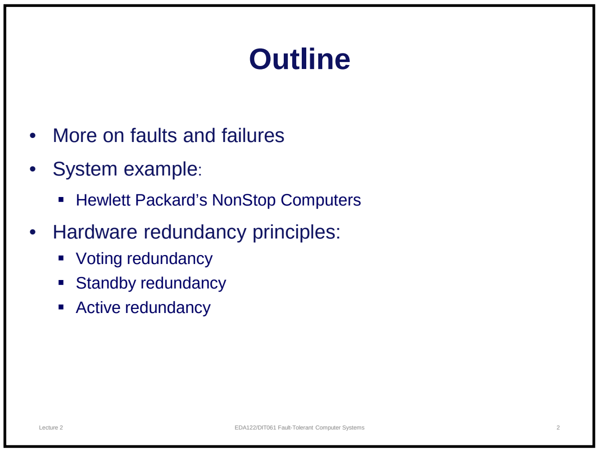
EDA122/DIT061 Fault-Tolerant Computer Systems

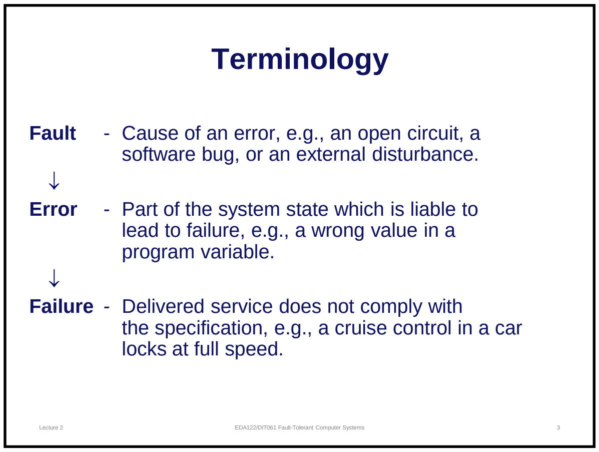
Welcome to Lecture 2

Hardware redundancy



Outline

- More on faults and failures
- System example:
  - Hewlett Packard's NonStop Computers
- Hardware redundancy principles:
  - Voting redundancy
  - Standby redundancy
  - Active redundancy



Terminology

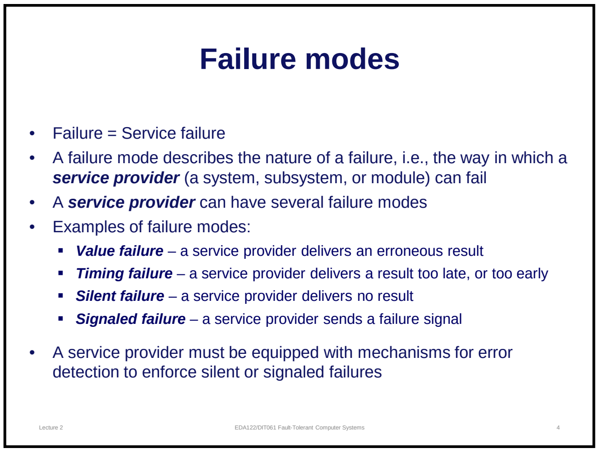
Fault - Cause of an error, e.g., an open circuit, a software bug, or an external disturbance.

↓

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

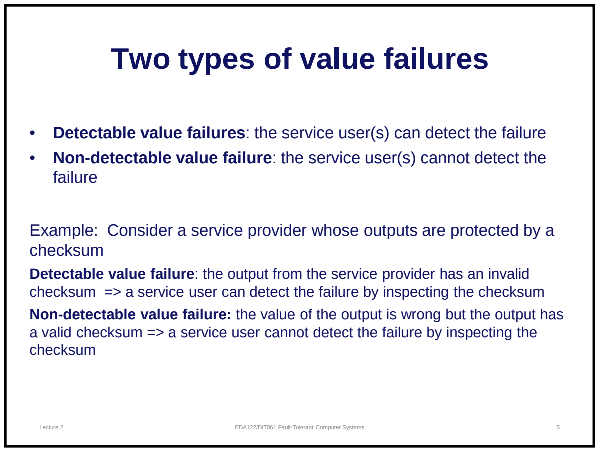
↓

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



Failure modes

- Failure = Service failure
- A failure mode describes the nature of a failure, i.e., the way in which a ***service provider*** (a system, subsystem, or module) can fail
- A ***service provider*** can have several failure modes
- Examples of failure modes:
  - ***Value failure*** – a service provider delivers an erroneous result
  - ***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
- A service provider must be equipped with mechanisms for error detection to enforce silent or signaled failures



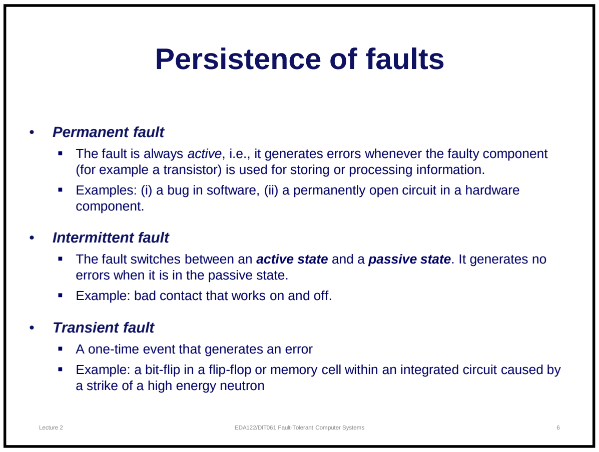
Two types of value failures

- **Detectable value failures**: the service user(s) can detect the failure
- **Non-detectable value failure**: the service user(s) cannot detect the failure

Example: Consider a service provider whose outputs are protected by a checksum

Detectable value failure: the output from the service provider has an invalid checksum => a service user can detect the failure by inspecting the checksum

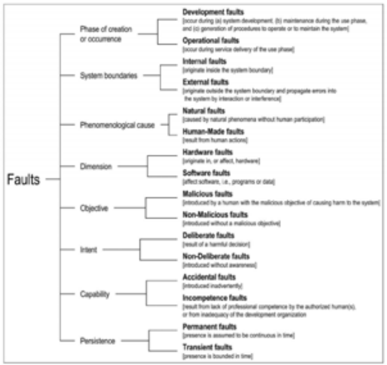
Non-detectable value failure: the value of the output is wrong but the output has a valid checksum => a service user cannot detect the failure by inspecting the checksum



Persistence of faults

- ***Permanent fault***
  - The fault is always *active*, i.e., it generates errors whenever the faulty component (for example a transistor) is used for storing or processing information.
  - Examples: (i) a bug in software, (ii) a permanently open circuit in a hardware component.
- ***Intermittent fault***
  - The fault switches between an ***active state*** and a ***passive state***. It generates no errors when it is in the passive state.
  - Example: bad contact that works on and off.
- ***Transient fault***
  - A one-time event that generates an error
  - Example: a bit-flip in a flip-flop or memory cell within an integrated circuit caused by a strike of a high energy neutron

Elementary fault classes


Faults
Phase of creation or occurrence
Development faults
[occur during (a) system development, (b) maintenance during the use phase, and (c) generation of procedures to operate or to maintain the system]
Operational faults
[occur during service delivery of the use phase]
System boundaries
Internal faults
[originate inside the system boundary]
External faults
[originate outside the system boundary and propagate errors into the system by interaction or interference]
Phenomenological cause
Natural faults
[caused by natural phenomena without human participation]
Human-Made faults
[result from human actions]
Dimension
Hardware faults
[originate in, or affect, hardware]
Software faults
[affect software, i.e., programs or data]
Objective
Malicious faults
[introduced by a human with the malicious objective of causing harm to the system]
Non-Malicious faults
[introduced without a malicious objective]
Intent
Deliberate faults
[result of a harmful decision]
Non-Deliberate faults
[introduced without awareness]
Capability
Accidental faults
[introduced inadvertently]
Incompetence faults
[result from lack of professional competence by the authorized human(s), or from inadequacy of the development organization
Persistence
Permanent faults
[presence is assumed to be continuous in time]
Transient faults
[presence is bounded in time]

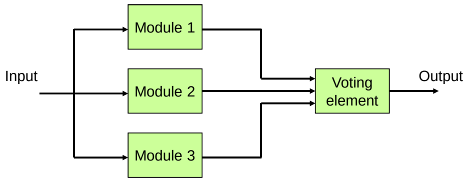
From Avizienis, et al., ”Basic Concepts and Taxonomy of Dependable and Secure Computing”

Hardware redundancy principles

- Voting redundancy
- Standby redundancy
- Active redundancy

Voting redundancy

Triple Modular Redundancy (TMR)



Voting redundancy

- Majority voting is used to mask errors in the module outputs
- **Failure mode assumption for modules (worst-case):** *non-detectable value failures* ⇒ modules are ***not*** required to have internal mechanisms for error detection and failure signaling
- Voting redundancy can also cope with *detectable value failures*, *signaled failures* and *silent failures*.
- Requires $2f+1$ units to tolerate f faulty units

Hardware redundancy principles

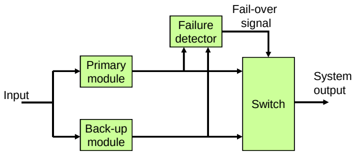
- Voting redundancy
- Standby redundancy
- Active redundancy

Standby redundancy

- One active (primary) module is backed-up by one or several spare modules
- Relies on failure detection and system reconfiguration
- Switching to a spare module (system reconfiguration) is called a *fail-over*

- **Failure mode assumptions**: *silent failures*, *signaled failures*, or *detectable value failures*
- Requires $f+1$ units to tolerate f faulty units

Hot standby redundancy



Module 1 and 2 perform the same computations.

Module 1 delivers system output when the system is started.

The failure detector checks the output of the primary module and issues a *fail-over* command when it detects a failure in the output of the primary module.

Failure mode assumptions for main modules: *silent failures*, *signaled failures*, or *detectable value failures*

HP’s NonStop Computer Systems

- Highly available computers for on-line transaction processing (OLTP) systems
- Typical applications:
  - Automatic teller machines, Stock trading, Funds transfer, 911 emergency centers, Medical records, Travel and hotel reservations, etc
- Availability: 0,99999 – “five nines”, or 5 min downtime per year
- Data integrity: 1 FIT = 10^{-9} undetected errors per hour (one undetected data error per billion hours)

Marketing information from HP (from 2005)

- Telecommunications
  - 135 public telephone companies rely on NonStop technology.
  - More than half of all 911 calls in the United States and the majority of wireless calls worldwide depend on NonStop servers.
- Finance
  - Eighty percent of all ATM transactions worldwide and 66 percent of all point-of-sale transactions worldwide are handled by NonStop servers.
  - NonStop technology powers 75 percent of the world’s 100 largest electronic funds transfer networks and 106 of the world’s 120 stock and commodity exchanges.

NonStop System with self-checked processors

Self-checked processors

- “Fail-fast” (fail-silent) processors
- Stops promptly if an error occurs
- Prevents error propagation

Error detection techniques

- Duplication and comparison
- Error correcting code
- Self-checking logic

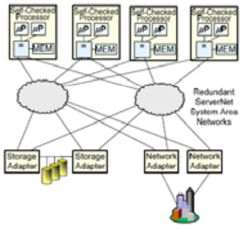


Figure 1: 4 Processor NonStop System with Duplicated and Compared Microprocessors

NonStop System with self-checked processors

Process pairs

- Critical software is implemented as a process pair, with one primary and one backup process executing on different processors
- The primary process execute the program and sends state changes regularly to the backup process
- Backup process takes over if the primary process fails by itself or as a result of a processor failure (fail-over)
- Example of standby redundancy

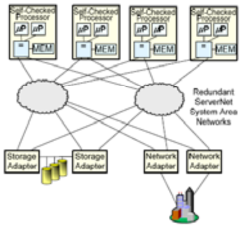


Figure 1: 4 Processor NonStop System with Duplicated and Compared Microprocessors

Evolution of self-checked processors

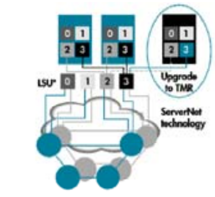
- Self-checking processors (mid 1970’s to mid 1980’s)
  - Custom designed processors
  - Self-checking logic circuits
  - Memories protected by error correcting codes
- Lock-stepped microprocessors (mid 1980’s to late1990’s)
  - Two microprocessors run synchronously using the same clock and their write operations to the main memory are compared
    - A mismatch between memory writes stops the processor
  - Main memory protected by error correcting codes
    - Non-correctable memory errors stops the processor
- Loosely synchronized processors (late 1990’s – now)
  - Comparison of I/O-operation (e.g. disk read/write)
  - Dual or triple modular redundancy

Reasons why tightly lock-stepped microprocessors have become infeasible

- Modern microprocessors are nondeterministic - asynchronous events, such as interrupts, can be handled differently by two microprocessors even if they use the same clock
- Power management techniques using variable clock frequencies cannot be used with lock-stepped microprocessors
- Increasing susceptibility to soft errors (radiation induced errors) requires low level error detection and rollback recovery routines, which complicates lock-stepped operation of microprocessors
- Multi-core processors cannot be tightly synchronized

NonStop Advanced Architecture

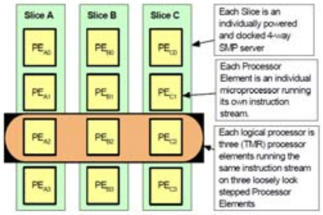
(released 2005)



* Note: LSU = logical synchronization unit

- Four Intel Itanium processors per board
- Four logical processors
- The output from the LSU represents a logical processor
- Each logical processor consists of two processors (dual modular redundancy) or three processors (triple modular redundancy)
- TMR allows hardware faults to be masked without fail-over
- Fail-over is performed if a logical processor fails

Logical Processors



Each Slice is an individually powered and clocked 4-way SMP server

Each Processor Element is an individual microprocessor running its own instruction stream.

Each logical processor is three (TMR) processor elements running the same instruction stream on three loosely lock stepped Processor Elements

Figure 2: Four Logical NSK Processors built from TMR 4-way SMP servers.

The Itanium Processor

Itanium 2 processor from Intel Corp.

For more info see: http://en.wikipedia.org/wiki/Intel_Itanium

Principles of Fault Tolerance (1)

Fault/Error Containment

- Each processor has its own memory
- Processors communicate via messages passed over the redundant System Area Network (SAN)

No single point of failure

- All system components (processors, I/O adapters, I/O devices, and the System Area Network) are replicated

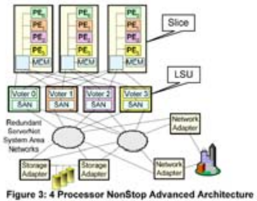


Figure 3: 4 Processor NonStop Advanced Architecture System TMR Configuration

Principles of Fault Tolerance (2)

Failure mode assumptions

- Most system components are self-checking and *fail-fast*
  - Simple errors (e.g., memory access errors or I/O timeouts) can be corrected, or masked, by retry.
  - For other errors, components are *fail-fast*: the components stops when an error is detected (a.k.a. fail silent)

Error detection

- Voting or comparison in the Logical Synchronization Unit (LSU)
- ServerNet messages protected by CRC-checksums
- Disk data protected by end-to-end checksum

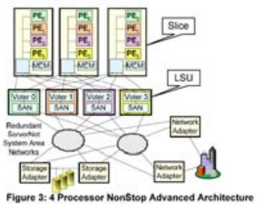


Figure 3: 4 Processor NonStop Advanced Architecture System TMR Configuration

Protection of disk data

- Disk data is stored on a mirrored pair of disk drives
- For each block of disk data, the processor calculates a checksum covering the data and the block address
- The checksum is written to disk along the with the data
- The processor verifies the checksums when it reads the data from disk
- If the checksum is incorrect, the data is read from the other unit of the mirrored pair
- Note: The inclusion of the block address in the checksum allows detection of errors where the disk returns data from another block than the one pointed out by the block address.

Principles of Fault Tolerance (3)

Dual modular redundancy (DMR)

- Two slices and four LSUs
- A logical processor consists of two processor elements (PEs) and one LSU
- The Logical Synchronization Unit compares the results from the two PEs
- The logical processor stops if the LSU detects a mismatch
- Applications must be configured as a process pair to be fault tolerant.
- A failure of the logical processor initiates fail-over to backup process

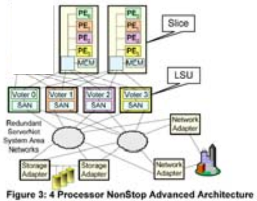


Figure 3: 4 Processor NonStop Advanced Architecture System DMR Configuration

Principles of Fault Tolerance (4)

Triple modular redundancy (TMR)

- A logical processor consists of three processor elements (PEs) and one or two LSUs
- Voting in the LSU masks processor faults.
- With one LSU, an LSU failure stops the logical processor
- With two LSUs, the system tolerates failures of any two hardware units (PEs or LSUs)
- TMR allows hardware faults to be masked without fail-over
- Reconfigures to DMR if one PE is faulty

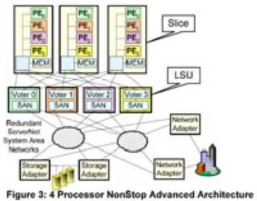


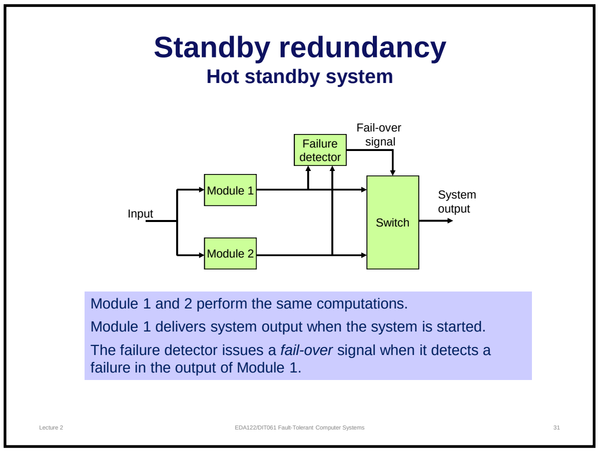
Figure 3: 4 Processor NonStop Advanced Architecture System TMR Configuration

Key Features of The NonStop Advanced Architecture

- Dual or Triple Modular Redundant Servers
- Built from standard 4-way SMP processor modules
- Each processor element (PE) have its own private memory and runs its own private copy of the operating system
- Uses a proprietary operating system called NonStop OS that supports process pairs
- Logical processors are formed by two or three processing elements located in different processor modules
- The processors that comprise a logical processor are loosely synchronized by one or two Logical Synchronization Unit (LSU), which also perform voting.
- Critical processes (tasks) can be moved from one logical processor to another logical processor (fail-over)
- Redundant ServerNet SANs (System Area Networks) connect processor modules and I/O devices
- Logical disk volumes are implemented by a pair of mirrored disks

Classification of Standby Systems

- Hot standby redundancy
- Warm standby redundancy
- Cold standby redundancy



Standby redundancy

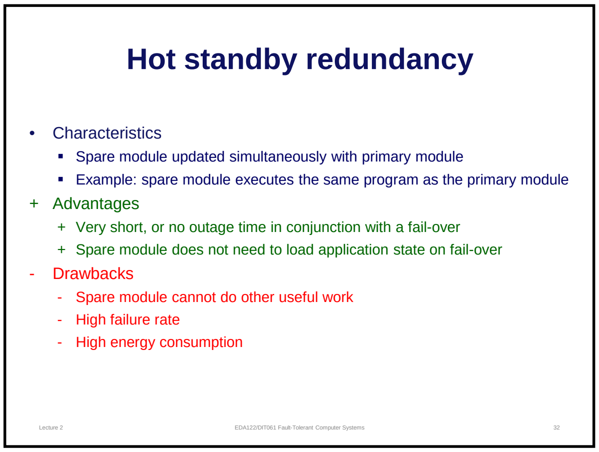
Hot standby system

Failure detector
Fail-over signal
Module 1
Input
Module 2
Switch
System output

Module 1 and 2 perform the same computations.

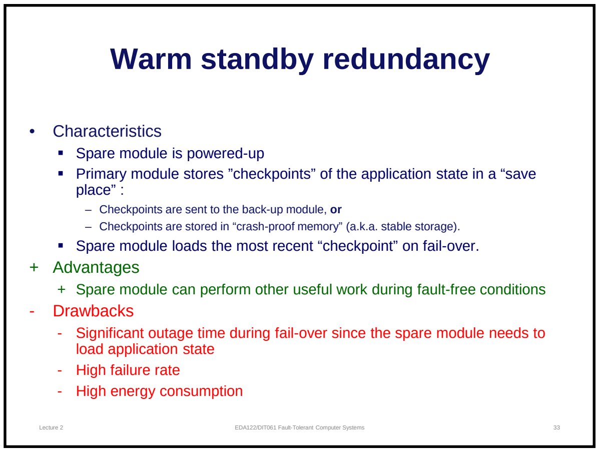
Module 1 delivers system output when the system is started.

The failure detector issues a *fail-over* signal when it detects a failure in the output of Module 1.



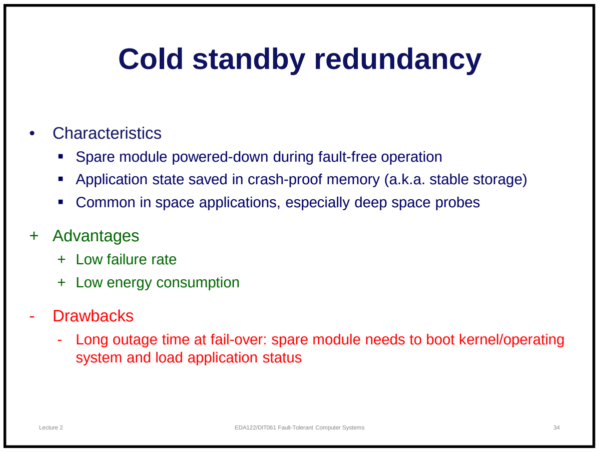
Hot standby redundancy

- Characteristics
  - Spare module updated simultaneously with primary module
  - Example: spare module executes the same program as the primary module
- + Advantages
  - + Very short, or no outage time in conjunction with a fail-over
  - + Spare module does not need to load application state on fail-over
- - Drawbacks
  - - Spare module cannot do other useful work
  - - High failure rate
  - - High energy consumption



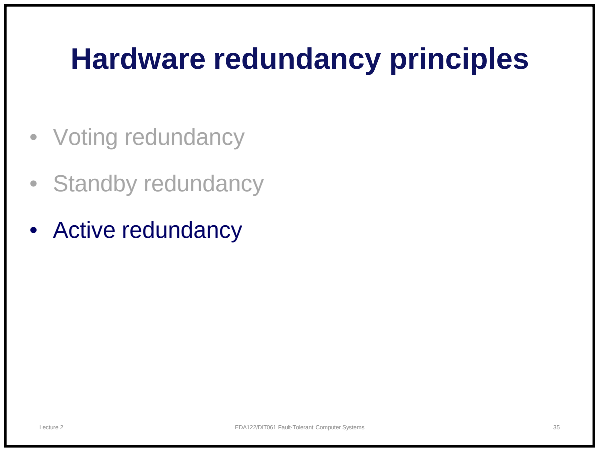
Warm standby redundancy

- Characteristics
  - Spare module is powered-up
  - Primary module stores ”checkpoints” of the application state in a “save place” :
    - Checkpoints are sent to the back-up module, **or**
    - Checkpoints are stored in “crash-proof memory” (a.k.a. stable storage).
  - Spare module loads the most recent “checkpoint” on fail-over.
- + Advantages
  - + Spare module can perform other useful work during fault-free conditions
- - Drawbacks
  - - Significant outage time during fail-over since the spare module needs to load application state
  - - High failure rate
  - - High energy consumption



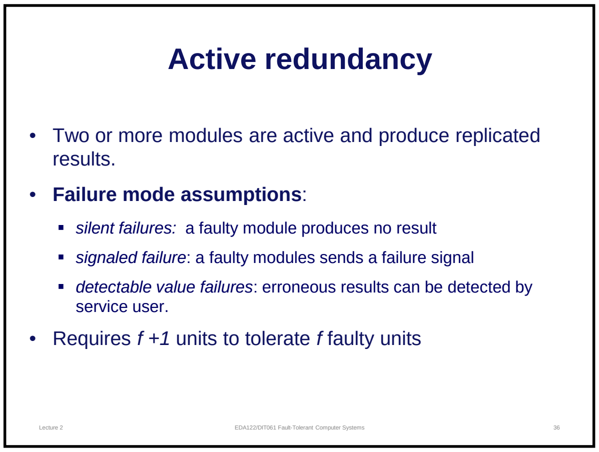
Cold standby redundancy

- Characteristics
  - Spare module powered-down during fault-free operation
  - Application state saved in crash-proof memory (a.k.a. stable storage)
  - Common in space applications, especially deep space probes
- + Advantages
  - + Low failure rate
  - + Low energy consumption
- - Drawbacks
  - - Long outage time at fail-over: spare module needs to boot kernel/operating system and load application status



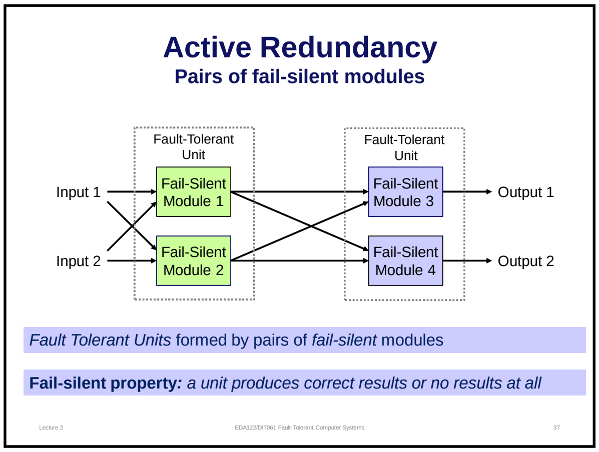
Hardware redundancy principles

- Voting redundancy
- Standby redundancy
- Active redundancy



Active redundancy

- Two or more modules are active and produce replicated results.
- **Failure mode assumptions**:
  - *silent failures:* a faulty module produces no result
  - *signaled failure*: a faulty modules sends a failure signal
  - *detectable value failures*: erroneous results can be detected by service user.
- Requires *f +1* units to tolerate *f* faulty units

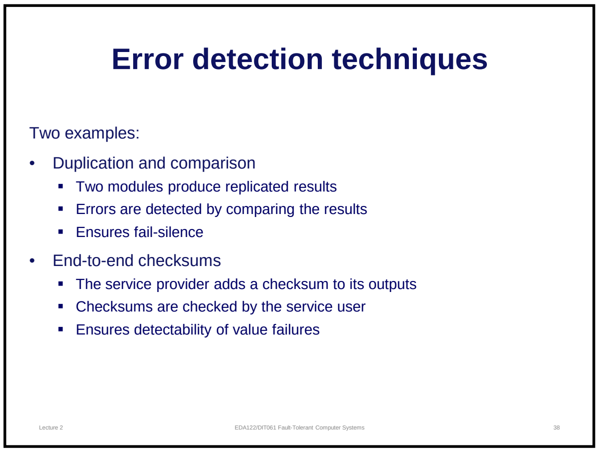


Active Redundancy

Pairs of fail-silent modules

Fault Tolerant Units formed by pairs of *fail-silent* modules

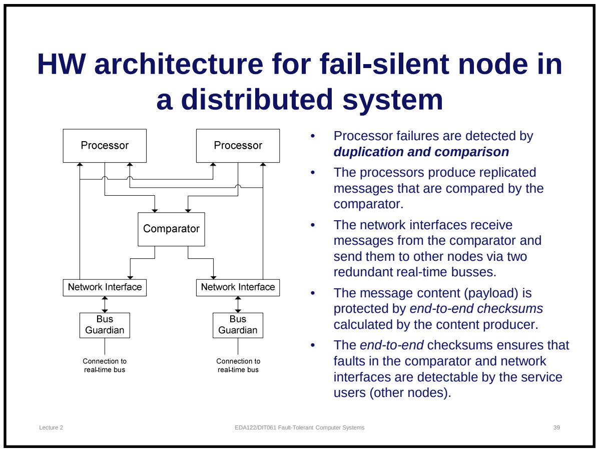
Fail-silent property*:* *a unit produces correct results or no results at all*



Error detection techniques

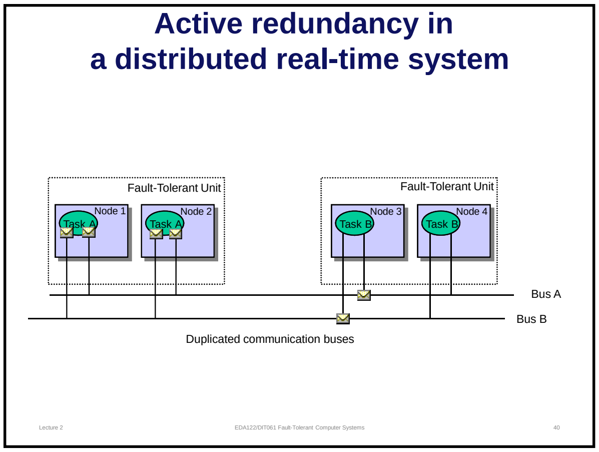
Two examples:

- Duplication and comparison
  - Two modules produce replicated results
  - Errors are detected by comparing the results
  - Ensures fail-silence
- End-to-end checksums
  - The service provider adds a checksum to its outputs
  - Checksums are checked by the service user
  - Ensures detectability of value failures

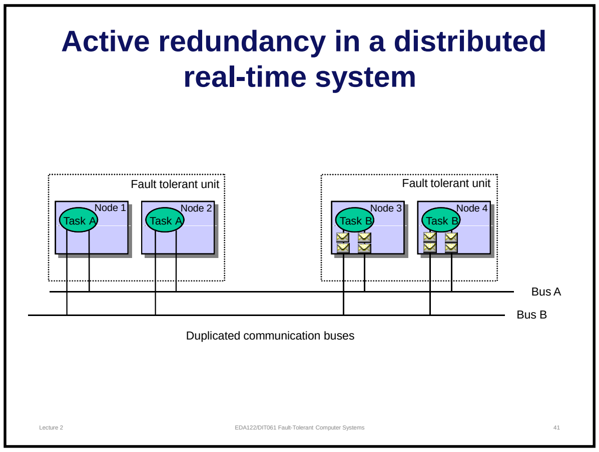


HW architecture for fail-silent node in a distributed system

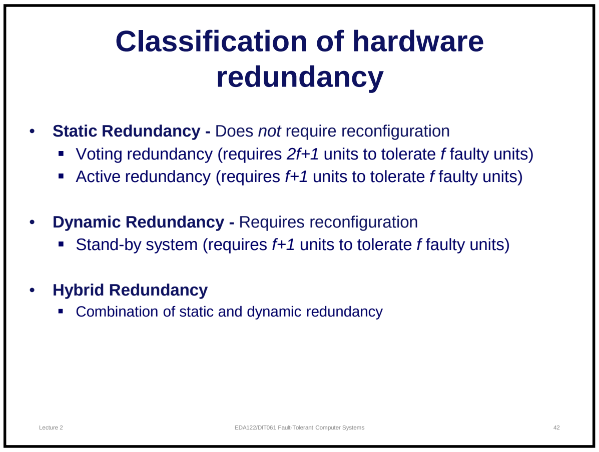
- Processor failures are detected by ***duplication and comparison***
- The processors produce replicated messages that are compared by the comparator.
- The network interfaces receive messages from the comparator and send them to other nodes via two redundant real-time busses.
- The message content (payload) is protected by *end-to-end checksums* calculated by the content producer.
- The *end-to-end* checksums ensures that faults in the comparator and network interfaces are detectable by the service users (other nodes).



Active redundancy in a distributed real-time system

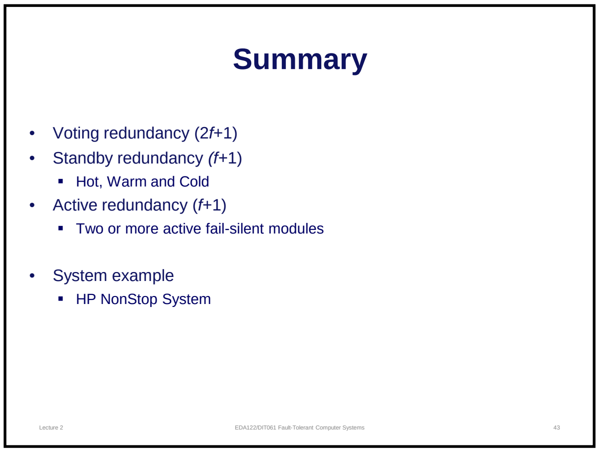


Active redundancy in a distributed real-time system



Classification of hardware redundancy

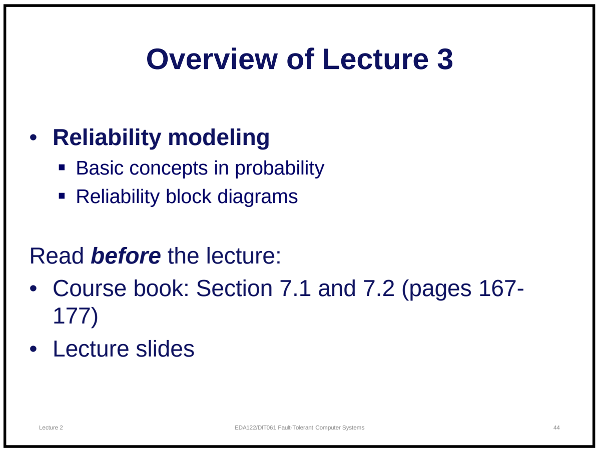
- **Static Redundancy -** Does *not* require reconfiguration
  - Voting redundancy (requires *2f+1* units to tolerate *f* faulty units)
  - Active redundancy (requires *f+1* units to tolerate *f* faulty units)
- **Dynamic Redundancy -** Requires reconfiguration
  - Stand-by system (requires *f+1* units to tolerate *f* faulty units)
- **Hybrid Redundancy**
  - Combination of static and dynamic redundancy



Summary

- Voting redundancy (2*f*+1)
- Standby redundancy (*f*+1)
  - Hot, Warm and Cold
- Active redundancy (*f*+1)
  - Two or more active fail-silent modules

- System example
  - HP NonStop System



Overview of Lecture 3

- **Reliability modeling**
  - Basic concepts in probability
  - Reliability block diagrams

Read ***before*** the lecture:

- Course book: Section 7.1 and 7.2 (pages 167-177)
- Lecture slides