Reliability of Computer Systems
Part 2: Reliable Systems / Fault-tolerant Systems

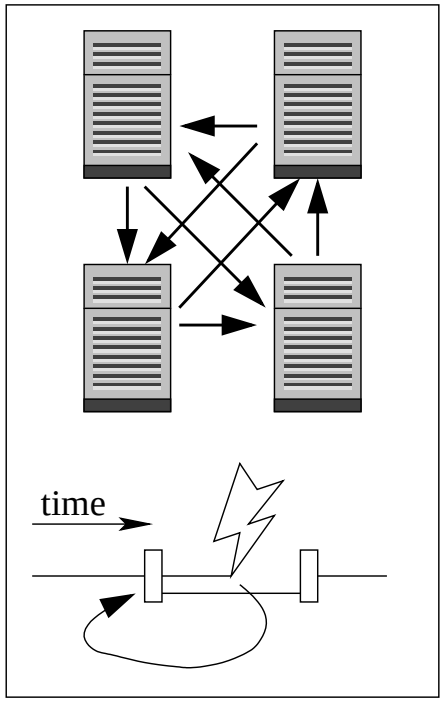
Peter Sobe

HTW Dresden, Germany
Faculty of Computer Science and Mathematics

29th July 2016

Overview

- Technical treats to improve the reliability of a system.
- Failure classes (crash vs. wrong results)
- Fault detection, failure detection and diagnosis
- Failure tolerance techniques



Dependability

Dependable system:
Something or someone depends on the adequate system functionality.

This means:

- system may not fail, i.e. must be available
- system output is required to be accurate
- system must fulfill timing requirements
- fallback to safe operation (degraded operation sometimes possible)
- system is required to be secure (in terms of security)

Dependability

Terms:

- Dependability covers Reliability, Availability, but also addresses Survivability and Safety.
- Safety is the absence of dangerous states or malfunctions that affect something's or someone's safety.
- Survivability and Maintainability (Repair, Configuration even under unforeseen conditions)
- Security and Confidentiality

Source:
A. Avizienis and J.-C. Laprie and B. Randell:
Dependability and its Threats - A Taxonomy. IFIP Congress, Topical Sessions, 2004, 91-120

Fault-tolerant System

Fault-tolerant system:
A system that fulfills predefined functionalities, in spite of failures of a limited number of components in the system

Examples:

- Storage system consisting of several disks, failures of a few disks can be tolerated
- A data base operating system that tolerates the crash of a node that processes a transaction on the database
- A computer-based fligh control system that continues its service even when a single computer crashes

Fault-tolerant System

Implementation choices:

- software fault tolerance
- hardware fault tolerance
- system level fault tolerance (without distinguishing between hardware and software faults)
- hardware-based and software-based techniques

Software-based techniques require redundancy of the hardware which is commonly present in distributed systems

Fault classes (1)

Fault classes:

- Crash: abrupt end of processing
- Fail stop, fail silent: Processing ends, no further output, no further messages to other subsystems
- Timing fault: System delivers result (output, message, signal) too early or too late
- Omission fault: A component omits an action, i.e. it does not produce a required output value. After that it works properly. Such faults may cause arbitrary faults in higher system levels.
- Arbitrary faults: Output of wrong values, choice of wrong message destinations
- Byzantine Faults: Special case of arbitrary faults, inconsistent communication in distributed systems, malicious faults (bad, evil)

Fault classes (2)

Another classification:

- Permanent faults
- Temporary faulty: Transient faults and intermitting faults

Additional notes:

- Timing faults: In an asynchronous system, there is no proper and no improper timing. It can not be decided, if a timing fault is present or a crash fault occurred.
- In practice, assumptions on a proper timing are made and timeouts are applied.
- In higher system levels, arbitrary faults occur seldom. Mostly, fault tolerance techniques are implemented for crash faults and for fail-stop faults.

Fault tolerance techniques (1)

Redundancy - is the presence of more function-ready technical components than it would be necessary for fulfilling the primary functionality of the system. Redundancy covers everything that is unnecessary in the fault free case.

Fault tolerance techniques (2)

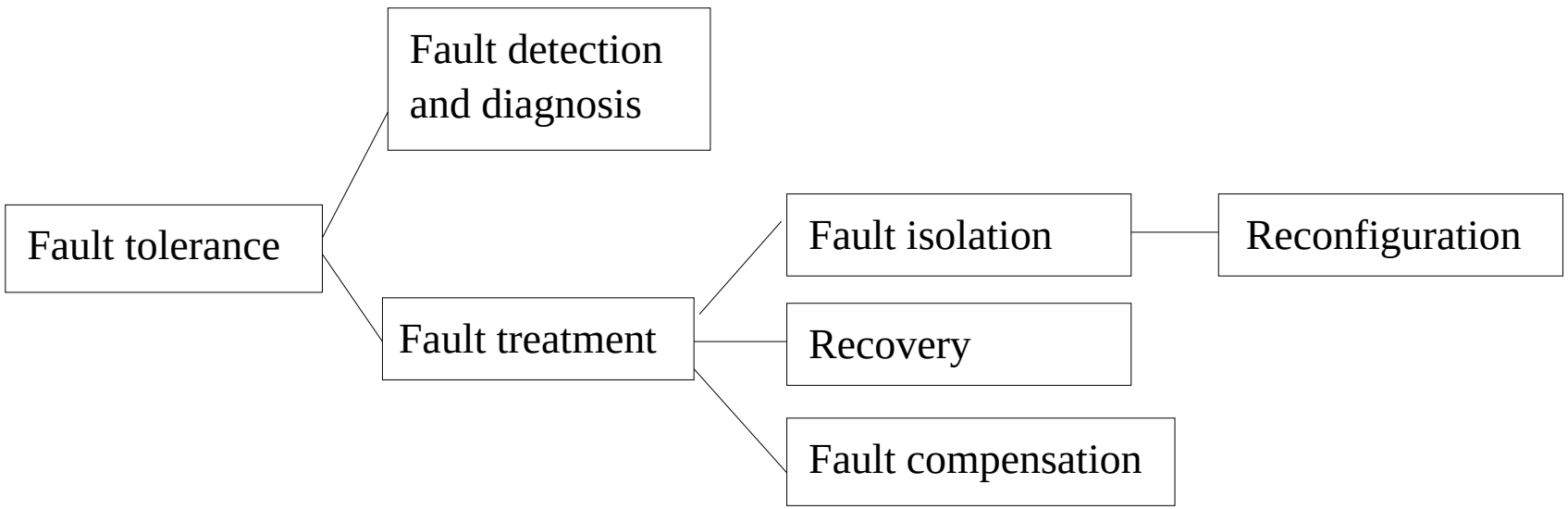
Situation of distributed system:

- Distributed systems can be seen as a serial configuration of components.
- Distributed systems contain redundancy when not all of the components involved in the primary function (additional compute nodes, communication links, memory in distant nodes).
- uncritical applications in distributed systems can be equipped with techniques that tolerate fail-stop and timing faults. These techniques require relatively little additional resources
- critical applications should apply techniques that tolerate arbitrary faults, even Byzantine faults (including attacks).

Fault tolerance techniques (3)

The actual benefit of redundancy depends on the selected (combination of) fault tolerance technique(s) and on the reliability of the single components.

Techniques:



Fault tolerance techniques (4)

Fault isolation:

- removes the influence of faults by removing the faulty components from the system, removal can be done virtually (e.g. exclusion of the component from the communication network)

Recovery:

- transfers components from a faulty state into a faultfree state, possible in case of temporary faults

Fault compensation (fault masking):

- leaves the faulty component in the system
- calculates a faultfree total result from parts of the components results.

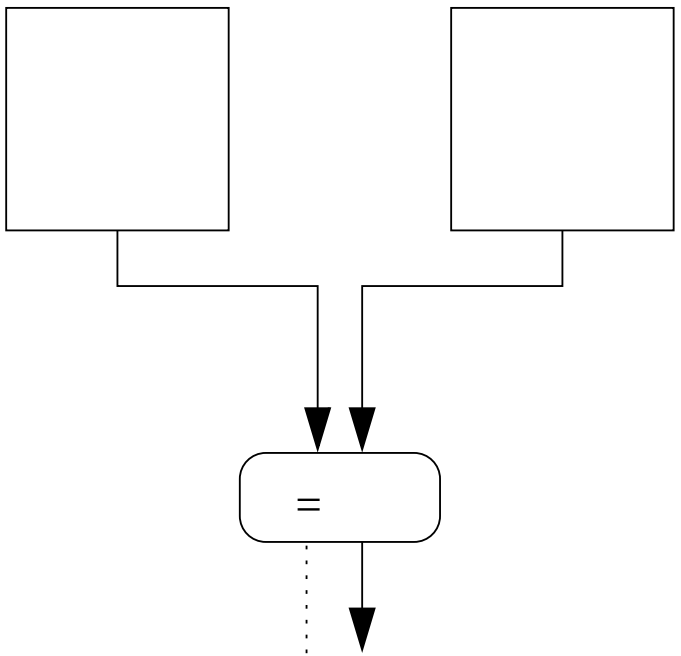
Fault tolerance techniques (5)

Reconfiguration

- is a fault isolation in connection with inclusion of new faultfree components (taken from the redundant ones).
- covers the function failures only, does not bring the components back to a fault free state (when the faulty state propagated to other components)
- typically applied in combination with recovery (e.g. rollback to an earlier faultfree state and continuation)

Fault tolerance techniques (6)

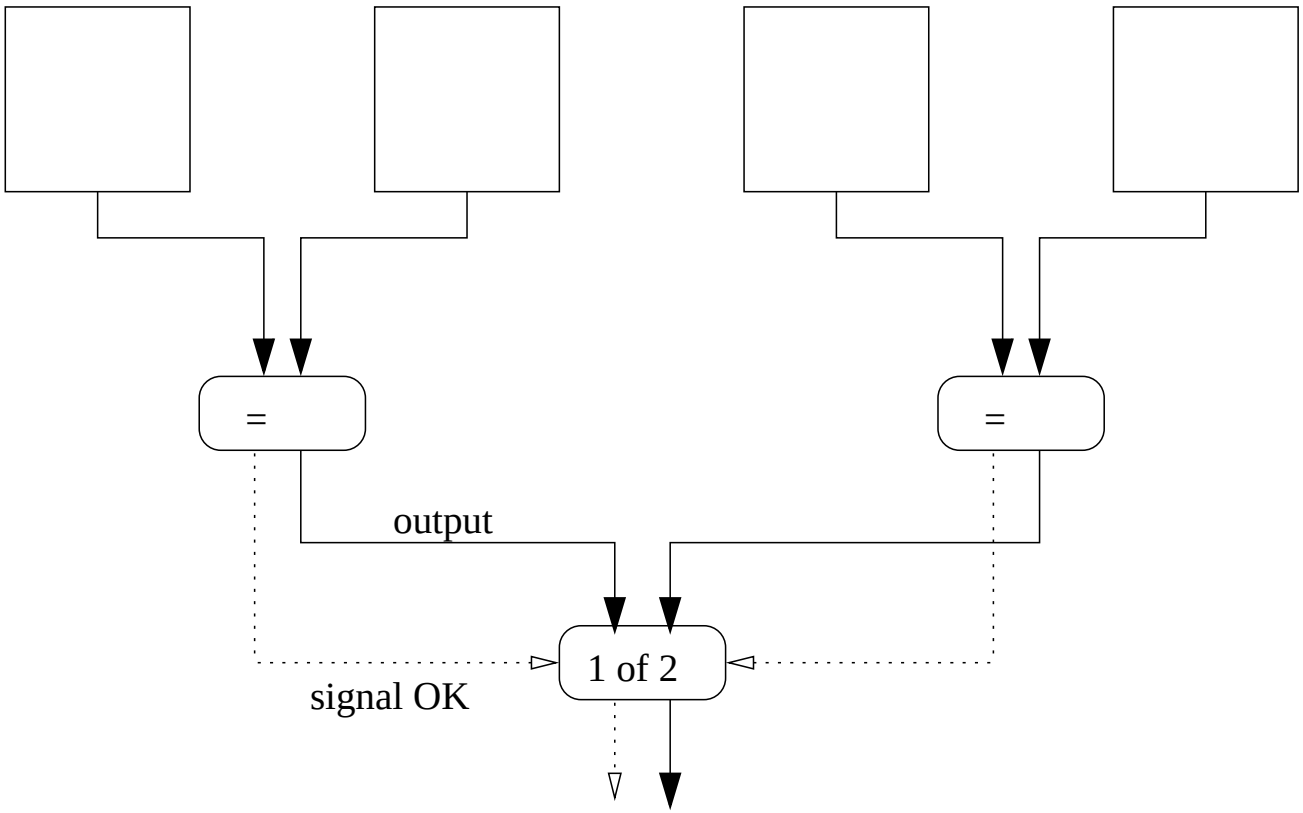
Fault detection using a DMR system
(double modular redundancy)



DMR detects single faults. Fault-detection only, continuation of processing must be arranged by other techniques.

Fault tolerance techniques (6)

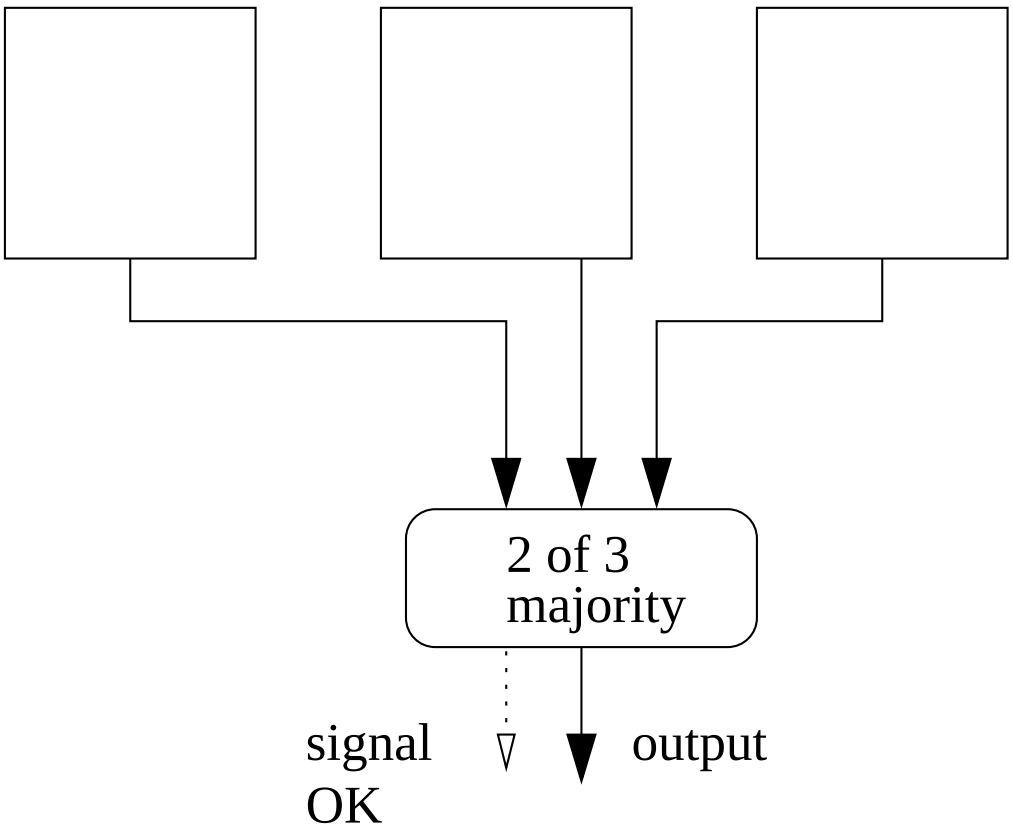
Fault isolation and compensation by a PSR system (pair and spare redundancy)



Tolerates faults of single components.

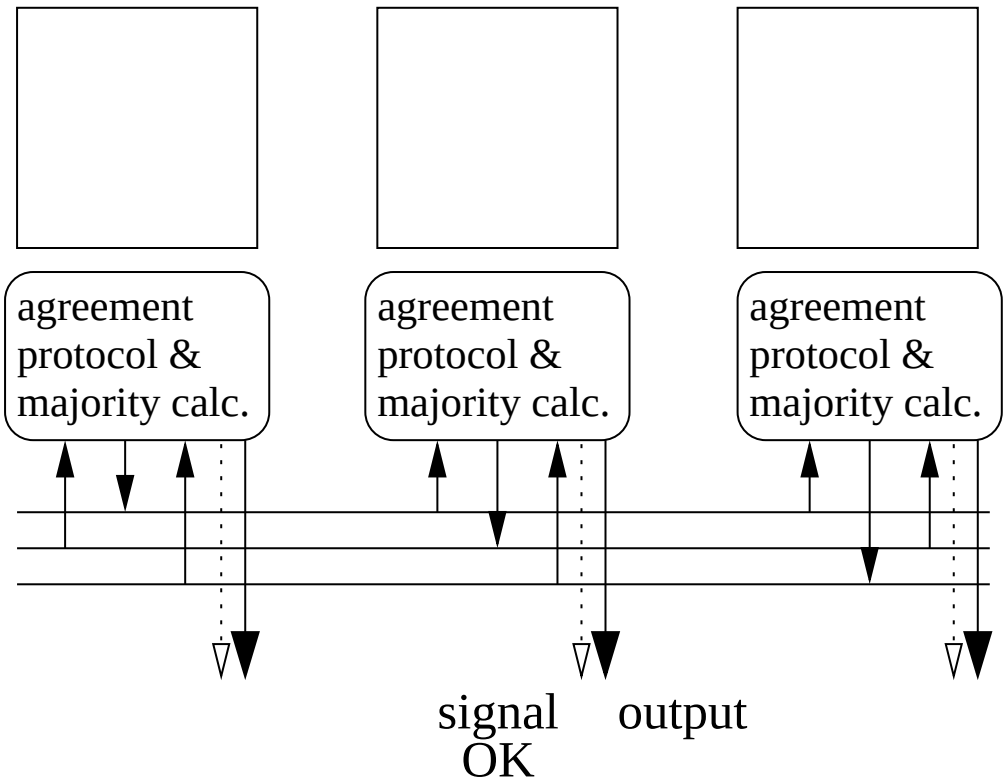
Fault tolerance techniques (7)

Fault compensation using a TMR system
(triple modular redundancy)



Fault tolerance techniques (8)

Fault compensation using replication with distributed majority voting



Functional Layers of Systems (1)

Structuring of a system into functional layers (Hardware, Device driver, Operating system, Middleware, Application)

Faults cause different effects (fault classes) in different layers, the effect (in terms of a fault class) may change from bottom layers to top layers

Example: a faulty sector on a magnetic disk is a permanent fault. It causes intermitting faults for a sequence of data accesses in the application layer

System layers and fault-tolerance techniques (1)

A placement of fault tolerance techniques creates new system levels.

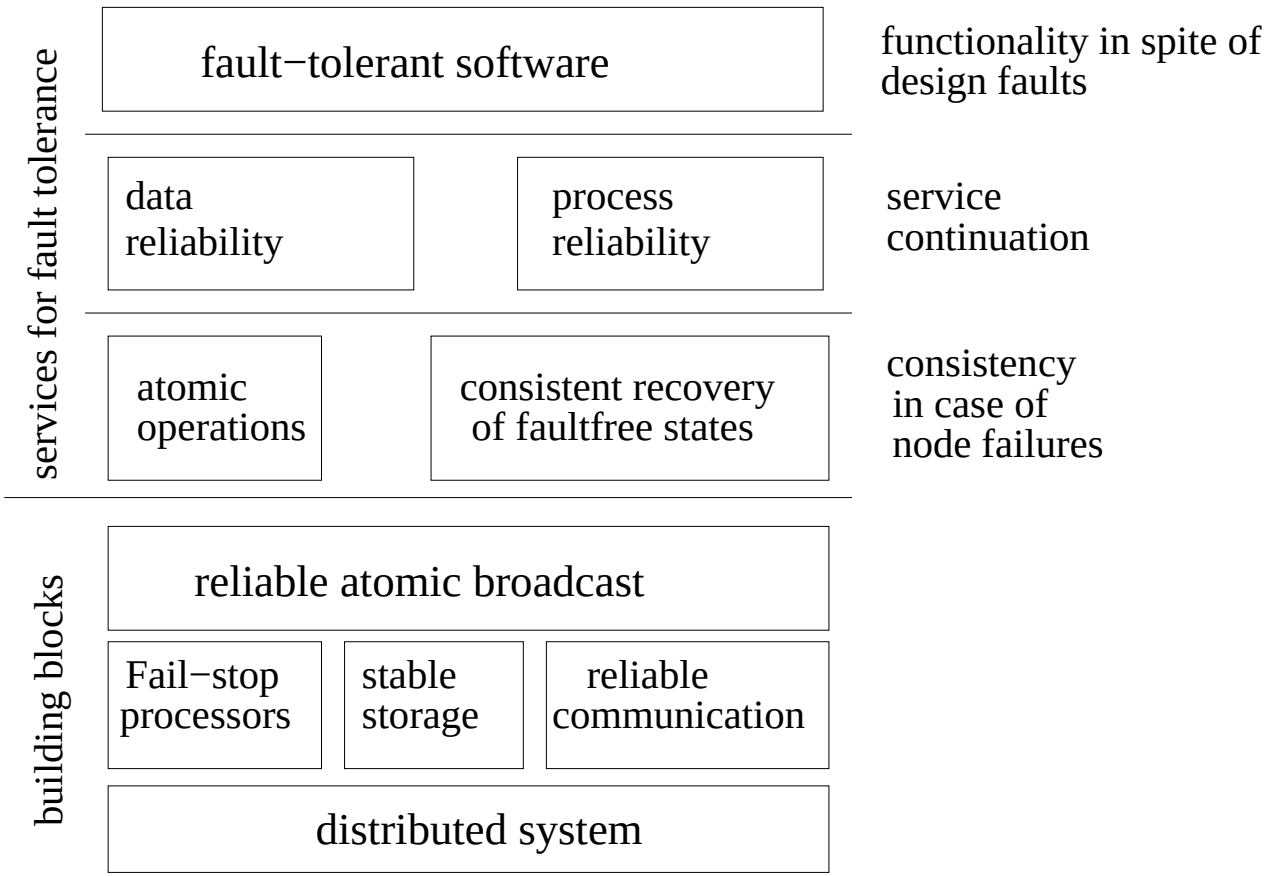
Fault tolerance techniques can make use of other fault tolerance techniques in lower system levels.

Example:

- Bottom level with Fault detection and reconfiguration
- Upper level with Recovery (checkpointing and rollback)

System layers and fault-tolerance techniques (2)

A suggested layering of fault tolerance techniques



System layers and fault-tolerance techniques (3)

Layering of techniques :

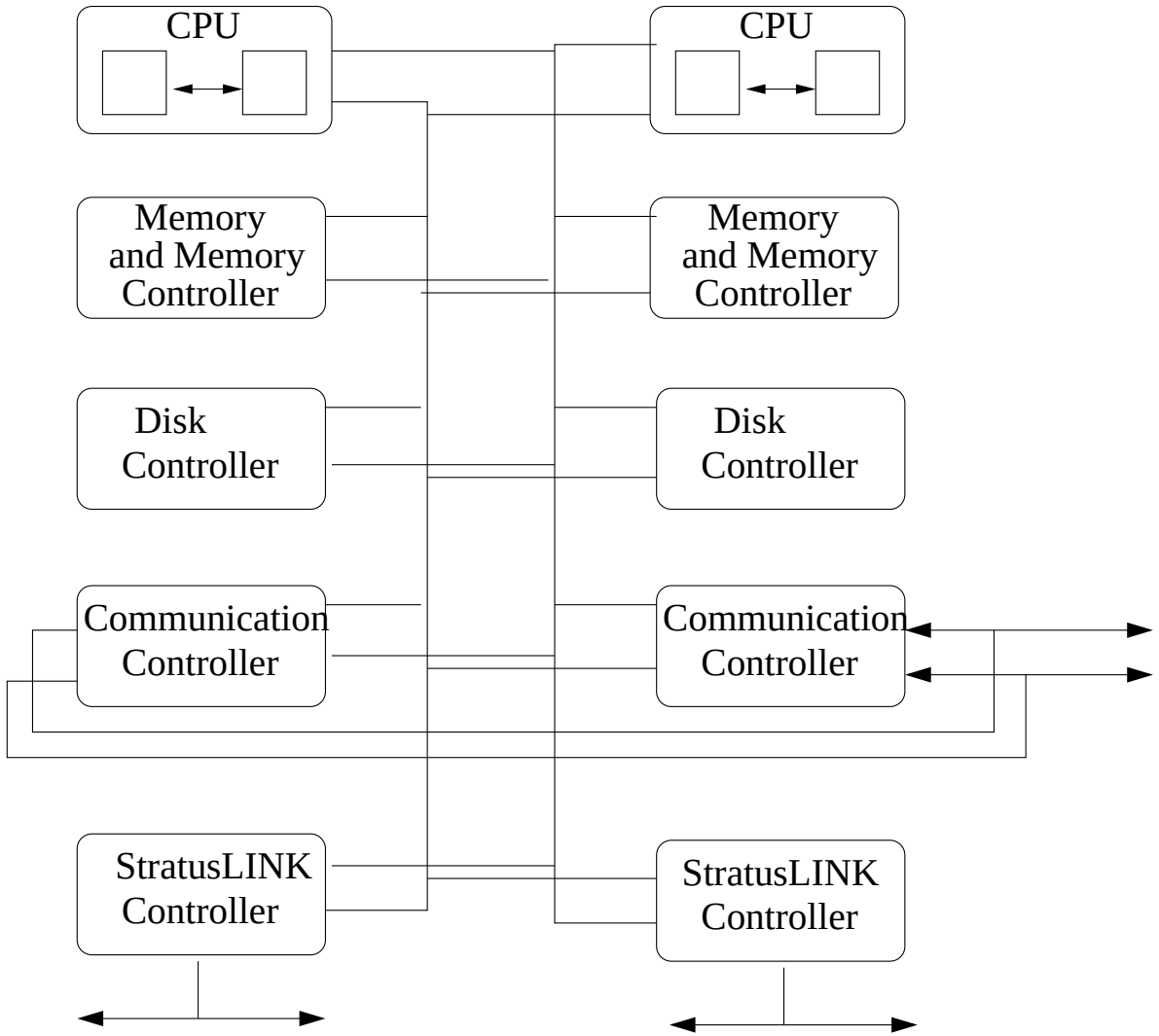
- Stable storage for storage of checkpoints (application states).
- Assumption of Fail-Stop-Processors (fault model)
- A simple pairwise fault detection (I-am-alive-messages)
- Creation of Checkpoints in a fixed time schedule
- On detection of a fault, stop of the application
- In case of an application stop due to a fault: Restart from the last valid checkpoint

Base techniques for fault tolerance (1)

- Stable storage: for fault tolerant storage of checkpoints and application data
- Fail-Stop-processors (-nodes): in result to a detected fault, output and message sending is stopped by technical means. The processor (or node) seems to behave in a fail-stop manner
- Reliable Communication: All faultfree nodes must be able to communicate with each other. The network is responsible for this property. Application of redundant communication lines, fault tolerant routing, regulated sending (avoidance of so called ”babbling idiots”)
- Atomic Broadcast: consistent distribution of information, all-or-nothing property (all fault free receivers must receive the message), uniform order of message arrival in case of multiple broadcasts

Examples (1)

Stratus (Tandem/Compaq): Structure of a PSR modul



Fault tolerance is transparent to the application

Examples (2)

Modern fault-tolerant systems:

- Appliction of commodity-off-the-shelf hardware (that is not fault-tolerant and not highly reliable) in combination with distributed systems and software-implemented fault tolerance
- Simple approach: Doubling of all critical components, e.g. two web servers, two communication networks
- Distributed Frameworks for fault-tolerant processing in distributed systems

Summary of Part 2

- A variety of fault tolerance techniques that can be applied depending on the class of faults to be tolerated
- Fault detection strongly depends on the fault classes
  - Alive test and timeout checks
  - Acceptance checks, flow control checks, process duplication
- Fault treatment often by reconfiguration (repair) and rollback
- Fault masking by pair and spare redundancy, TMR and NMR configurations
- Replicated process groups with agreement
- Redundancy is a necessary base
- Distributed systems support fault tolerance by the presence of multiple nodes