Ingineria Calculatoarelor
Fault Tolerant Computing

- Cursul 1 -

Facultatea de Automatică și Calculatoare
Universitatea Politehnica București

XKCD of the Day

http://xkcd.com/292/

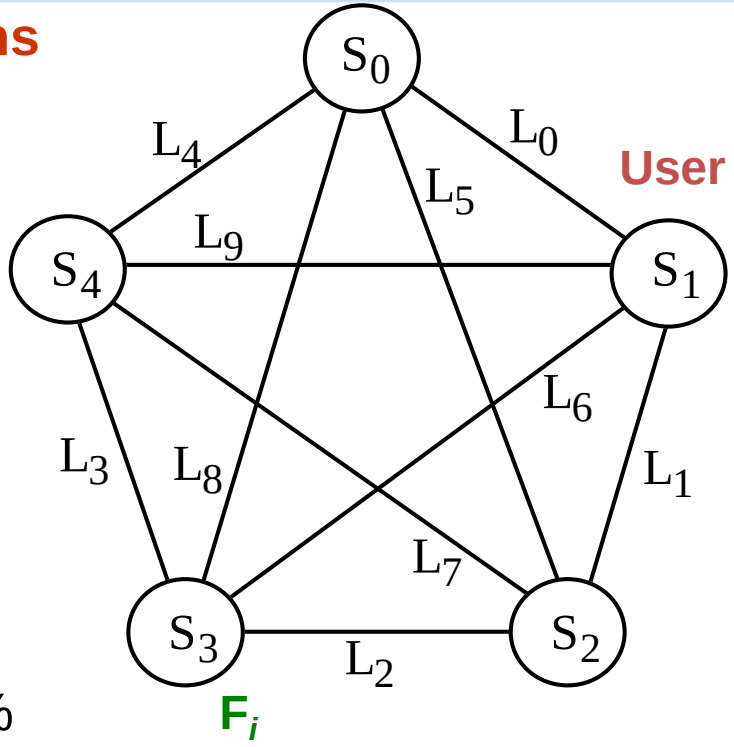
A Case Study

Data availability and integrity concerns

Distributed DB system with 5 sites
Full connectivity, dedicated links
Only direct communication allowed
Sites and links may malfunction
Redundancy improves availability

S: Probability of a site being available
L: Probability of a link being available

Single-copy availability = *SL*
Unavailability = 1 – *SL*
= 1 – 0.99 × 0.95 = 5.95%



Data replication methods, and a challenge

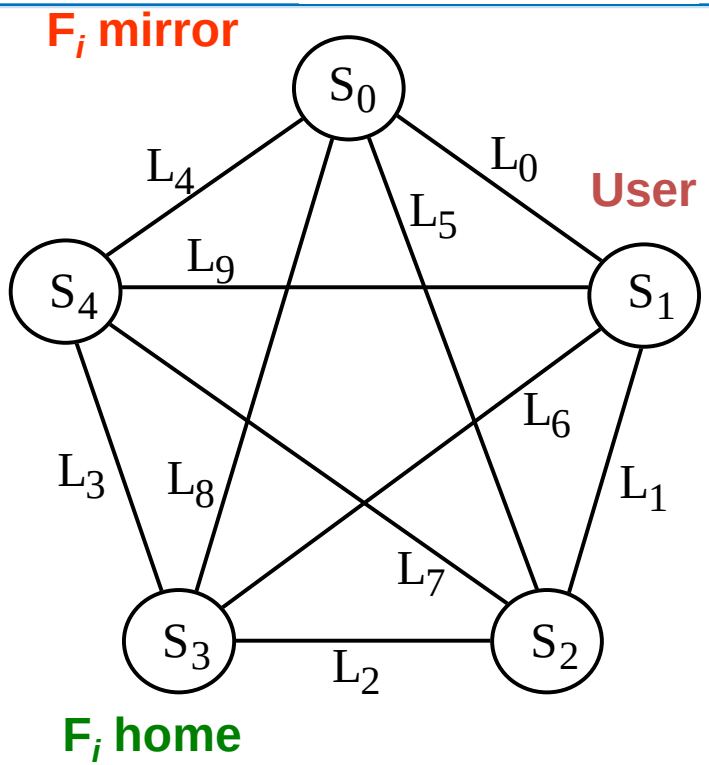
File duplication: home / mirror sites
File triplication: home / backup 1 / backup 2
Are there availability improvement methods with less redundancy?

Data Duplication: Home and Mirror Sites

S: Site availability e.g., 99%
L: Link availability e.g., 95%

$$A = SL + (1 - SL)SL$$

- SL: Primary site can be reached
- SL: Mirror site can be reached
- $(1 - SL)SL$: Primary site inaccessible

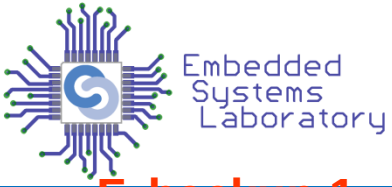


Duplicated availability = $2SL - (SL)^2$
Unavailability = $1 - 2SL + (SL)^2$
= $(1 - SL)^2$ = 0.35%

Data unavailability reduced from 5.95% to 0.35%

Availability improved from ≈ 94% to 99.65%

Data Triplication: Home and Two Backups



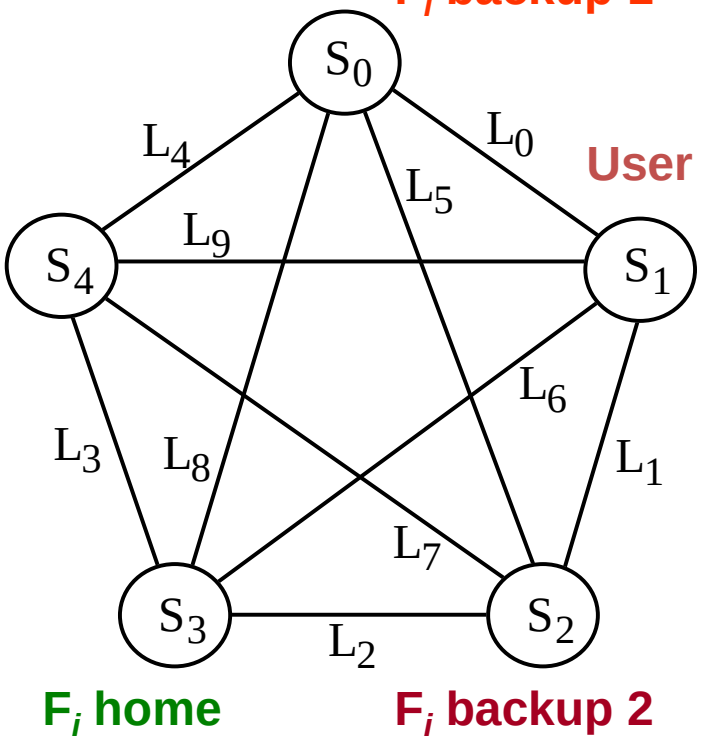
S: Site availability e.g., 99%
L: Link availability e.g., 95%

$$A = SL + (1 - SL)SL + (1 - SL)^2SL$$

- SL: Primary site can be reached
- $(1 - SL)SL$: Primary site inaccessible
- $(1 - SL)^2SL$: Primary and backup 1 inaccessible

Backup 1 can be reached

Backup 2 can be reached

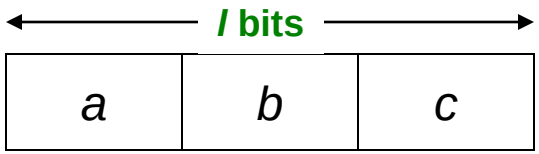


Triplicated avail. $= 3SL - 3(SL)^2 - (SL)^3$
Unavailability $= 1 - 3SL - 3(SL)^2 + (SL)^3$
$= (1 - SL)^3 = 0.02\%$

Data unavailability reduced from 5.95% to 0.02%

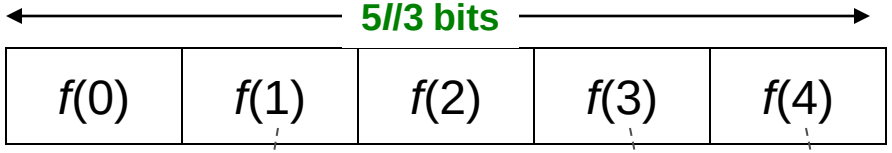
Availability improved from ≈ 94% to 99.98%

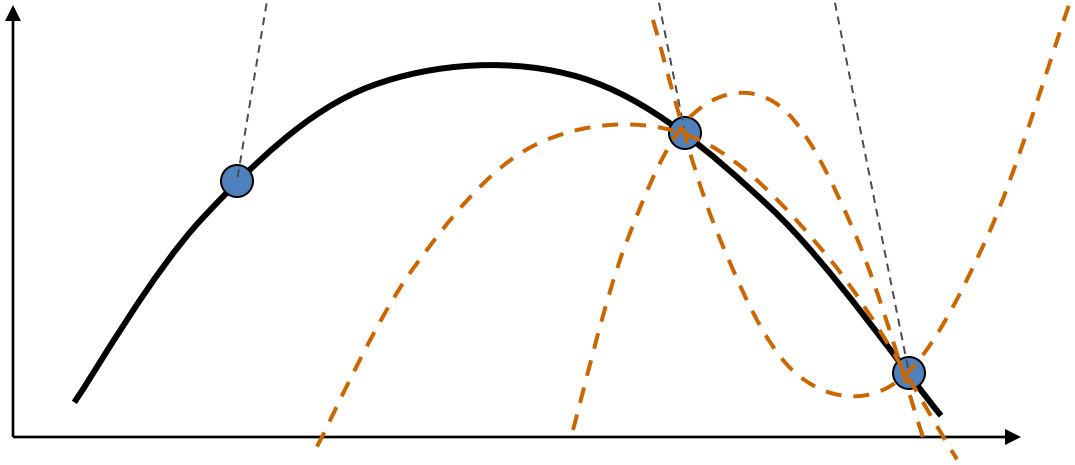
Dispersion for Data Security and Integrity

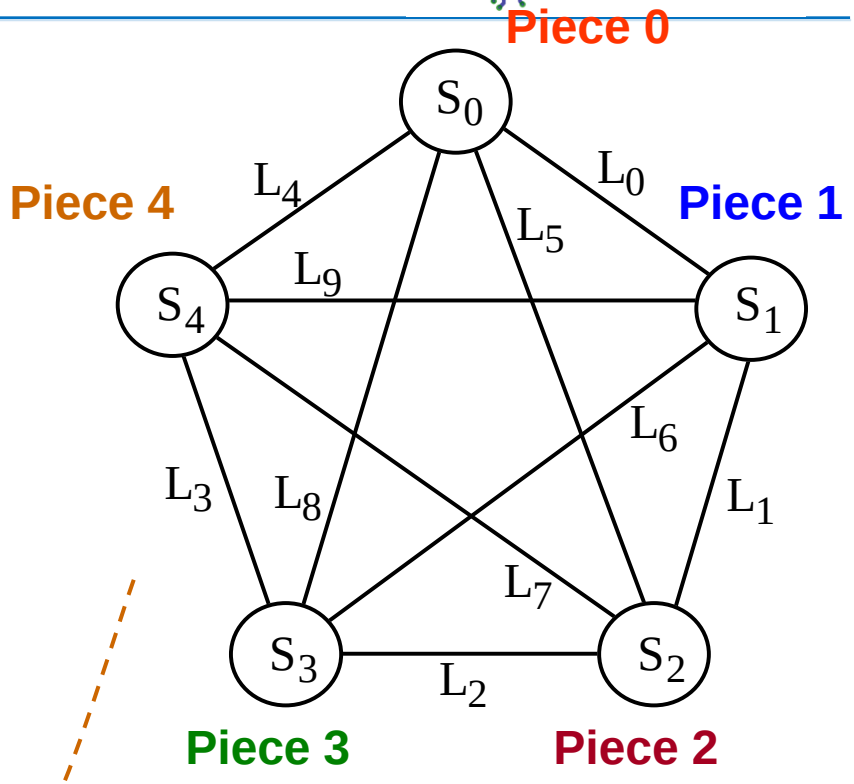


Encoding with
67% redundancy

$f(x) = ax^2 + bx + c$







Note that two pieces would be inadequate for reconstruction

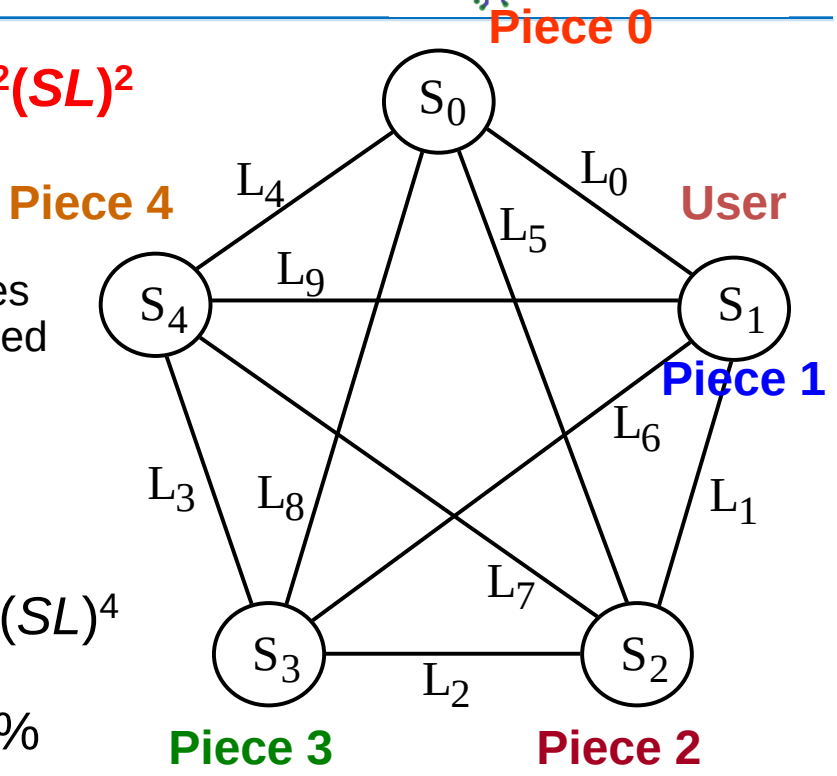
Data Dispersion: Three of Five Pieces

$$A = (SL)^4 + 4(1 - SL)(SL)^3 + 6(1 - SL)^2(SL)^2$$

All 4 pieces can be reached

Exactly 3 pieces can be reached

Only 2 pieces can be reached



S: Site availability e.g., 99%
L: Link availability e.g., 95%

Dispersed avail. = $6(SL)^2 - 8(SL)^3 + 3(SL)^4$
Availability = 99.92%
Unavailability = 1 – Availability = 0.08%

Scheme →	Nonredund.	Duplication	Triplication	Dispersion
Unavailability	5.95%	0.35%	0.02%	0.08%
Redundancy	0%	100%	200%	67%

Questions Ignored in Our Simple Example

Embedded Systems Laboratory

1. How redundant copies of data are kept consistent

When a user modifies the data, how to update the redundant copies (pieces) quickly and prevent the use of stale data in the meantime?

2. How malfunctioning sites and links are identified

Malfunction diagnosis must be quick to avoid data contamination

3. How recovery is accomplished when a malfunctioning site / link returns to service after repair

The returning site must be brought up to date with regard to changes

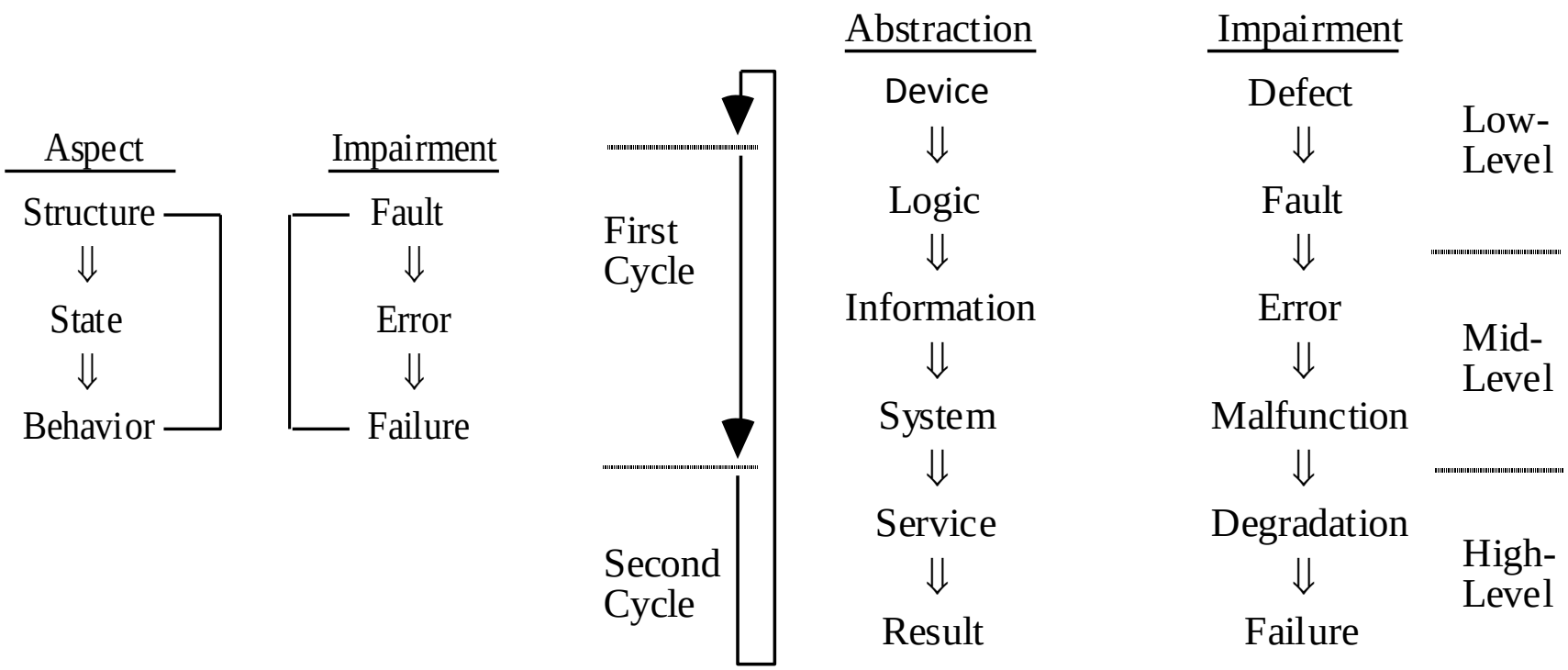
4. How data corrupted by the actions of an adversary is detected

This is more difficult than detecting random malfunctions

The example does demonstrate, however, that:

- Many alternatives are available for improving dependability
- Proposed methods must be assessed through modeling
- The most cost-effective solution may be far from obvious

The Fault-Error-Failure Cycle



Cause-effect diagram for an extended six-level view of impairments to dependability.

A Multilevel Model of Fault Tolerance

Embedded Systems Laboratory

Device

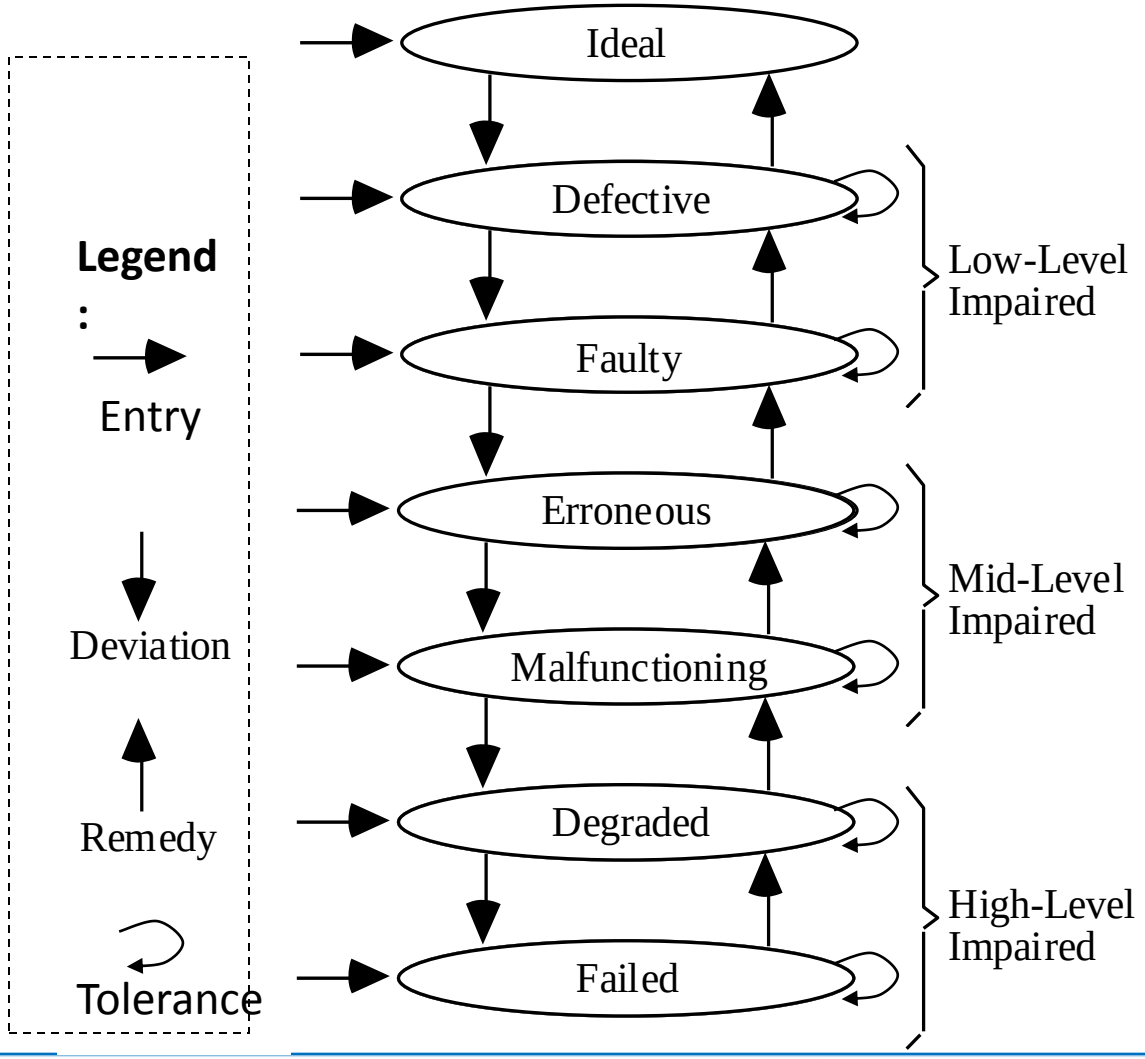
Logic

Information

System

Service

Result



Examples and Analogies

Example: **Automobile brake system**

Defect	Brake fluid piping has a weak spot or joint
Fault	Brake fluid starts to leak out
Error	Brake fluid pressure drops too low
Malfunction	Braking force is below expectation
Degradation	Braking requires higher force or takes longer
Failure	Vehicle does not slow down or stop in time

Note in particular that not every defect, fault, error, malfunction, or degradation leads to failure

Dependable Computer Systems

Long-life systems: Fail-slow, Rugged, High-reliability

Spacecraft with multiyear missions, systems in inaccessible locations

Methods: Replication (spares), error coding, monitoring, shielding

Safety-critical systems: Fail-safe, Sound, High-integrity

Flight control computers, nuclear-plant shutdown, medical monitoring

Methods: Replication with voting, time redundancy, design diversity

Non-stop systems: Fail-soft, Robust, High-availability

Telephone switching centers, transaction processing, e-commerce

Methods: HW/info redundancy, backup schemes, hot-swap, recovery

Just as performance enhancement techniques gradually migrate from supercomputers to desktops, so too dependability enhancement methods find their way from exotic systems into personal computers

Concepts from Probability Theory

Embedded Systems Laboratory

Probability density function: pdf

$f(t) = \text{prob}[t \le x \le t + dt] / dt = dF(t) / dt$

Cumulative distribution function: CDF

$F(t) = \text{prob}[x \le t] = \int_0^t f(x)\,dx$

Expected value of x

$E_x = \int_{-\infty}^{+\infty} x\,f(x)\,dx = \sum_k x_k\,f(x_k)$

Variance of x

$\sigma_x^2 = \int_{-\infty}^{+\infty} (x - E_x)^2\,f(x)\,dx$

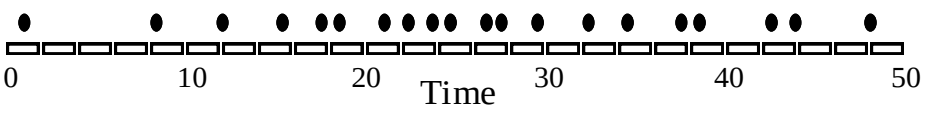
$= \sum_k (x_k - E_x)^2\,f(x_k)$

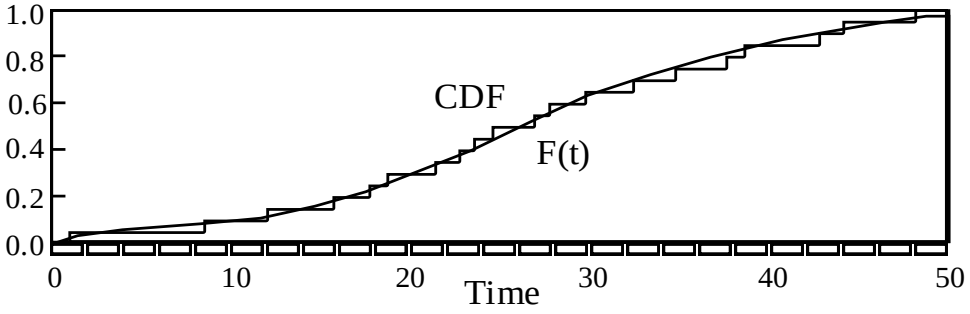
Covariance of x and y

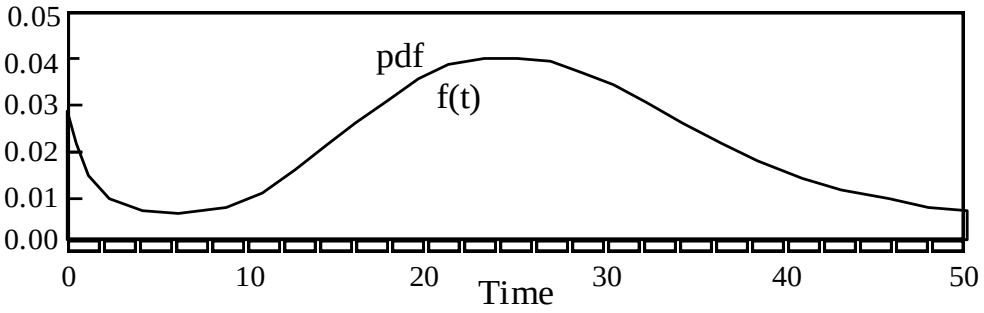
$\psi_{x,y} = E[(x - E_x)(y - E_y)]$

$= E[xy] - E_x E_y$

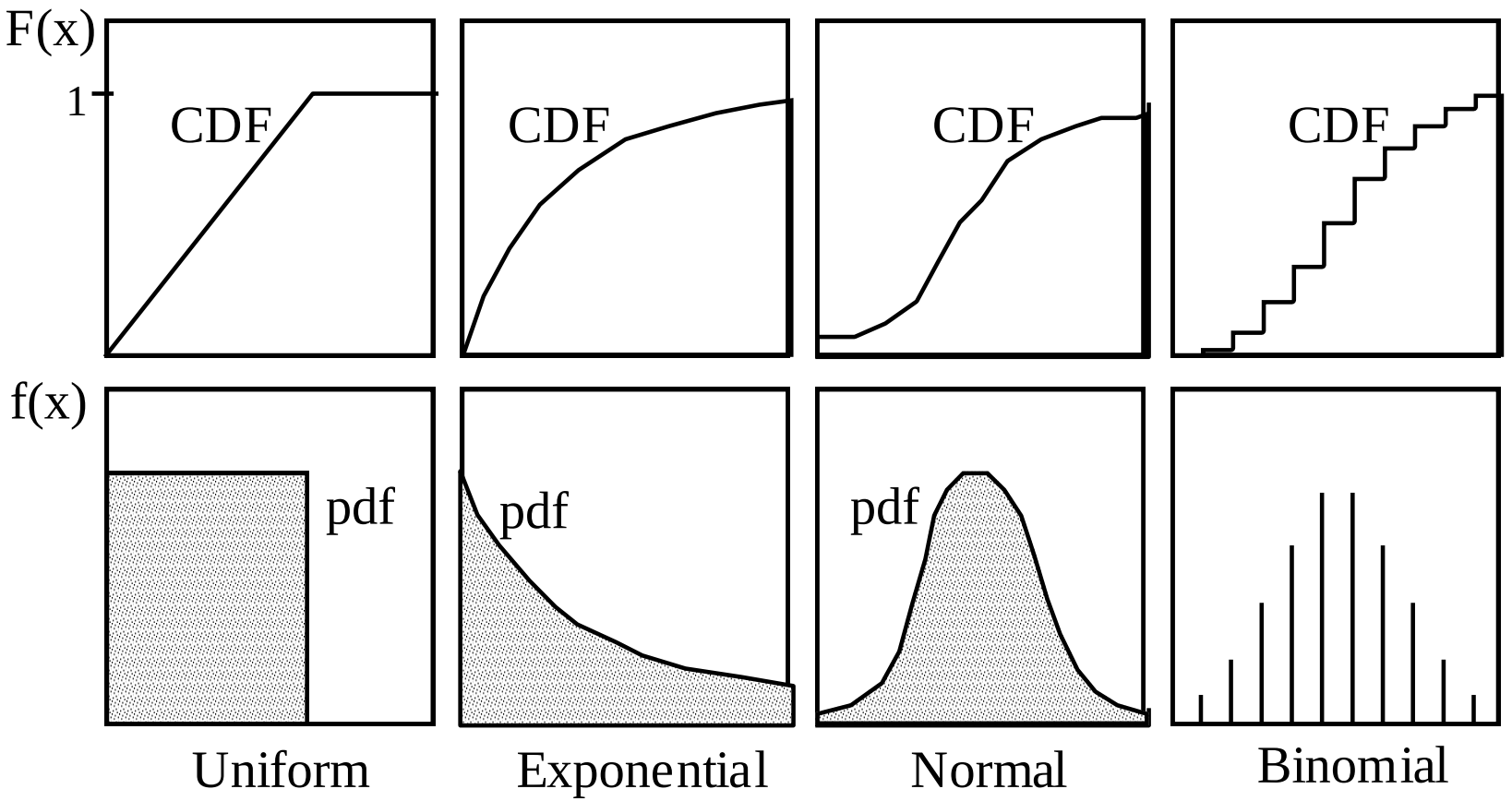
Lifetimes of 20 identical systems





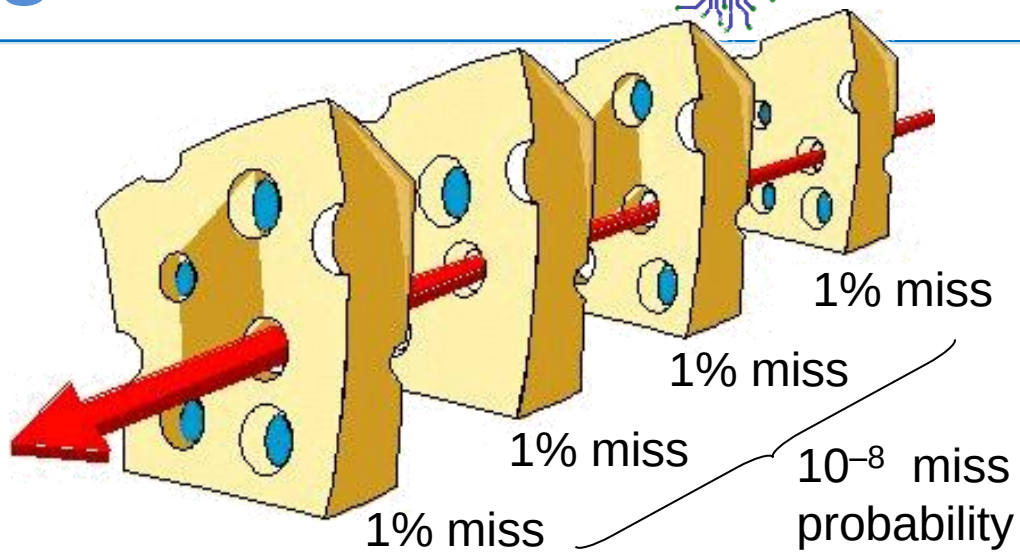


Some Simple Probability Distributions



Layers of Safeguards

With multiple layers of safeguards, a system failure occurs only if warning symptoms and compensating actions are missed at every layer, which is quite unlikely



Is it really?

The computer engineering literature is full of examples of mishaps when two or more layers of protection failed at the same time

Multiple layers increase the reliability significantly only if the “holes” in the representation above are fairly randomly and independently distributed, so that the probability of their being aligned is negligible

Dec. 1986: ARPANET had 7 dedicated lines between NY and Boston; A backhoe accidentally cut all 7 (they went through the same conduit)

Reliability and MTTF

Embedded Systems Laboratory

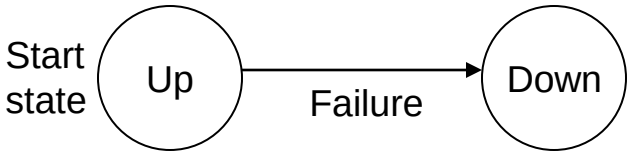
Reliability: $R(t)$

Probability that system remains in the "Good" state through the interval $[0, t]$

Two-state nonrepairable system

$$R(t + dt) = R(t)\,[1 - z(t)\,dt]$$

Hazard function



$R(t) = 1 - F(t)$ ------ CDF of the system lifetime, or its unreliability

Constant hazard function $z(t) = \lambda \Rightarrow R(t) = e^{-\lambda t}$
(system failure rate is independent of its age)

Exponential reliability law

Mean time to failure: MTTF

$$\text{MTTF} = \int_0^{+\infty} t\,f(t)\,dt = \int_0^{+\infty} R(t)\,dt$$

Expected value of lifetime

Area under the reliability curve (easily provable)

Failure Distributions of Interest

Embedded Systems Laboratory

Discrete versions

Exponential: $z(t) = \lambda$

$R(t) = e^{-\lambda t}$ MTTF = $1/\lambda$

Geometric

$R(k) = q^k$

Rayleigh: $z(t) = 2\lambda(\lambda t)$

$R(t) = e^{(-\lambda t)^2}$ MTTF = $(1/\lambda)\ \sqrt{\pi}\ /\ 2$

Weibull: $z(t) = \alpha\lambda(\lambda t)^{\alpha-1}$

$R(t) = e^{(-\lambda t)^\alpha}$ MTTF = $(1/\lambda)\ \Gamma(1 + 1/\alpha)$

Discrete Weibull

Erlang:

MTTF = k/λ

Gamma:

Erlang and exponential are special cases

Normal:

Reliability and MTTF formulas are complicated

Binomial

Comparing Reliabilities

Embedded Systems Laboratory

Reliability difference: $R_2 - R_1$

Reliability gain: R_2 / R_1

Reliability improvement factor

$RIF_{2/1} = [1 - R_1(t_M)] / [1 - R_2(t_M)]$

Example:

$[1 - 0.9] / [1 - 0.99] = 10$

Reliability improv. index

$RII = \log R_1(t_M) / \log R_2(t_M)$

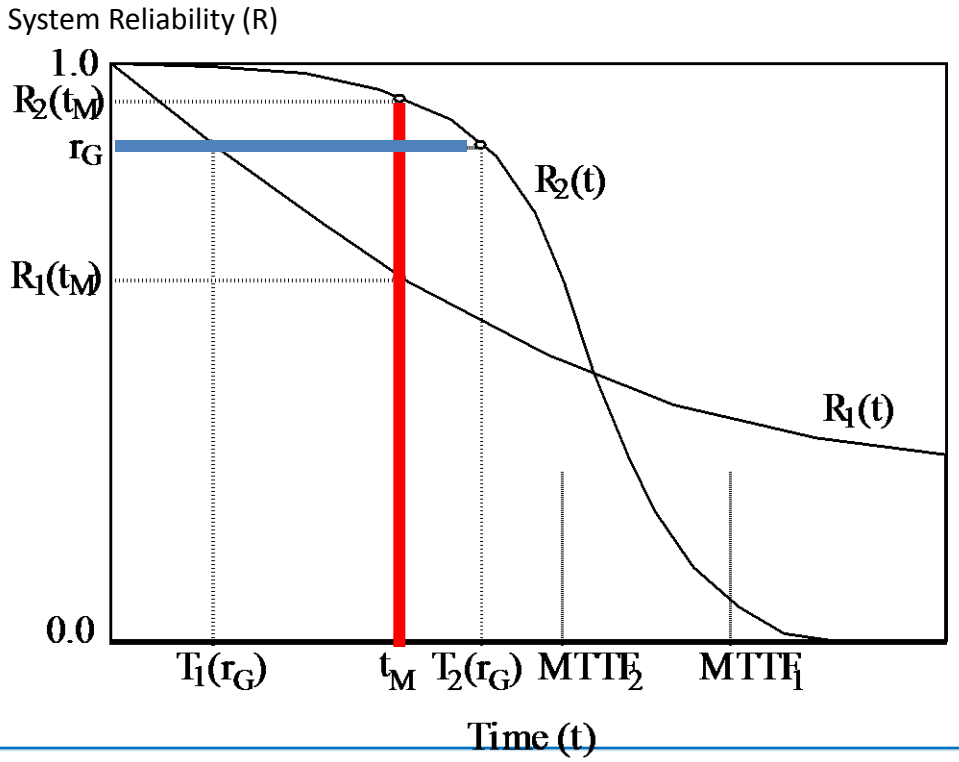
Mission time extension

$MTE_{2/1}(r_G) = T_2(r_G) - T_1(r_G)$

Mission time improv. factor:

$MTIF_{2/1}(r_G) = T_2(r_G) / T_1(r_G)$

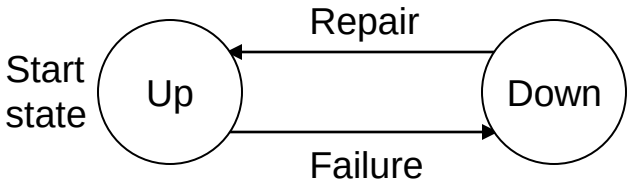
Reliability functions for Systems 1/2

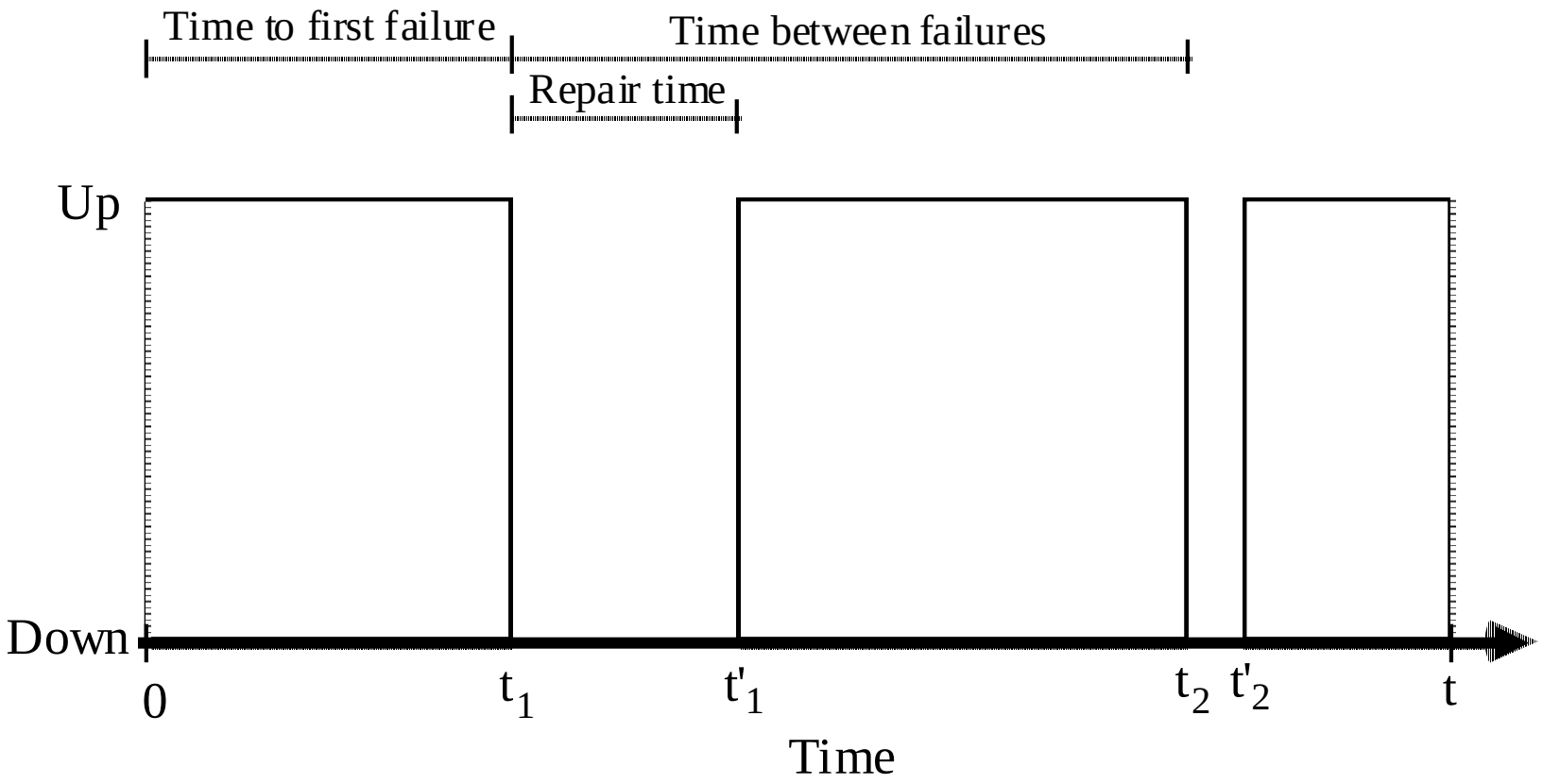


System Up and Down Times

Embedded Systems Laboratory

Short repair time implies
good maintainability (serviceability)





Availability, MTTR, and MTBF

(Interval) Availability: $A(t)$

Fraction of time that system is in the "Up" state during the interval [0, t]

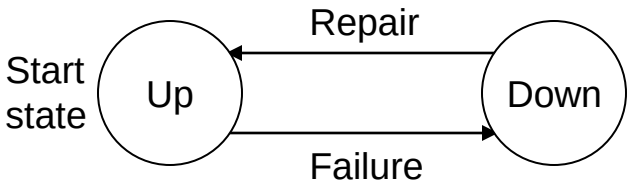
Steady-state availability: $A = \lim_{t\to\infty} A(t)$

Pointwise availability: $a(t)$

Probability that system available at time t

$A(t) = (1/t) \int_0^t a(x)\,dx$

Two-state repairable system



Availability = Reliability, when there is no repair

Availability is a function not only of how rarely a system fails (reliability) but also of how quickly it can be repaired (time to repair)

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} = \frac{\text{MTTF}}{\text{MTBF}} = \frac{\mu}{\lambda + \mu}$$

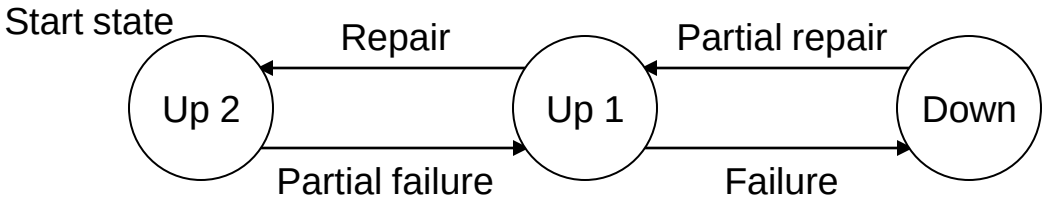
Repair rate $1/\mu = \text{MTTR}$ (Will justify this equation later)

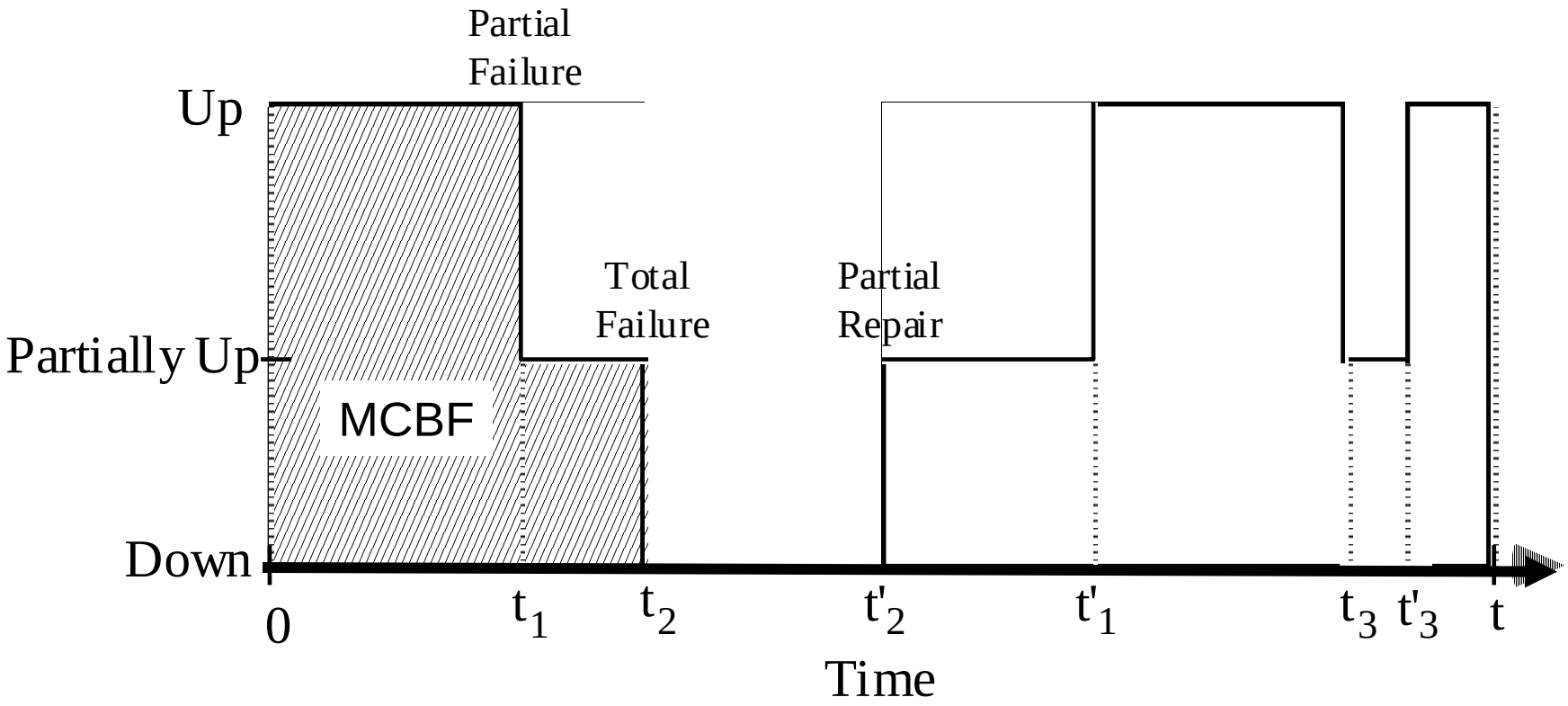
In general, $\mu >> \lambda$, leading to $A \cong 1$

System Up, Partially Up, and Down Times

Embedded Systems Laboratory

Important to prevent direct transitions to the "Down" state (coverage)





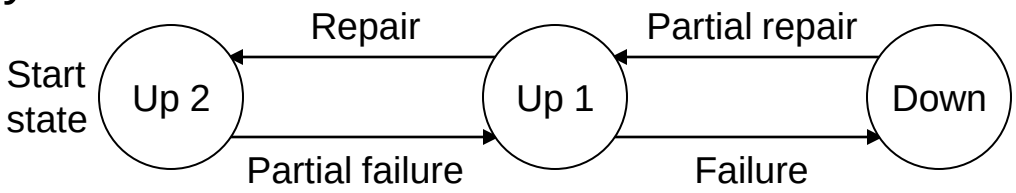
Performability and MCBF

Embedded Systems Laboratory

Performability: *P*

Composite measure, incorporating both performance and reliability

Three-state degradable system



Simple example

Worth of "Up2" twice that of "Up1"

p_{Upi} = probability system is in state Up*i*

$P = 2p_{Up2} + p_{Up1}$

$p_{Up2} = 0.92$, $p_{Up1} = 0.06$, $p_{Down} = 0.02$, $P = 1.90$

(system performance equiv. To that of 1.9 processors on average)

Question:
What is system availability here?

Performability improvement factor of this system (akin to RIF) relative to a fail-hard system that goes down when either processor fails:

PIF = $(2 - 2 \times 0.92) / (2 - 1.90) = 1.6$

Integrity and Safety

Integrity and safety are similar

Integrity is inward-looking: capacity to protect system resources (e.g., data)

Safety is outward-looking: consequences of incorrect actions to users

A high-integrity system is robust

Data is not corrupted by low-severity causes

Safety is distinct from reliability: a fail-safe system may not be very reliable in the traditional sense

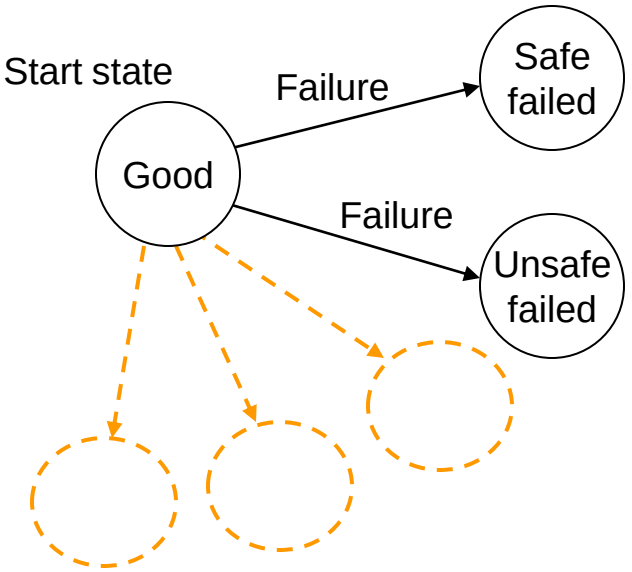
Integrity and Safety

Embedded Systems Laboratory

Risk: Prob. of being in "Unsafe Failed" state

There may be multiple unsafe states, each with a different consequence (cost)

Three-state fail-safe system



Simple analysis

Lump "Safe Failed" state with "Good" state; proceed as in reliability analysis

More detailed analysis

Even though "Safe Failed" state is more desirable than "Unsafe Failed", it is still not as desirable as the "Good" state; so keeping it separate makes sense

For example, if a repair transition is introduced between "Safe Failed" and "Good" states, we can tackle questions such as the expected outage of the system in safe mode, and thus its availability

Quantifying Safety

Embedded Systems Laboratory

Risk = Frequency × Magnitude

Consequence / Unit time Events / Unit time Consequence / Event

Risk = Probability × Severity

Magnitude or severity is measured in some suitable unit (say, dollars)

Privacy and Security

Privacy and security impairments are human-related

Accidental: operator carelessness, improper reaction to safety warnings

Malicious attacks: Hackers, viruses, and the like

Privacy is compromised when

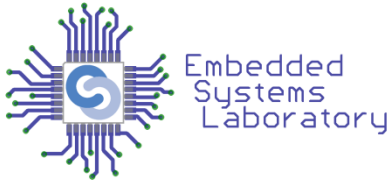
confidential or personal data are disclosed to unauthorized parties

Security is breached when

account information in a bank is improperly modified, say

Security is distinct from both reliability and safety: a system that automatically locks up when a security breach is detected may not be very reliable or safe in the traditional sense

Quantifying Security



In theory, security can be quantified in the same way as safety:

Risk = Frequency × Magnitude

Risk = Probability × Severity

But because security breaches are often not accidental, they are ill-suited to probabilistic treatment