Distributed Algorithms and Fault Tolerance

Introduction to Fault Tolerance

Thomas Ropars

thomas.ropars@imag.fr

ERODS research team – LIG/IM2AG/UGA

2015

Agenda

- Part 1: Introduction to fault tolerance
- Part 2: Error detection
- Part 3: Error recovery
- Part 4: Fault tolerance for distributed systems

Murphy's law

Whatever can go wrong will go wrong at the worst possible time and in the worst possible way.

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

Large scale computing systems

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

Large scale computing systems

Computing systems are everywhere

Usage of computing systems

- Flight controls in airplanes
- Car braking system
- Financial transactions
- Weather forecasting
- Store photos from last vacation
- Make this presentation

Computing systems are everywhere

Usage of computing systems

- Flight controls in airplanes
- Car braking system
- Financial transactions
- Weather forecasting
- Store photos from last vacation
- Make this presentation

How much can we rely on computing systems?

Computing systems can fail

Some failure causes

- The running environment (heat, humidity, unstable power supply)
- The users (do users know how to properly operate the system?)
- The manufacturing (in many contexts, computing systems are based on cheap components)
- The scale (very large scale systems have a higher probability to fail)
- Software bugs

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

Large scale computing systems

Dependability

Avizienis, Laprie and Randell (2001)

Definition

Dependability of a computing system is the ability to deliver service that can justifiably be trusted

- The service delivered by a system is its behavior as it is perceived by its user(s)
- A user is another system (physical, human) that interacts with the former at the service interface

Dependability

Avizienis, Laprie and Randell (2001)

- The function of a system is what the system is intended for, and is described by the system specification
- Correct service is delivered when the service implements the system function
- A system failure is an event that occurs when the delivered service deviates from correct service

Alternative definition of dependability

The ability of a system to avoid failures that are more frequent or more severe than is acceptable to the user(s).

System failures

System failures types

- Value failure: The output of the system is not acceptable for the given set of inputs
- Timing failure: The output of the system is not available within some expected time bounds

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

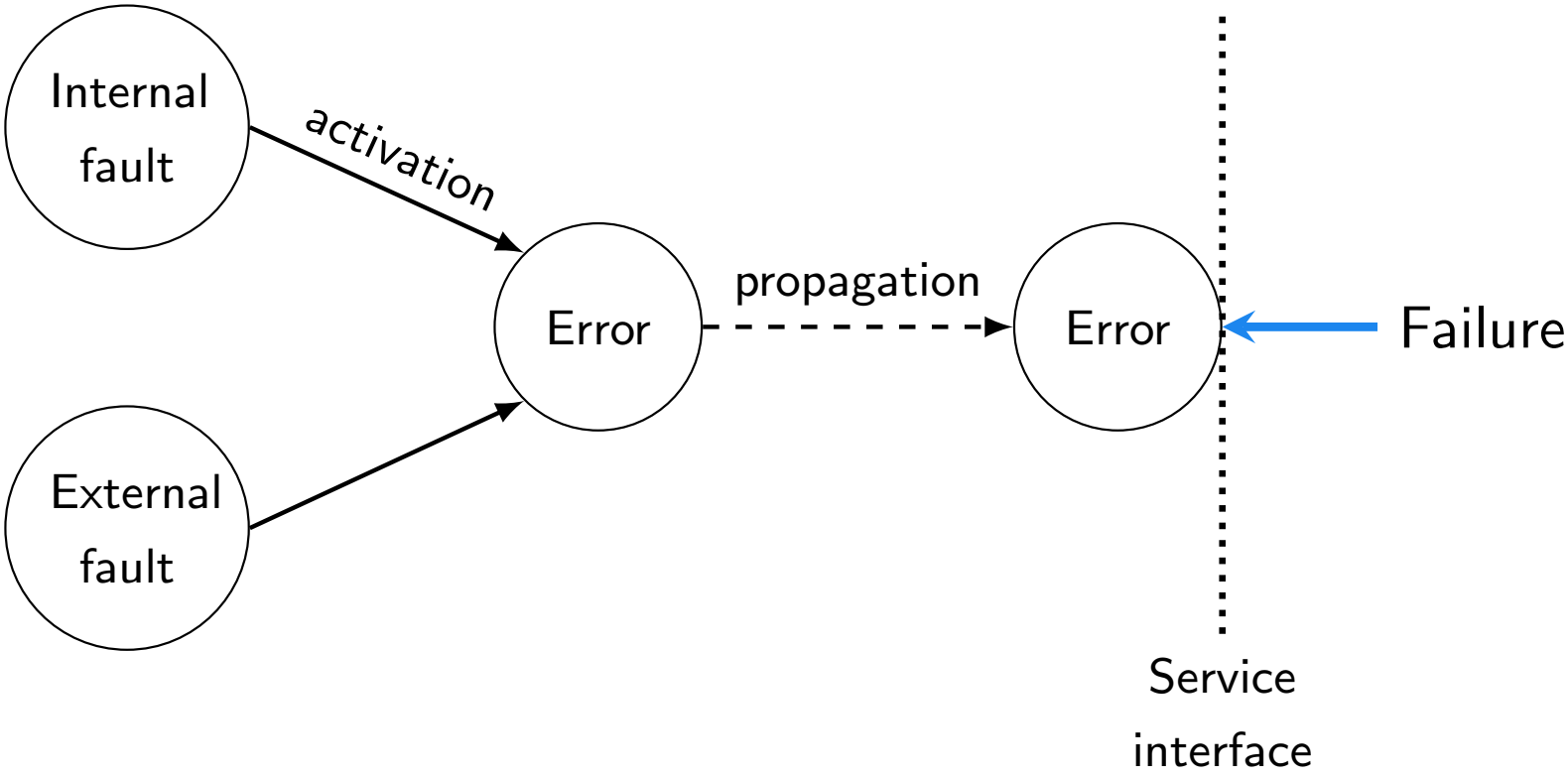
Large scale computing systems

Faults, Errors, Failures

Avizienis, Laprie and Randell (2001)

An error is the part of the system state that may cause a subsequent failure: a failure occurs when an error reaches the service interface and alters the service

A fault is the adjudged or hypothesized cause of an error



Fault-Error-Failure example

- Fault: High energy particle disrupts charge on DRAM transistor.
- Error: Bit flip in a memory world
- Failure: Computation produce incorrect result

Faults

Three major fault classes

- Design faults
- Physical faults
- Interaction faults

Faults

Three major fault classes

- Design faults
  - Software flaws
  - Hardware flaws
- Physical faults
- Interaction faults

Faults

Three major fault classes

- Design faults
  - Software flaws
  - Hardware flaws
- Physical faults
  - Physical deterioration
  - Physical interference
- Interaction faults

Faults

Three major fault classes

- Design faults
  - Software flaws
  - Hardware flaws
- Physical faults
  - Physical deterioration
  - Physical interference
- Interaction faults
  - Input mistakes
  - Attacks

Characterization of Faults

- Domain
  - Hardware faults
  - Software faults
- Intent
  - Non-malicious
  - Malicious

Characterization of Faults

- Domain
  - Hardware faults
  - Software faults
- Intent
  - Non-malicious
  - Malicious

There can be non-malicious deliberate faults:

Characterization of Faults

- Domain
  - Hardware faults
  - Software faults
- Intent
  - Non-malicious
  - Malicious

There can be non-malicious deliberate faults:

- Ex: During development, correctness might have to be traded for performance or simplicity

Characterization of Faults: Persistence

Transient (soft) faults/errors

- Occurs once and disappears
- Eg, bit-flip due to high-energy particles
- Tend to be due to transient physical phenomena

Intermittent faults/errors

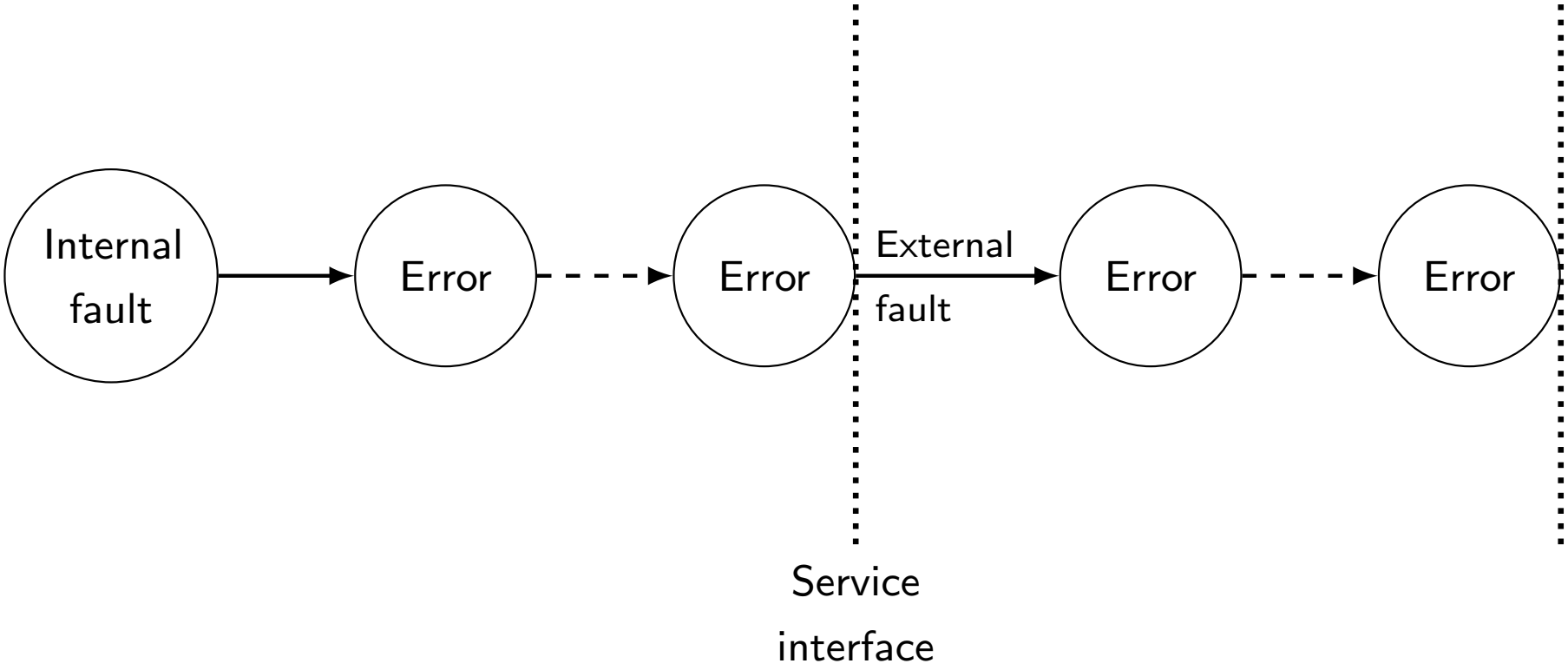
- Occurs occasionally
- Eg, a router drops some packets

Permanent (hard) faults/errors

- Occurs and doesn't go away
- Eg, a dead power supply

About faults and failures

The failure of a component is a fault from the perspective of the component using it.



Characterization of failures

- Catastrophic failures: The system is down. Large impact on the users and/or the environment.
- No response: The system does not respond anymore.
- Data failures: The output is provided but it is a wrong value.
- Timing failures: System returns a good behavior but out of the timing expectations
  - A system with timing constraints is called *real-time*.
- Random failures (Byzantines) - The system provides *random* outputs to the same inputs
  - Different users might observe different behaviors
  - Values might not be that random (malicious behavior)

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

Large scale computing systems

The attributes of dependability

Reliability

- The ability of a component to perform its required functions for a specified period of time

Availability

- The degree to which a component is operational and accessible when required for use

Safety

- Absence of catastrophic consequences on the user and environment

Confidentiality

- Absence of unauthorized disclosure of information

Integrity

- Absence of improper system state alteration

Maintainability

- Ease of repairing a failed system

About security

Security can be seen as a composite notion:

- Confidentiality
- Integrity: the prevention of the unauthorized modification or deletion of information
- Availability: the prevention of the unauthorized withholding of information

Reliability and MTBF

Definitions

- Reliability: A function of time $R(t)$ defined as the conditional probability that the system works properly during a time interval $[t_0, t]$ given that the system worked correctly at t_0.
- MTBF: Mean time between failures

$$R(t) = e^{-(\frac{t}{MTBF})}$$

More about MTBF

Typical MTBF of a disk range between 30 and 120 years (source: seagate)

- Does this mean that my disk can run for 30 years without failures?

More about MTBF

Typical MTBF of a disk range between 30 and 120 years (source: seagate)

- Does this mean that my disk can run for 30 years without failures?
  - No, this MTBF does not take into account aging
  - This number corresponds to the MTBF during normal life (e.g., 3 years)

More about MTBF

Typical MTBF of a disk range between 30 and 120 years (source: seagate)

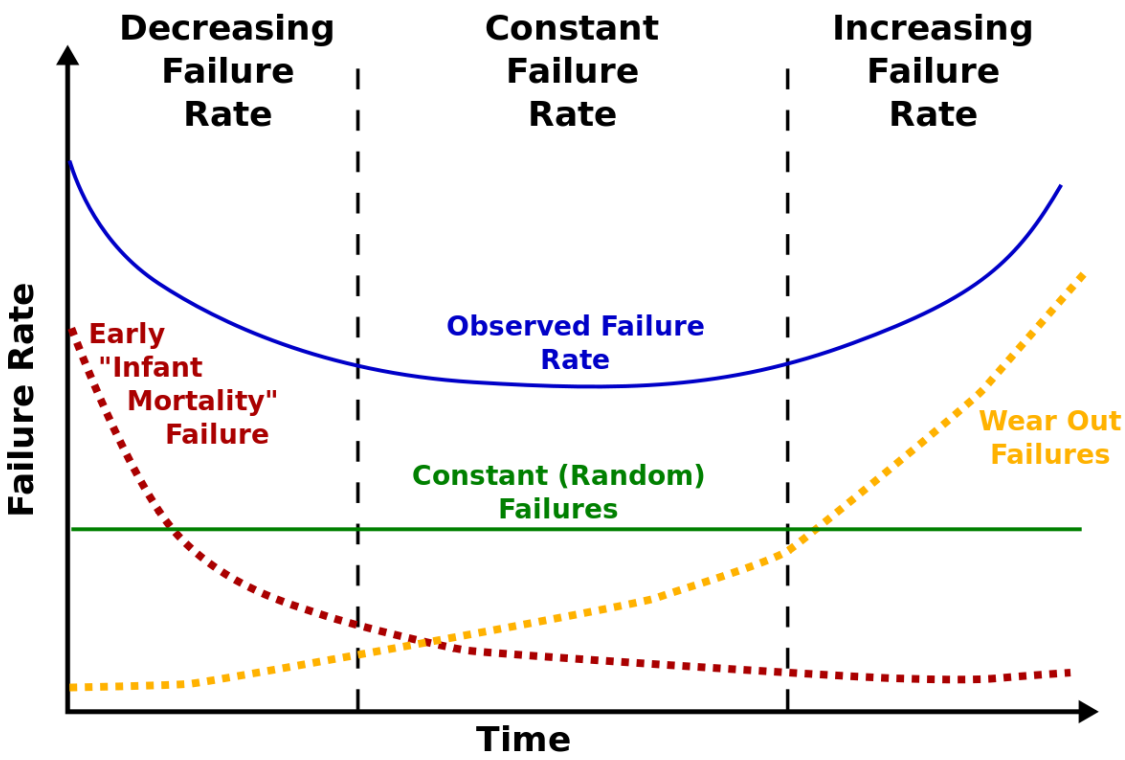
- Does this mean that my disk can run for 30 years without failures?
  - No, this MTBF does not take into account aging
  - This number corresponds to the MTBF during normal life (e.g., 3 years)

Computation

- 1000 disks run during 6 months and only 6 fail.
- Failure rate $= \frac{6}{1000 \times 0.5} = 0.012$
- MTBF $= \frac{1}{0.012} \simeq 83$ years
- But the failure rate increases with age

More about MTBF

The bathtub



- Infant mortality is due to defective products
- During normal operation, failure rate is low and almost constant

Reliability and MTBF

$$R(t) = e^{-(\frac{t}{MTBF})}$$

MTBF	Reliability in % (1 year)
120 years	99.2
30 years	96.7
5 years	81.8

Note that MTBF is very often given in number of hours. Here we use *years* as unit for the sake of readability.

MTBF of complex systems

In a system integrating many components, the failure of any of the components can result in the failure on the whole system.

We use 1000 disks to build a large storage server.

- Recall: 1000 disks run during 6 months and only 6 fail.
- Failure rate $= \frac{6}{1 \times 0.5} = 12$
- MTBF $= \frac{1}{12} = 1$ month
- Note that most data are still available when a disk fails

MTBF range of other complex systems

- A laptop/desktop
  - Typical MTBF in the order of 3 years
- A datacenter
  - Built out of 1000 *low-cost* nodes
  - MTBF $= 3/1000 \simeq 26$ hours
  - Large scale datacenters are in the scale of tens of thousands of nodes
  - Note that in this context, the failure of a node usually does not prevent the system from functioning.
- A supercomputer
  - Typical MTBF of a node $= 5$ years
  - Largest supercomputers $= 100000$ nodes
  - System MTBF $= 26$ minutes
  - Bad news: applications are usually tightly coupled

Availability

Definition

- The degree to which a component is operational and accessible when required for use.
- The proportion of time for which a component is able to perform its function

To compute availability, we need to introduce one more parameter

Availability

Definition

- The degree to which a component is operational and accessible when required for use.
- The proportion of time for which a component is able to perform its function

To compute availability, we need to introduce one more parameter

- The mean time to repair (MTTR)

$$A = \frac{Uptime}{Uptime + Downtime} = \frac{MTBF}{MTBF + MTTR}$$

Availability: Some facts

Gray and Siewiorek (1991)

- Computer built in the late 50's
  - MTBF = 12 hours
  - MTTR = 8 hours (a dozen of fulltime engineers to repair the machine)
  - A = 60%
- Telephone switching system
  - Designed to be unavailable less than 5 minutes a year
  - A = 99.999% (*five nines availability*)
  - This is a highly available system

Availability classes

System type	Unavailability (min/year)	Availability
Managed	5256	99 %
Fault tolerant	53	99.99 %
Highly available	5	99.999 %
Ultra available	0.05	99.99999 %

- Typical online services should be made highly available
- Five-nines availability is also targeted for satellites

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

Large scale computing systems

Means to attain dependability

Fault prevention

- How to prevent the occurrence or introduction of faults

Fault tolerance

- How to deliver correct service in the presence of faults

Fault removal

- How to reduce the number or severity of faults

Fault forecasting

- How to estimate the present number, the future incidence, and the likely consequences of faults

Fault prevention

- Act on
  - the procedures for creating and using a product
  - the technologies used for manufacturing a product
- Applied during
  - Specification (avoid incomplete/ambiguous specification)
  - Design (choice of tools)
  - Manufacturing/development (quality standard, automatic tools)
  - Operation (prevent incorrect usage)

Fault prevention

- Act on
  - the procedures for creating and using a product
  - the technologies used for manufacturing a product
- Applied during
  - Specification (avoid incomplete/ambiguous specification)
  - Design (choice of tools)
  - Manufacturing/development (quality standard, automatic tools)
  - Operation (prevent incorrect usage)

Limit: Faults can never be completely eliminated

Fault tolerance

Intuition: Add `something` to the system to allow it to deal with faults

That's the topic of the next lectures

Fault removal

- Detecting, localizing, and removing faults
- Static techniques
  - Formal verification
  - Revision of code by experts
- Dynamic techniques
  - Mainly tests
- Maintenance
  - Corrective maintenance: Aimed at removing faults that have produced one or more errors and have been reported.
  - Preventive maintenance: Aimed to uncover and remove faults before they might cause errors during normal operation.

Fault forecasting

- Study of fault/failure relationship
  - Root cause analysis
- Collection of failure data
  - Failure models
- Failure prediction
  - Hot topic
  - If I am able to predict the failure of a node, I may be able to migrate my service to another node before the failure occurs (virtually continuous availability)
  - Making accurate predictions is not that easy

Fault forecasting

- Study of fault/failure relationship
  - Root cause analysis
- Collection of failure data
  - Failure models
- Failure prediction
  - Hot topic
  - If I am able to predict the failure of a node, I may be able to migrate my service to another node before the failure occurs (virtually continuous availability)
  - Making accurate predictions is not that easy

No matter how much prevention, removal and forecasting is applied, faults will still occur. (but these techniques help reducing the number of faults)

Agenda

Motivations

Definitions

Faults, Errors, Failures

Availability, Reliability and MTBF

Means to attain dependability

Large scale computing systems

Failures causes

Analysis of the failures on an extreme scale supercomputer (Di Martino et al (2014))

- 75% of the (system-wide outage) are due to software
- Software includes fail-over procedures . . . that fail in a vast majority of cases
- A huge number of faults occurs in hardware. Few lead to failures.
  - 1.5 millions of memory errors in less than a year but only 0.003% failures.
- Hardware is responsible for 51% of the failures but only 23% of the total repair time.

Failures causes

Analysis of the failures on an extreme scale supercomputer (Di Martino et al (2014))

- 75% of the (system-wide outage) are due to software
- Software includes fail-over procedures . . . that fail in a vast majority of cases
- A huge number of faults occurs in hardware. Few lead to failures.
  - 1.5 millions of memory errors in less than a year but only 0.003% failures.
- Hardware is responsible for 51% of the failures but only 23% of the total repair time.

Lessons learned:

- Hardware fault tolerance techniques are very efficient
- Software errors can lead to *big* failures

Failures causes

Analysis of Business Data Processing cloud apps (Di Martino et al (2012))

- 34% of the failures are due to unexpected user data
- 32% of the failures are timeouts
- 95% of the failures are located in the same 5 modules (37% of LOC)

Failures causes

Analysis of Business Data Processing cloud apps (Di Martino et al (2012))

- 34% of the failures are due to unexpected user data
- 32% of the failures are timeouts
- 95% of the failures are located in the same 5 modules (37% of LOC)

Lessons learned:

- Exceptions need to be better handled
- A small number of errors create the vast majority of the failures $\rightarrow$ Better testing procedures are required

Failures in the cloud

Pham, Cao et al (2012)

Some examples

- Microsoft Azure (2009)
  - 1-day system slow-down (many server failures)
  - Source: Ongoing operating system update
  - Users can create a copy of their application on a different site for free
- Amazon S3 (2008)
  - 8-hour outage
  - Source: A single-bit corruption in a handful of messages

More failures in the cloud

- Amazon Web Services data center (2015)
  - 6-hour outage
  - Source: Problem with metadata management in DynamoDB (NoSQL database)
  - The failure propagated to many other services
  - The two biggest clients of AWS are Amazon.com and Netflix
    - Netflix claimed not to be impacted
    - Analyses of consumers complains show a large increase during that period

Cost of an outage

- Amazon.com experienced a 45 minutes outage in 2013
- Cost in lost sales estimated to $4 millions

References

- A. Avizienis, J. C. Laprie, and B. Randell (2001). *Fundamental concepts of dependability*.
- D. Sorin (2009). *Fault Tolerant Computer Architecture*.

- Other sources:
  - Lecture notes of N. Palix
  - Lecture notes of D. Sorin