems



Fly-by-wire

Work stations



## Fault tolerance – What?

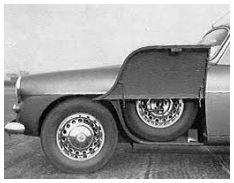
***Fault tolerance*** *means to* ***avoid service failures*** *in the* ***presence of faults****.*

Avizienis, et al., "Basic Concepts and Taxonomy of Dependable and Secure Computing"



## Fault-Tolerance – How?

- By introducing ***redundancy*** (extra resources)
- Forms of redundancy
  - hardware redundancy
  - software redundancy
  - time redundancy
  - information redundancy





## Objectives of fault tolerance

**(Dependability Attributes)**

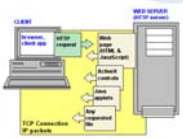
- **Reliability** – readiness for correct service
- **Availability** – continuity of correct service
- **Safety** – absence of catastrophic consequences on user(s) and the environment
- **Integrity** – absence of improper system alterations



## Applications Areas for Fault Tolerance (1)

**Business-critical applications**

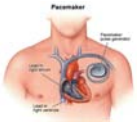
- Web servers
- Cloud computing
- Financial transaction system
- E-business
- General-purpose file servers
- …

## Applications Areas for Fault Tolerance (2)
### Embedded systems


Active safety systems for road vehicles


Fly-by-wire systems


Medical devices


Factory automation


Railway signaling



## Applications Areas for Fault Tolerance (3)
### Cyber-physical systems

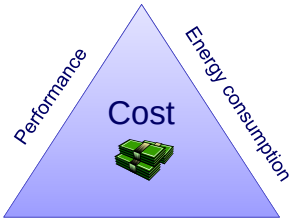
- Smart grids (smart electrical power grids)
- Cooperative active safety systems for road vehicles
- Remote surgery
- …


Smart Grid


Surgical robot system

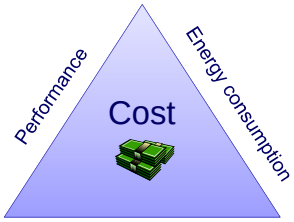


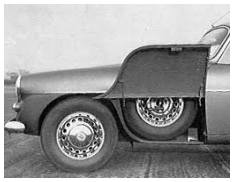
## Trade-offs in Computer System Design





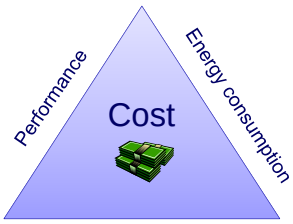

## Trade-offs in Computer System Design





No performance degradation after fail-over. (No difference between the spare and the original tire.)



## Trade-offs in Computer System Design





Graceful degradation after fail-over: Reduced speed and stability.



## Redundancy in HP's NonStop System

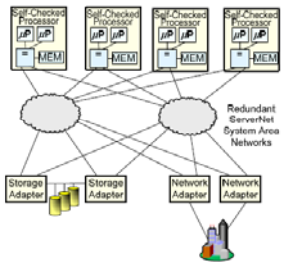
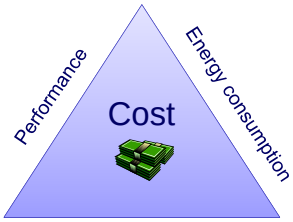



Figure 1: 4 Processor NonStop System with Duplicated and Compared Microprocessors

## Trade-offs in Computer System Design



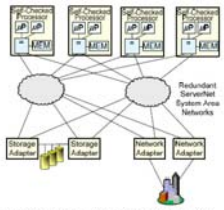




Figure 1: 4 Processor NonStop System with Duplicated and Compared Microprocessors



## Course Outline

- 17 lectures (17 x 2 h) including 3 guest lectures
- 10 exercise classes (10 x 2 h)
- 2 laboratory classes (2 x 4 h)
- 7,5 credits (hp)



## Course Homepage

*www.cse.chalmers.se/edu/course/EDA122*

*Also available via the student portal*

Here you find:
- The course PM (contains all administrative information)
- Lecture slides
- Messages from the examiner
- Old exams, etc



## Teachers

Johan Karlsson, ext. 1670, room 4107
johan@chalmers.se (examiner and lecturer)

Negin Fathollah Nejad, ext. 5404, room 4127
negin@chalmers.se (teaching assistant)

Fatemeh Ayat, ext. 1084, room 4467
fataya@chalmers.se (lab assistant)

Behrooz Sangchoolie, ext. 1084, room 4467
behrooz.sangchoolie@chalmers.se (lab assistant)



## Examination

- Written examination
- Exam dates: 23 October, 2012, afternoon
  15 January, 2013, afternoon
  ?? August, 2013, afternoon
- Grades: Failed, 3, 4, 5 (Chalmers),
  Failed, G, VG (GU)
- Participation in laboratory classes + approved laboratory report for laboratory class 1



## Literature

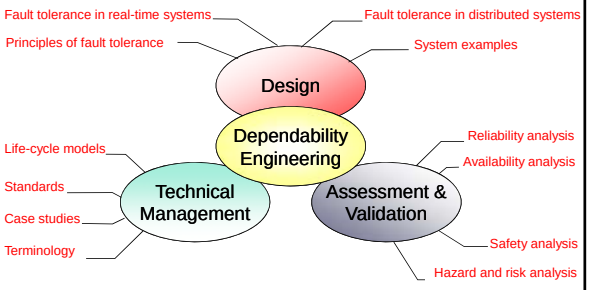
- Course book: Neil Storey, "Safety-Critical Computer Systems", Prentice Hall, ISBN 0-201-42787-7
- Reprints of articles on selected topics in fault-tolerant computing (available on the course homepage)
- Lecture slides
- Compendium of exercise problems
- PMs for laboratory classes (Lab PM)

## Course Evaluation

- ***Two to six student representatives***, representing different programmes
- Student representatives will receive a voucher valid for 200 SEK at Cremona
- Three meetings:
  Two during the course and one after the course
- Student representatives are expected to
  1. ***Provide feedback from all students***
  2. ***Review and help design the course questionnaire***
  3. ***Participate in all meetings***



## Overview of topics






## Learning goals

After completion of the course the student should be able to:

- Formulate *dependability requirements* for computer systems used in business-, safety- and mission-critical applications.
- Describe the *structure and principles* of commonly used system architectures of fault tolerant computers.
- Perform probabilistic dependability analysis of computer system using *fault-trees*, *reliability block diagrams*, *time-continuous Markov chains* and *stochastic Petri nets*.
- Master the terminology of dependable computing.
- Describe basic concepts in *life-cycle models* and *standards* employed in the development of safety-critical systems.



## Outline for the rest of this lecture

- Faults, Errors and Failures
- Safety
- Basic terminology
- Voting redundancy



## Dependability Threats and Failure Pathology

**Fault** - Cause of an error, e.g., an *open circuit*, a *software bug*, or an *external disturbance*.

↓

**Error** - Part of the system state which is liable to lead to failure, e.g., a wrong value in a program variable.

↓

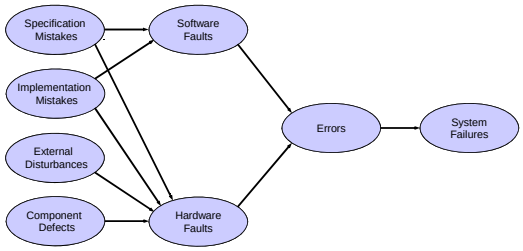
**Failure** - Delivered service does not comply with the specification, e.g., a cruise control in a car locks at full speed.



## Fault Types

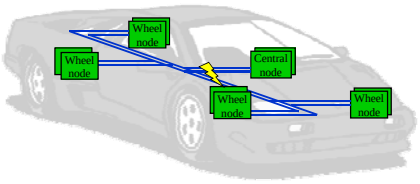
- Random faults (physical faults)
  - Aging faults
    - E.g., degradation of transistors in integrated circuits
  - External disturbances
    - Ionizing particle radiation
    - Electromagnetic interference
- Systematic faults (development faults in HW or SW)
  - Specification faults
  - Design faults
  - Implementation faults

## Cause-and-Effect Relationship



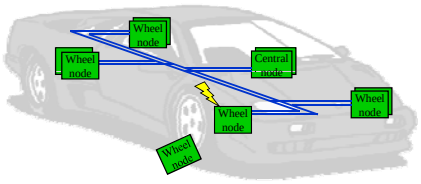



## Brake-By-Wire System



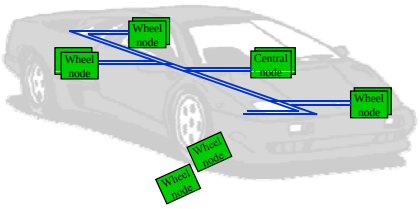



## Brake-By-Wire System






## Brake-By-Wire System






## Safety

*__Safety__ is a property of a system that it will not endanger human life or the environment*

*A __safety-related__ system is one by which the safety of equipment or plant is assured*

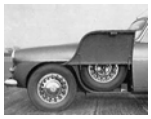
*The term __safety-critical system__ is normally used as a synonym for a safety-related system, although it may suggest a system of high criticality*

(Neil Storey)



## Important concepts

- Fault tolerance
  - To avoid service failures in the presence of faults
- Graceful degradation
  - Gradual reduction of service in the presence of faults



## Fault tolerance vs. Fault prevention

- **Fault tolerance** – to avoid service failure during operation
  - Requires mechanisms for error handling during operation, e.g.,
    - Error detection
    - System recovery
    - Fail-over (e.g., moving an executing program from a faulty computer to a non-faulty computer)
- **Fault prevention** – to prevent or reduce the occurrence of faults
  - Applied during development, e.g.,
    - Robust design (e.g., select hardware components with low failure rates)
    - Testing (find and remove design faults in HW and SW)
    - Proving correctness by formal methods



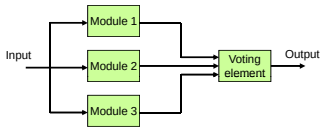
## Hardware Redundancy

- Voting redundancy (this lecture)
- Stand-by redundancy (lecture 2)
- Active redundancy (lecture 2)



## Voting redundancy

### Triple Modular Redundancy (TMR)



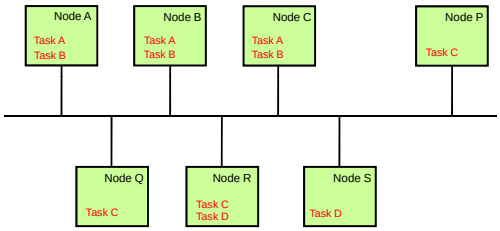

The three modules receive the same inputs and perform the same calculations.

The modules are assumed to fail independently, one at a time.

The voting element uses majority voting to mask errors.

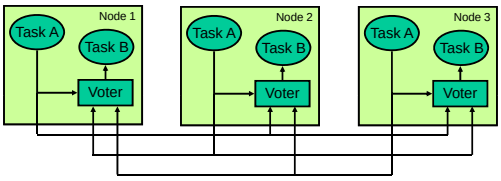


## Replicated tasks in a distributed system






## Voting in a distributed system




The tasks and voters are implemented in software.

The figure shows the exchange of data messages between the replicas of Task A and Task B in the previous slide.



## Failure = Service failure

- A failure occurs when a **service provider** (system, or subsystem) delivers an incorrect service.
- Example: A node is a subsystems in a distributed system
  - Node failure – a node delivers an incorrect service
- Example: A network is a subsystems in a distributed system
  - Network failure – a network delivers an incorrect service
- Example: A processor core is a subsystem in a multi-core processor
  - Core failure – a core delivers an incorrect service

## Fundamental Concepts
### Failure mode

A ***failure mode*** describes the nature of a failure

- Examples of failure modes:
  - Value failure – a service provider delivers an erroneous result
  - Content failure – same as value failure
  - Timing failure – a service provider delivers a result too late, or too early
  - Silent failure – a service provider delivers no result
  - Signaled failure – a service provider sends a failure signal
  - Interference failure – a service provider disturbs the service delivered by another service provider



## Failure model vs. Failure mode

- A ***failure model*** is a set of assumptions about likely failure modes for a service provider
- A ***failure mode*** describes the nature of a given class of failures



## Fundamental Concepts
### Error processing

***Error processing*** aims at removing errors from the computational state, if possible, before a failure occurs.

Error processing techniques:

- Error detection - to detect errors
- Error masking - to mask the effects of errors
- Recovery - to restore the system to an error-free state



## Recovery

We distinguish between two types of recovery:

- **Forward recovery**
  - The state of the service provider is moved *forward* in time
  - Example: Error free state is copied from another (redundant) service provider
- **Backward recovery**
  - The state of the service provider is moved *backward* in time
  - Example: Error free state is restored from a previously stored checkpoint
  - Checkpoint is stored in a crash proof memory, a.k.a. *stable storage*



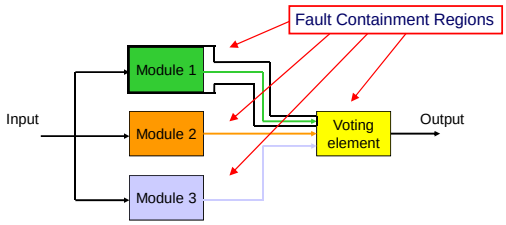
## Fundamental Concepts
### Fault/Error Containment

***Fault/Error containment*** aims at preventing faults/errors from affecting other (redundant) units in the system.

- A fault-tolerant system consist of several ***fault/error containment regions***

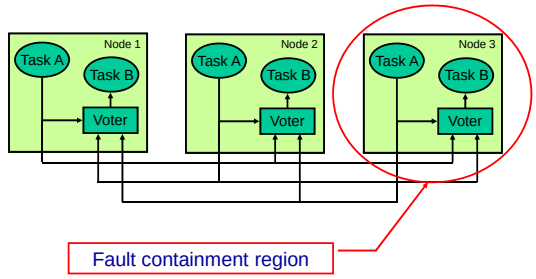


## Fault Containment Regions in a TMR System



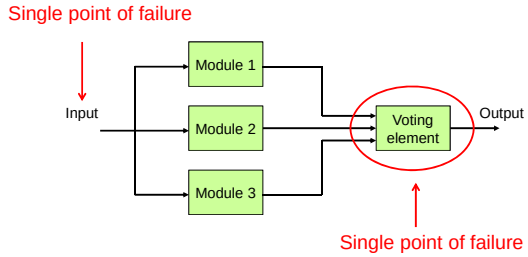

The designer must prevent that a fault in one module causes faults in the other module, or the voting element.

## Fault Containment Region in a Distributed TMR system



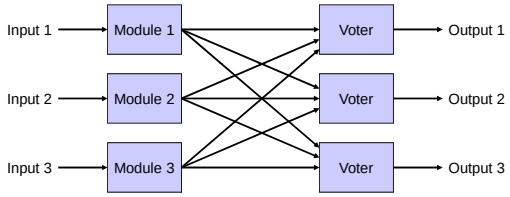



## Single Points of Failure in a TMR System



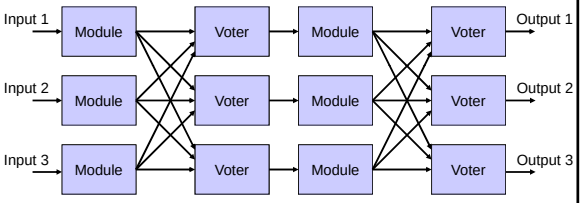



## Masking Redundancy
### TMR with triple voting and triple inputs






## Masking Redundancy
## Multi-stage TMR






## Summary

- Fault tolerance
- Graceful degradation
- Safety
- Failure pathology: faults → errors → failures
- Voting redundancy
- Fault/error containment
- Single point of failure
- Multi-stage voting



## Overview of Lecture 2

- More about hardware redundancy
- Case Study: HP's Non-Stop Advanced Architecture

Read ***before*** the lecture:
- Course book: Section 6.1, 6.3, 6.4, 6.5, 6.8
- Bernick er al., "Non-Stop Advanced Architecture"