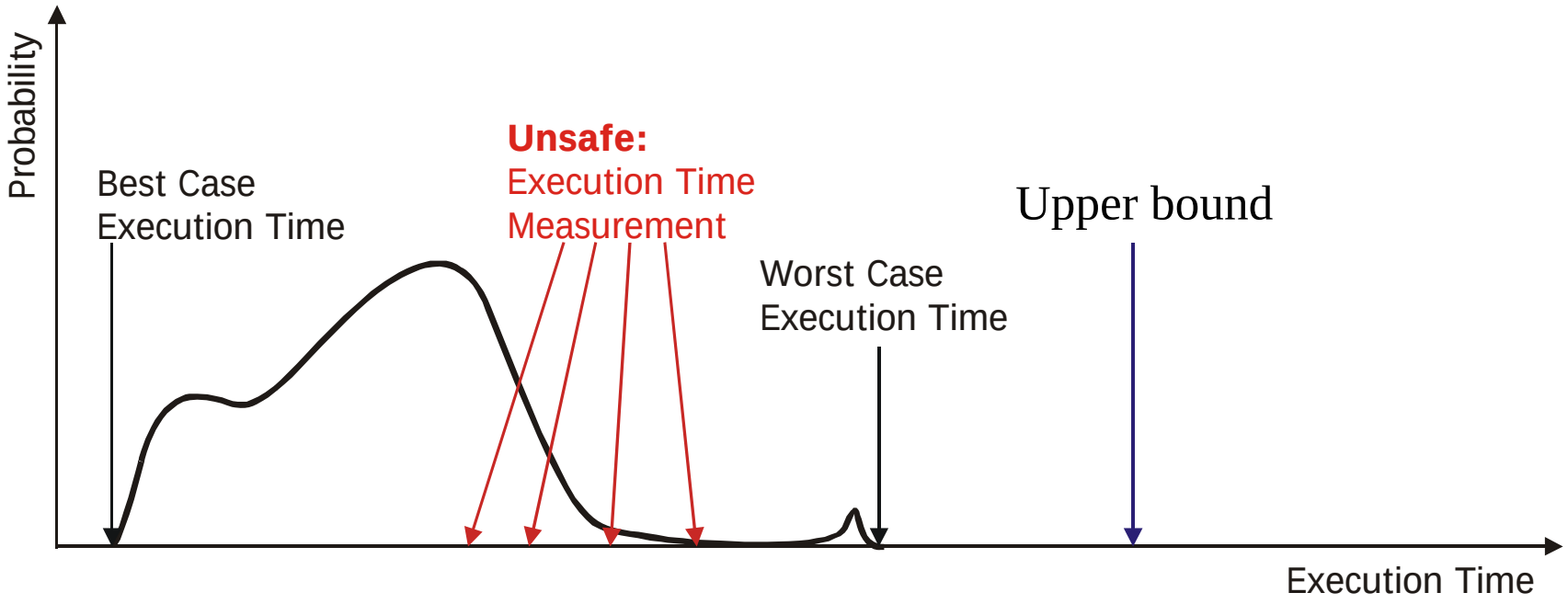
Embedded Systems 24

Measurement vs. Analysis

REVIEW



- typically huge variations in ET depending on input, cache effects,…
- cannot be covered within product development time
- rules of thumb add safety margins: pessimistic? optimistic?

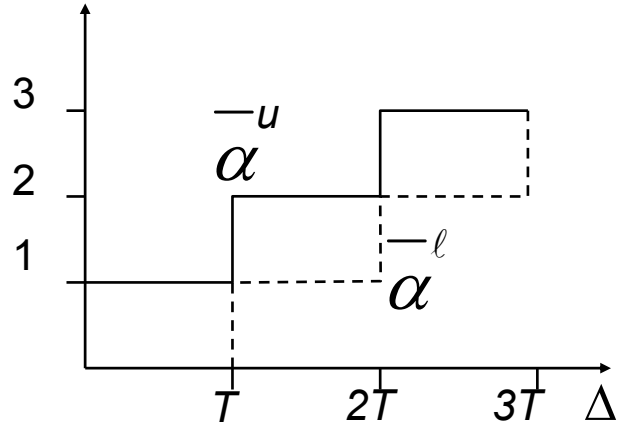
Real-Time Calculus: Arrival curves

REVIEW

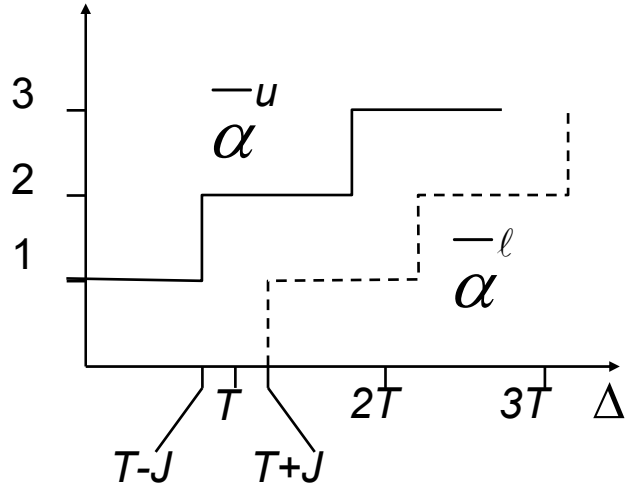
Arrival curves describe the maximum and minimum number of events arriving in some time interval Δ

Examples:

periodic event stream



periodic event stream with jitter

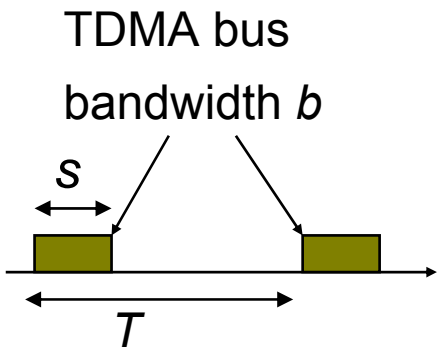


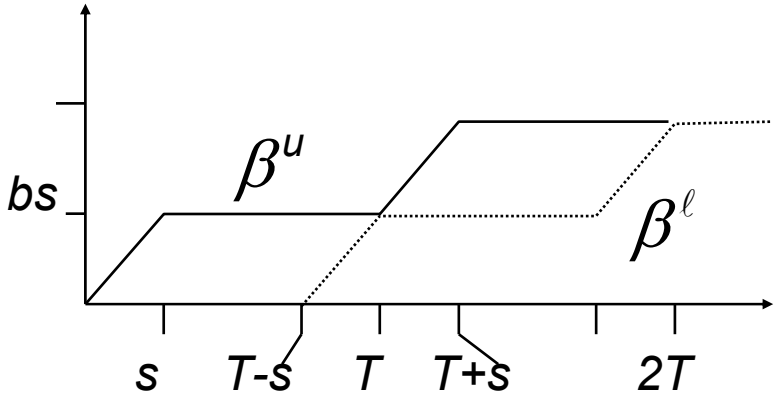
Service curves

REVIEW

Service curves β^u resp. β^ℓ describe the maximum and minimum service capacity available in some time interval Δ

Example:



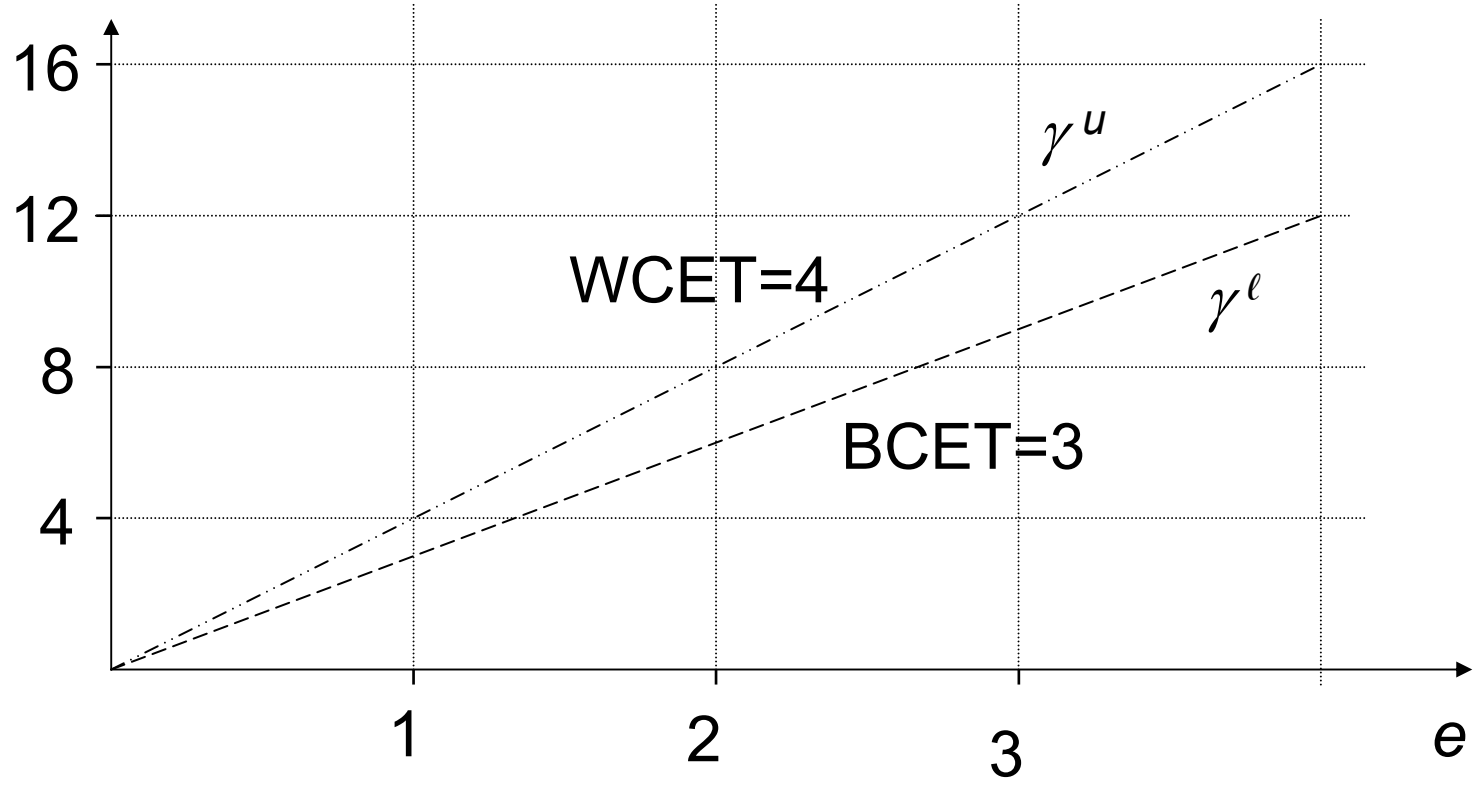


Workload characterization

REVIEW

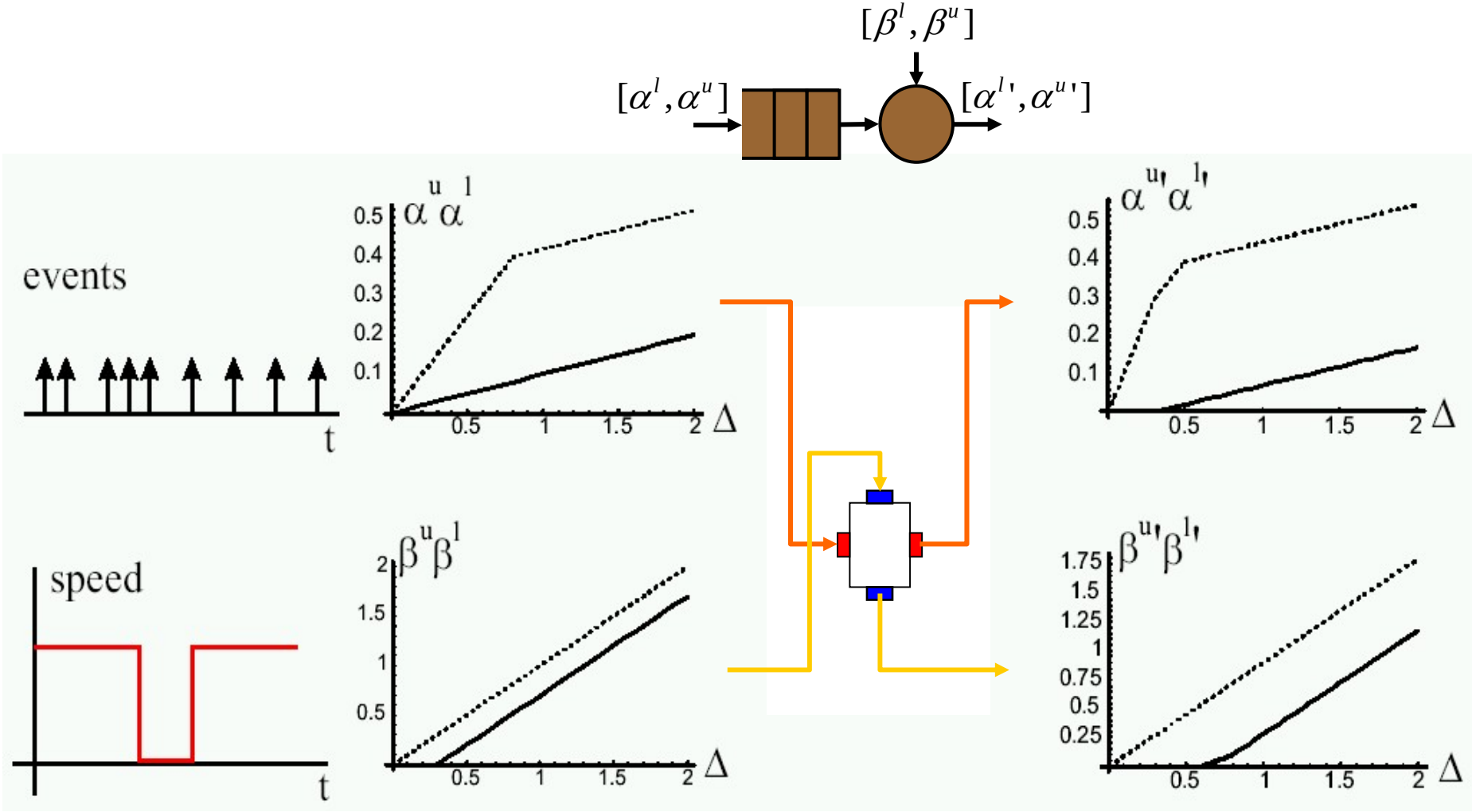
γ^u resp. γ^ℓ describe the maximum and minimum service capacity required as a function of the number *e* of events

Example



Transformation of Curves by Modules

REVIEW



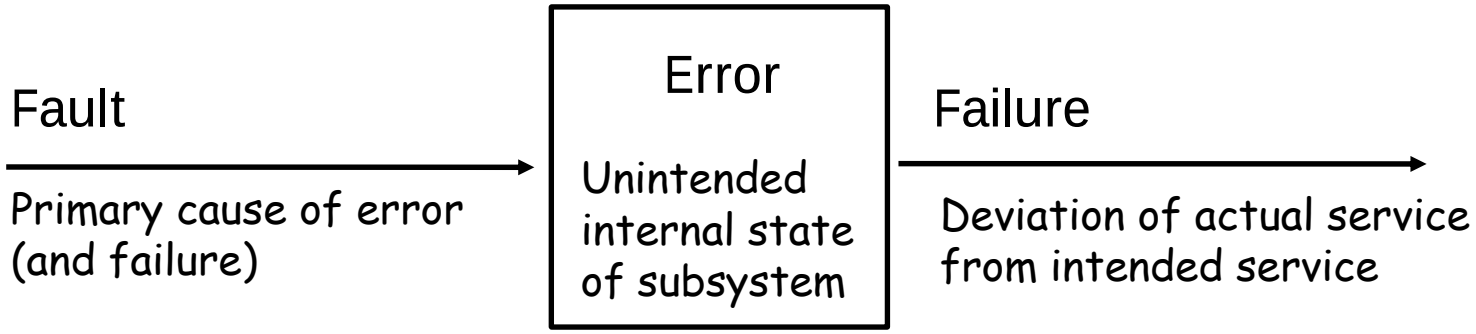
Safety vs. Reliability

- Safe means *sufficiently low probability of serious harm caused by the system:*
  - e.g. ISO 8402: „State in which risk of harm (to persons) or damage is limited to an acceptable level.“
- *Reliable* means *sufficiently high probability of delivering intended service.*
  - Reliability is the probability of the system delivering the service it was designed for throughout the horizon, given
    - a defined temporal horizon
    - the operational conditions

Faults, Errors & Failures

Standardized terminology: J. C. Laprie (ed.) 1992,

„Dependability: Basic Concepts and Terminology“



Example - landing gear in an airplane

- Landing gear sensor faulty: doesn’t report that gear is down
- Landing flaps and thrust-reverters are blocked by control software though plane is grounded
- Braking distance increases dramatically, plane may drive off runway

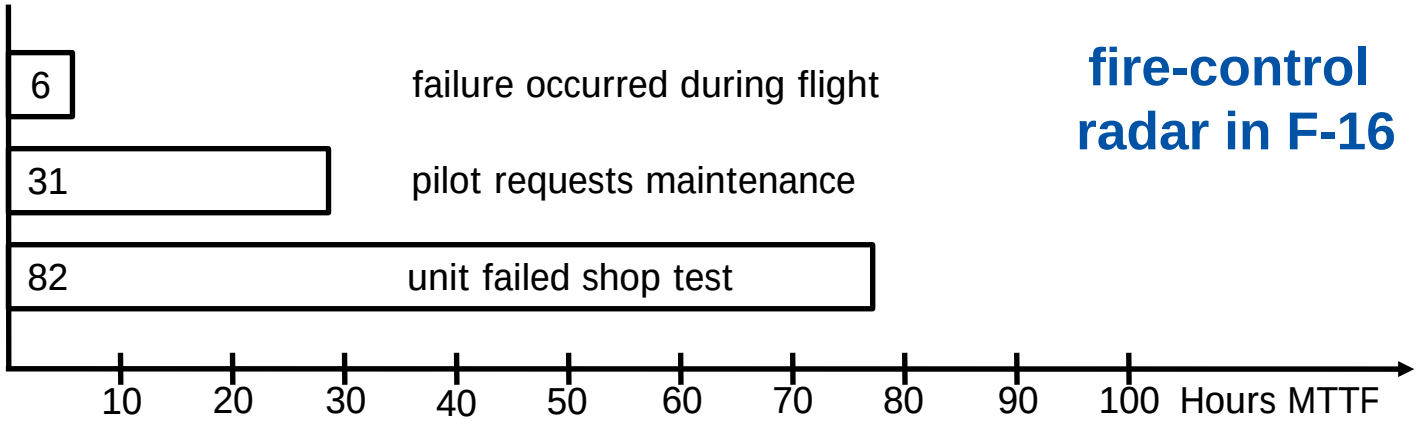
Dealing with Faults

- **Fault avoidance** aims at preventing the occurrence of faults: design reviews, testing, verification.

- **Fault tolerance** Is the ability of a system to continue to perform its tasks after the occurrence of faults
  - **Fault masking**: preventing faults from introducing errors
  - **Reconfiguration**: fault detection, location, containment and recovery

Types of faults

- A **permanent fault** remains in existence indefinitely if no corrective action is taken
- A **transient fault** disappears within a short period of time
- An **intermittent fault** may appear and disappear repeatedly.



fire-control radar in F-16

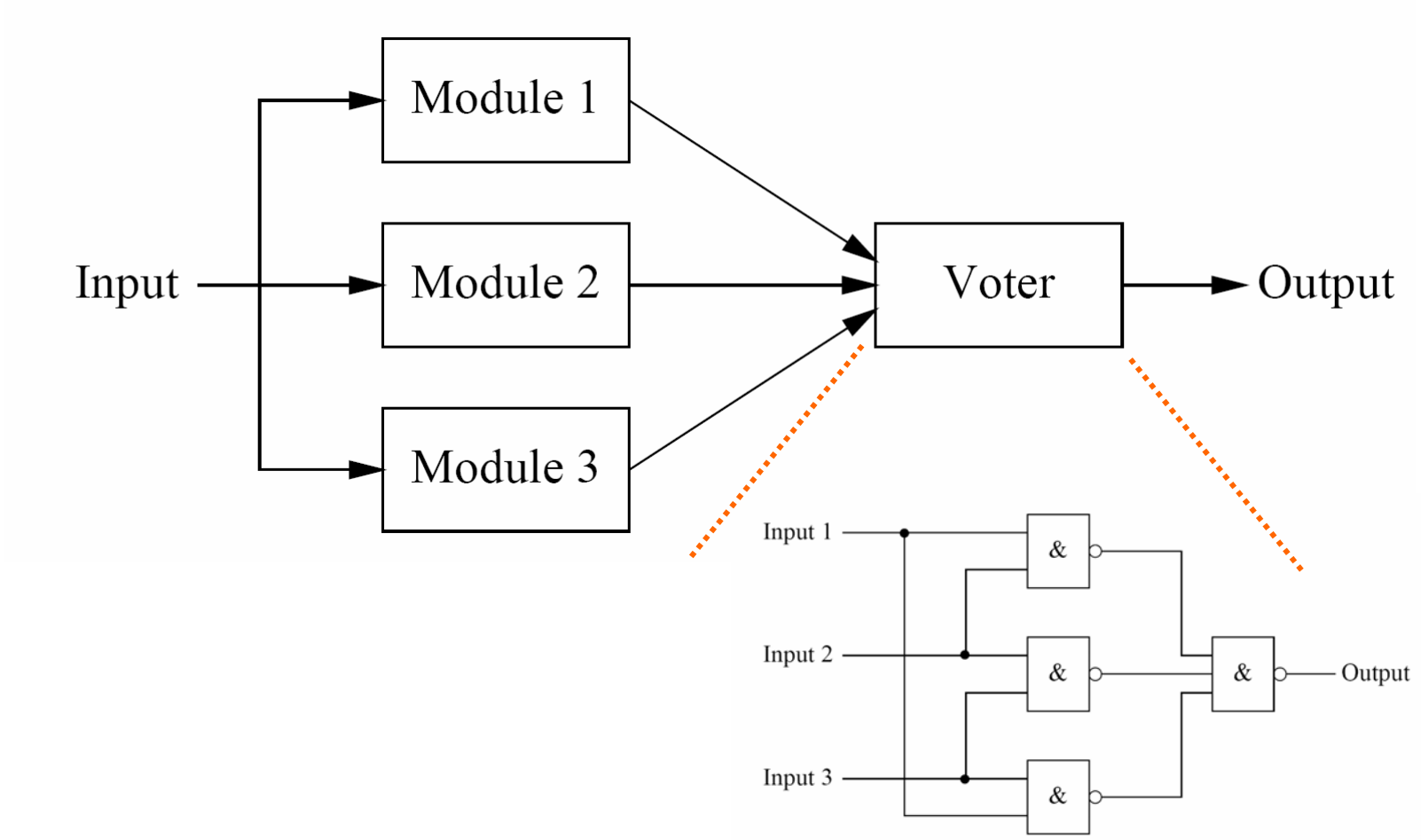
- Pilots noticed malfunctions every 6 flight hour
- Pilots requested maintenance every 31 hour
- Only 1/3 of the noticed malfunctions could be reproduced in the maintenance shop

Types of redundancy

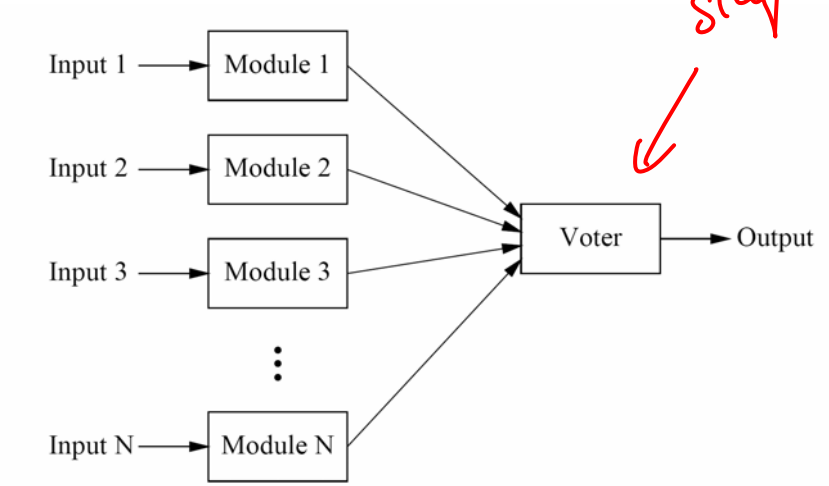
- **Hardware redundancy:** physical replication of hardware
- **Software redundancy:** different software versions of tasks, preferably written by different teams
- **Time redundancy:** multiple executions on the same hardware at different times
- **Information redundancy:** Coding data in such a way that a certain number of bit errors can be detected and/or corrected.

Static hardware redundancy

- Static redundancy based on voting.
- **Triple modular redundancy (TMR):**

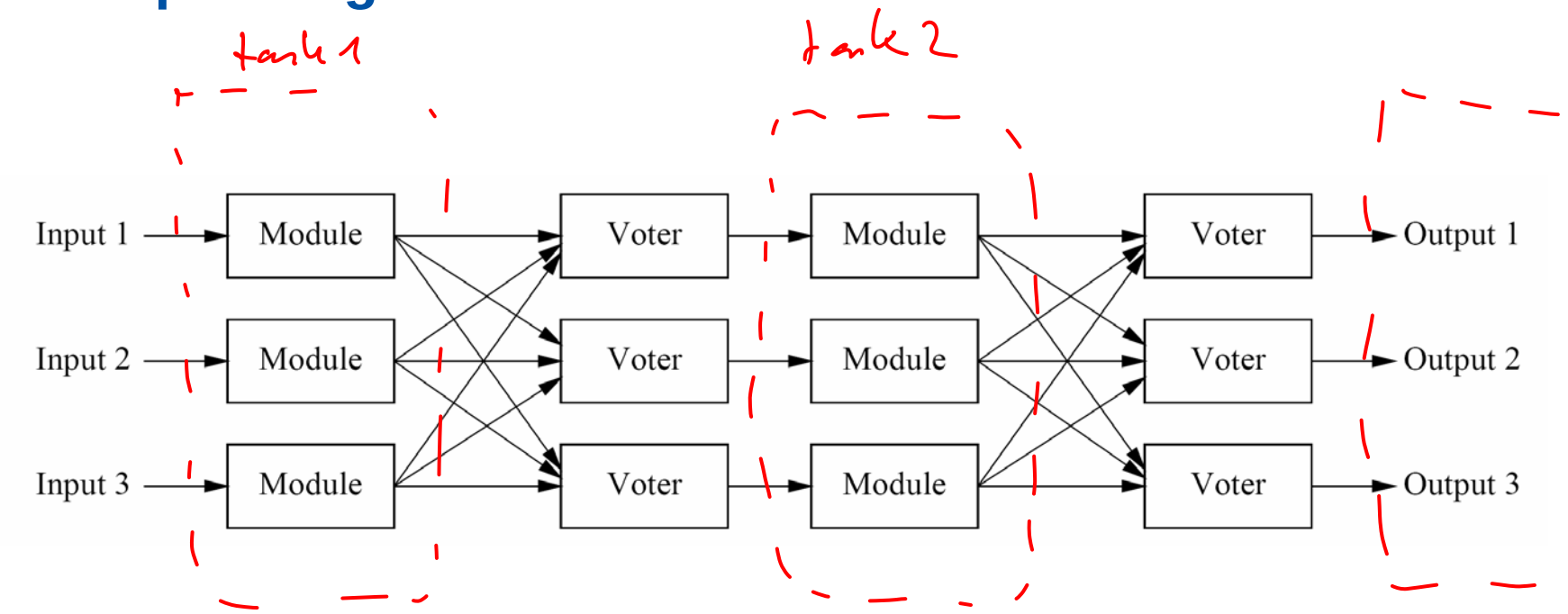


Static hardware redundancy: N-modular redundancy (NMR)

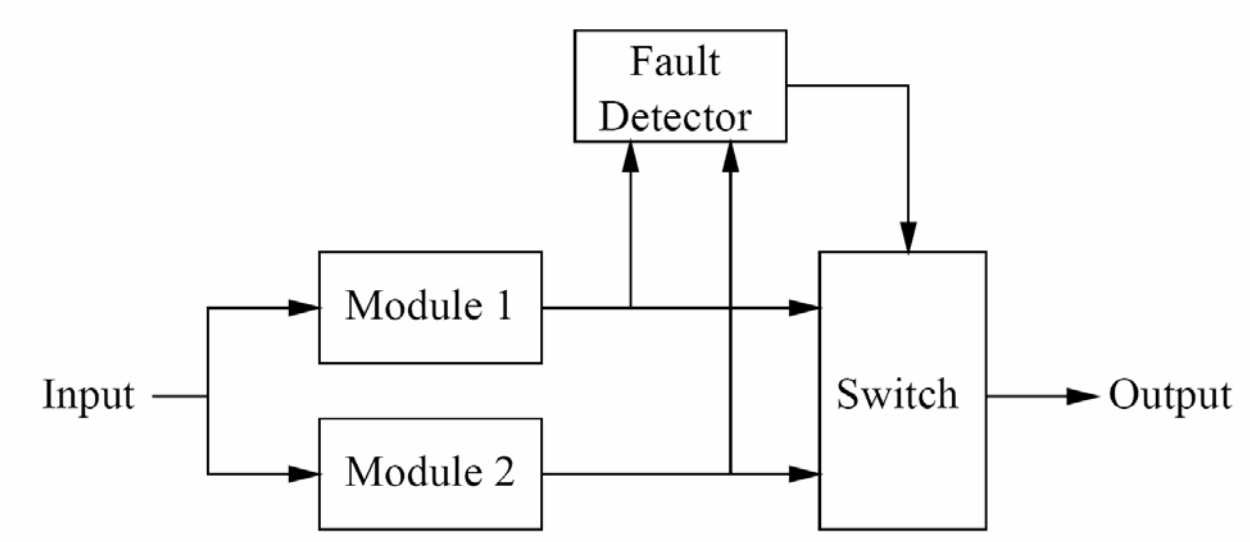


- System tolerates failure of (N-1)/2 modules
- Protects against random faults but not againts systematic faults
- Disadvantages: high cost, size, weight, energy. (typically: N≤4).

Static hardware redundancy: Multiple Stage TMR



Dynamic hardware redundancy: standby spare arrangement

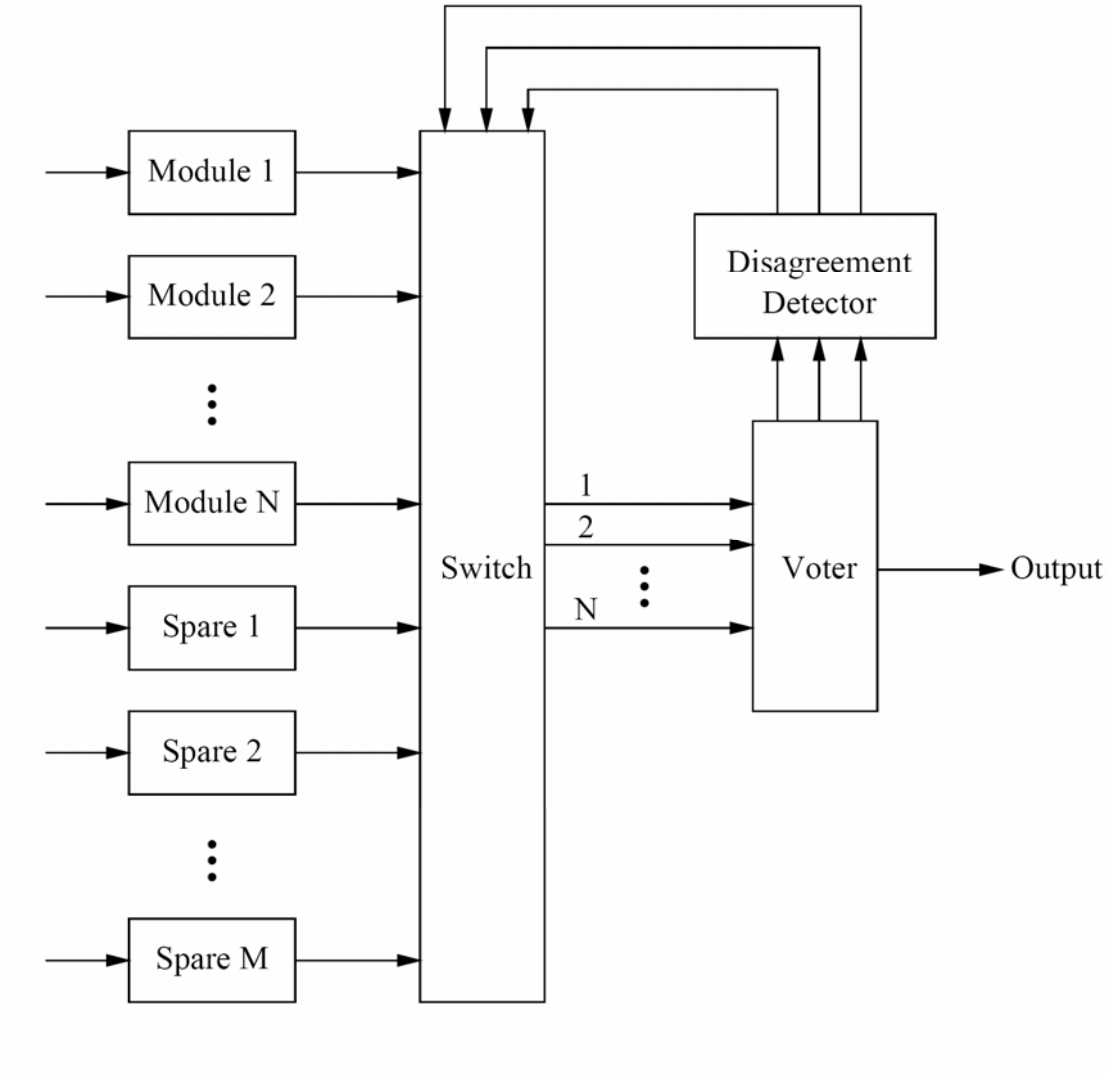


- Fault detection based on outputs (consistency check) not on voting
- Advantage: less redundant hardware
- Disadvantage: fault detection may take time ⇒ fault not masked

Standby spares

- **Hot standby:** spare is run continuously in parallel with active unit
  - Fast transfer of control
  - Increased power consumption
  - Same operating stress as active unit
- **Cold standby:** spare is unpowered until called into service
  - Reduces power consumption
  - Reduces wear and tear
  - More disruption at changeover

Hybrid redundancy:
N-modular redundancy with spares



Software fault tolerance

- **N-version programming** (≈ static redundancy)
  - Prepare N different versions
  - Run them in parallel or sequentially
  - Select result of majority at the end
- **Recovery blocks** (≈ dynamic redundancy)
  - Each job has a primary version and one or more alternatives
  - When primary version is completed, perform acceptance test
  - If acceptance test fails, run alternative version

Danger: common-mode failures

- Ambiguities in specification
- Choice of programming language, numerical algorithms,…
- Common background of software developers

Failure modes of subsystems

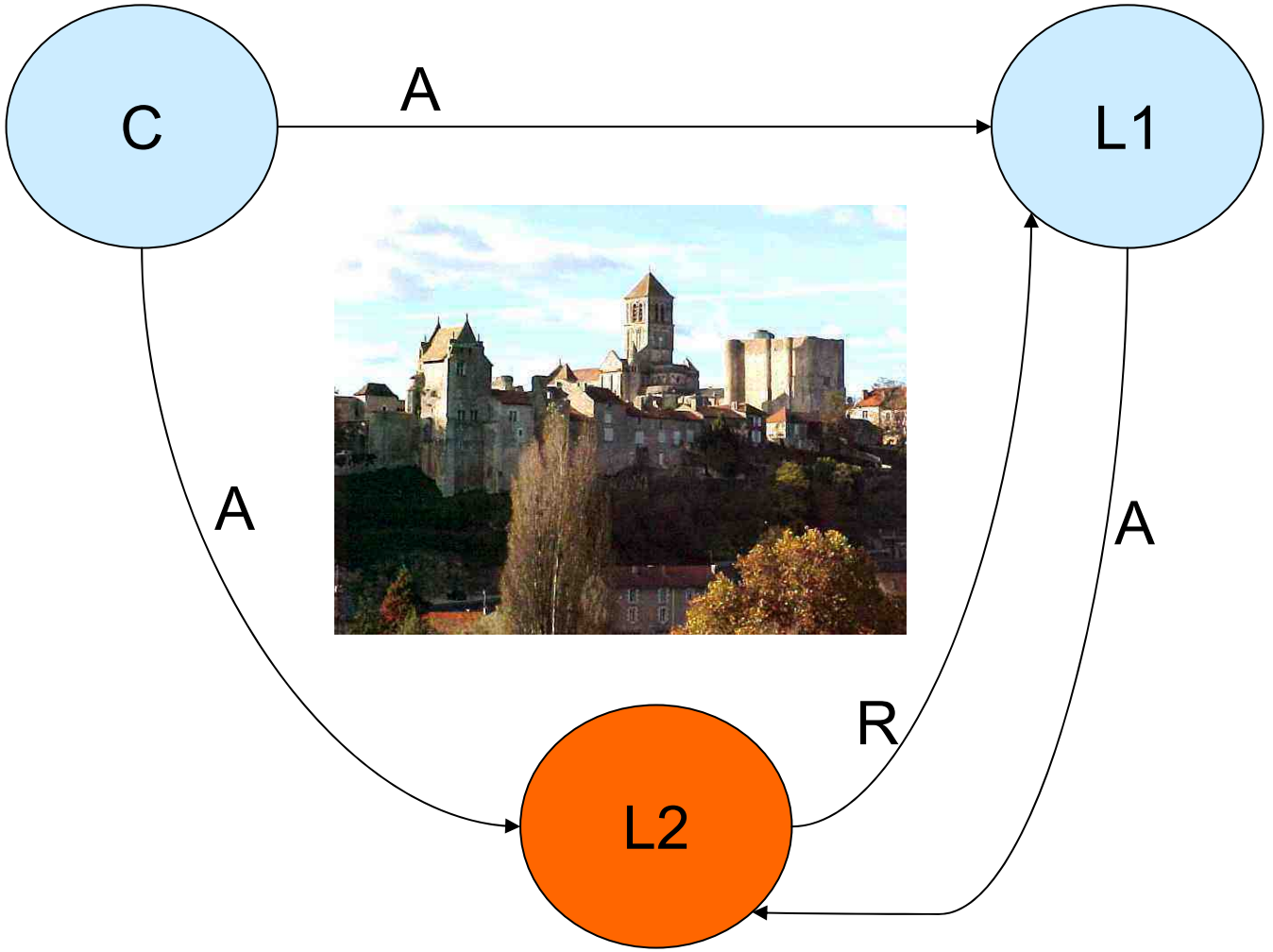
- Fail-silent failures
  - subsystem either produces correct results
    or produces (recognizable) incorrect results
    or remains quiet
  - **can be masked as long as at least one system survives**
- Consistent failures
  - If subsystem produces incorrect results all recipients receive same (incorrect) result
  - **can be masked iff the failing systems form a minority**
- Byzantine failures
  - subsystem reports different results to different dependent systems
  - **can be masked iff strictly less than a third of the systems fail**

Byzantine generals [Lamport/Shostak/Pease´82]

- Several divisions of the Byzantine army are camped outside an enemy city
- Each division is commanded by a general: there is one „commander“ and several „lieutenants“
- Each general may be a traitor
- Communication is reliable

- **Goal:** All loyal divisions must decide upon the **same** plan of action; if commander is loyal, loyal lieutenants should execute his order
- Basic idea: every lieutenant reports about the command received

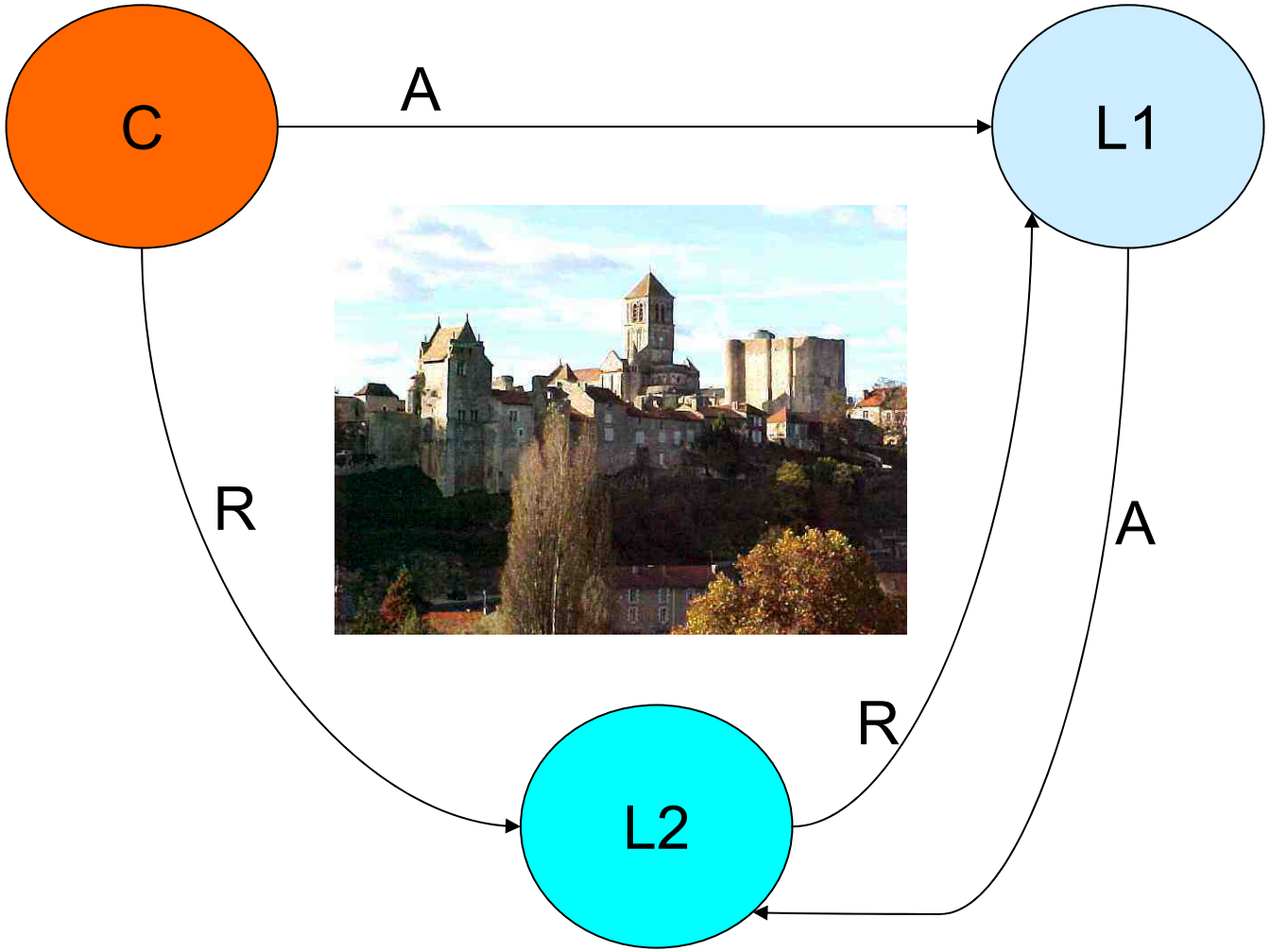
Decision: A

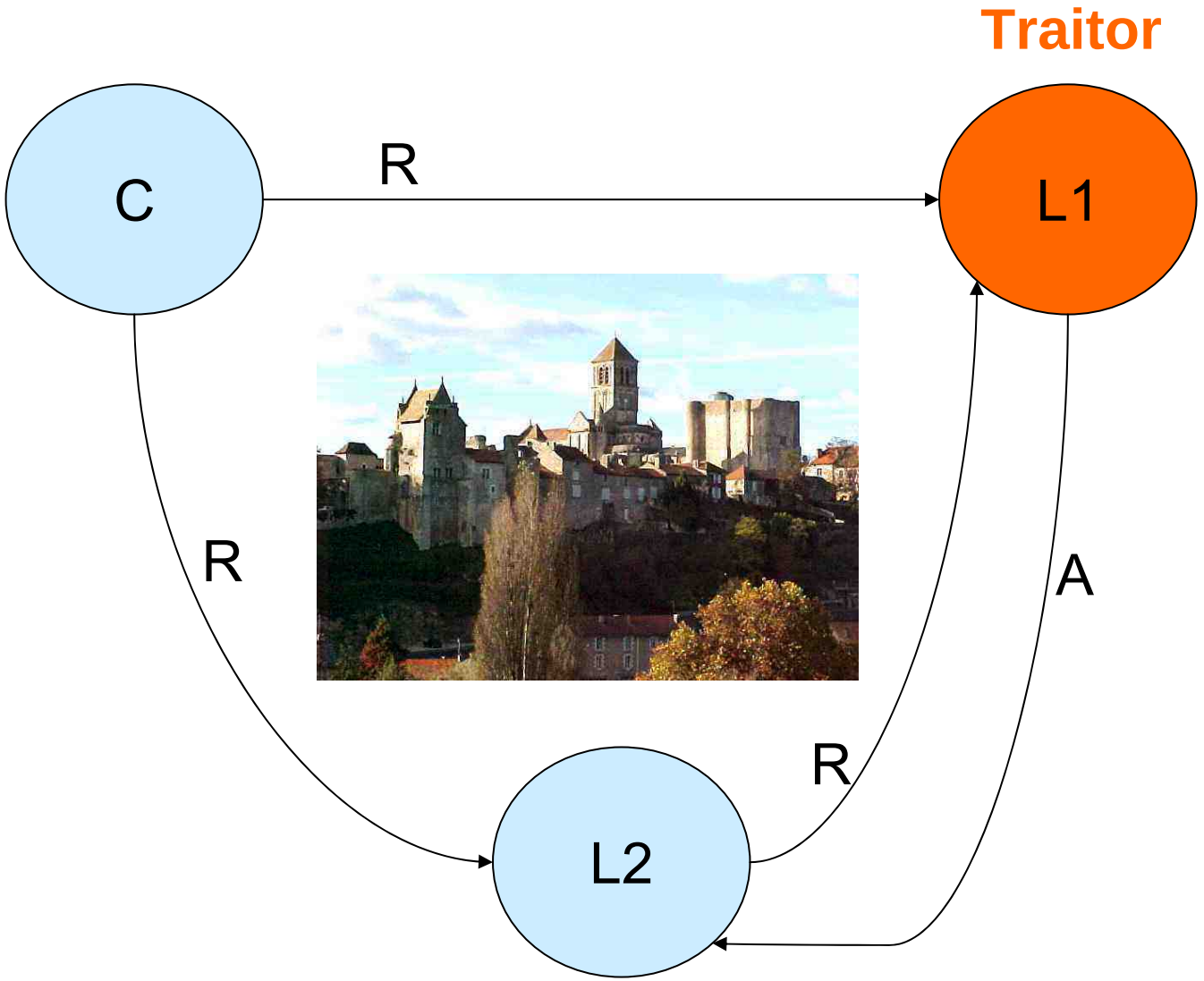


Traitor

Traitor

Decision: A

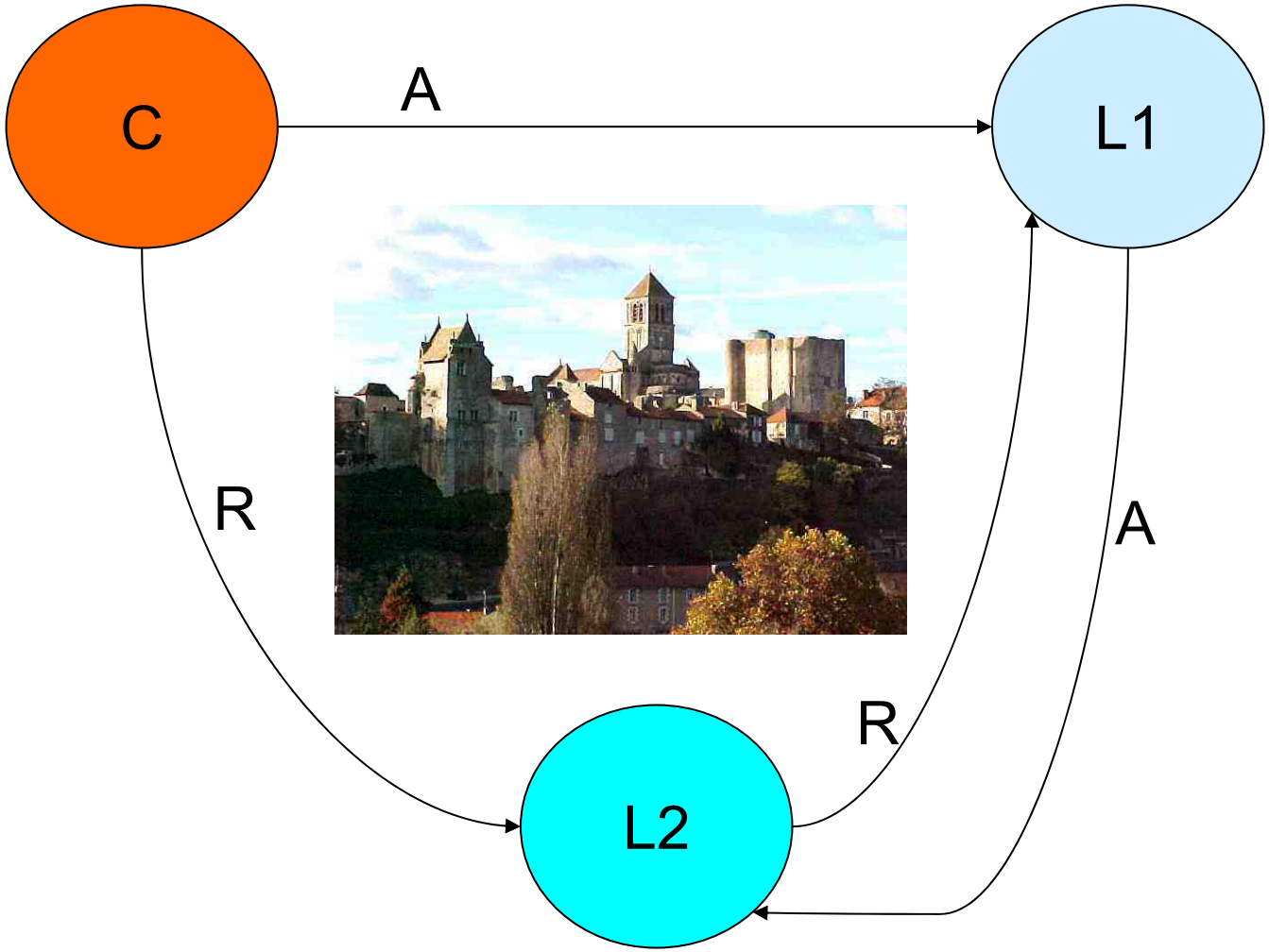




Decision: R

Traitor

Decision: A



Decision: R

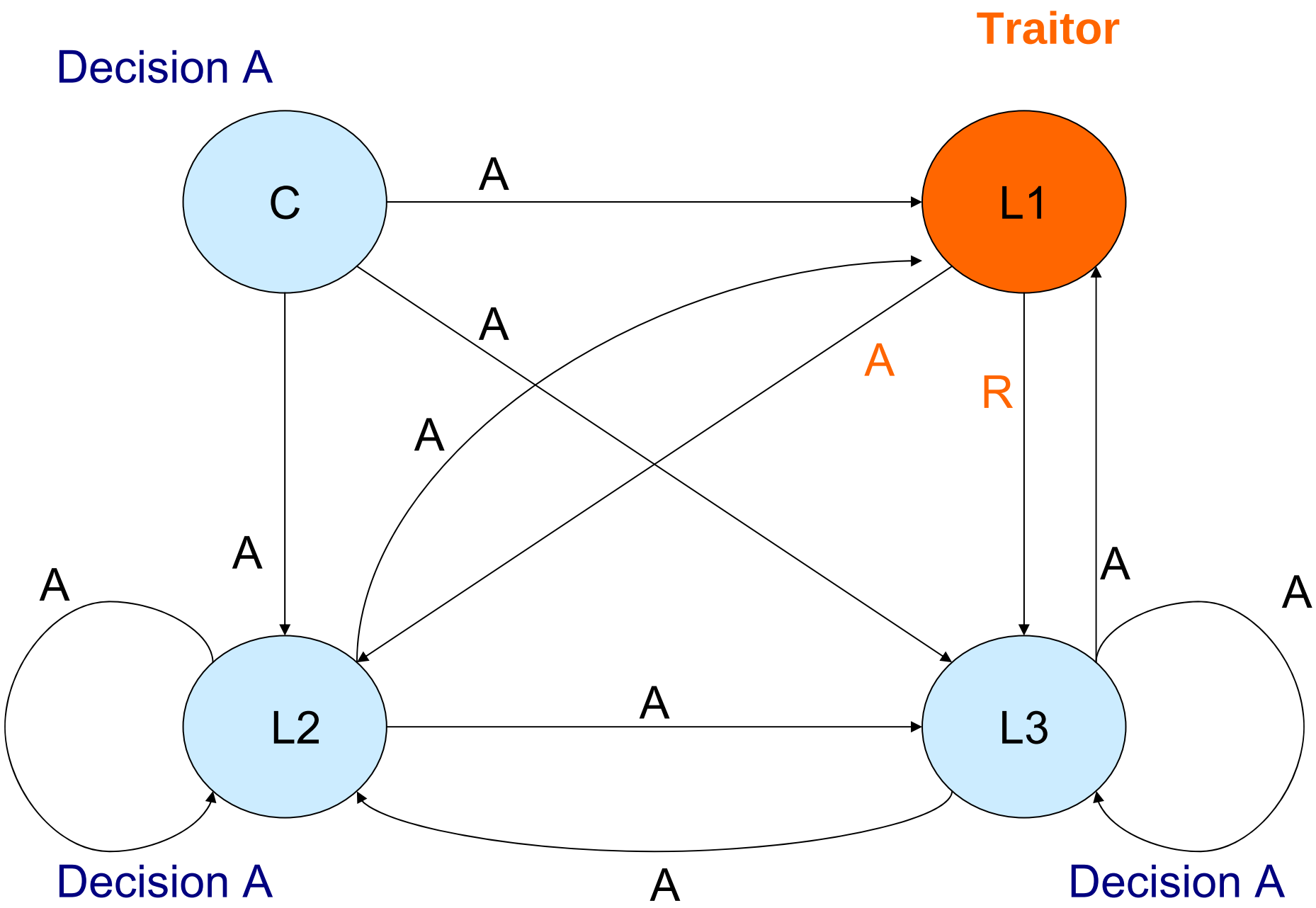
Solution

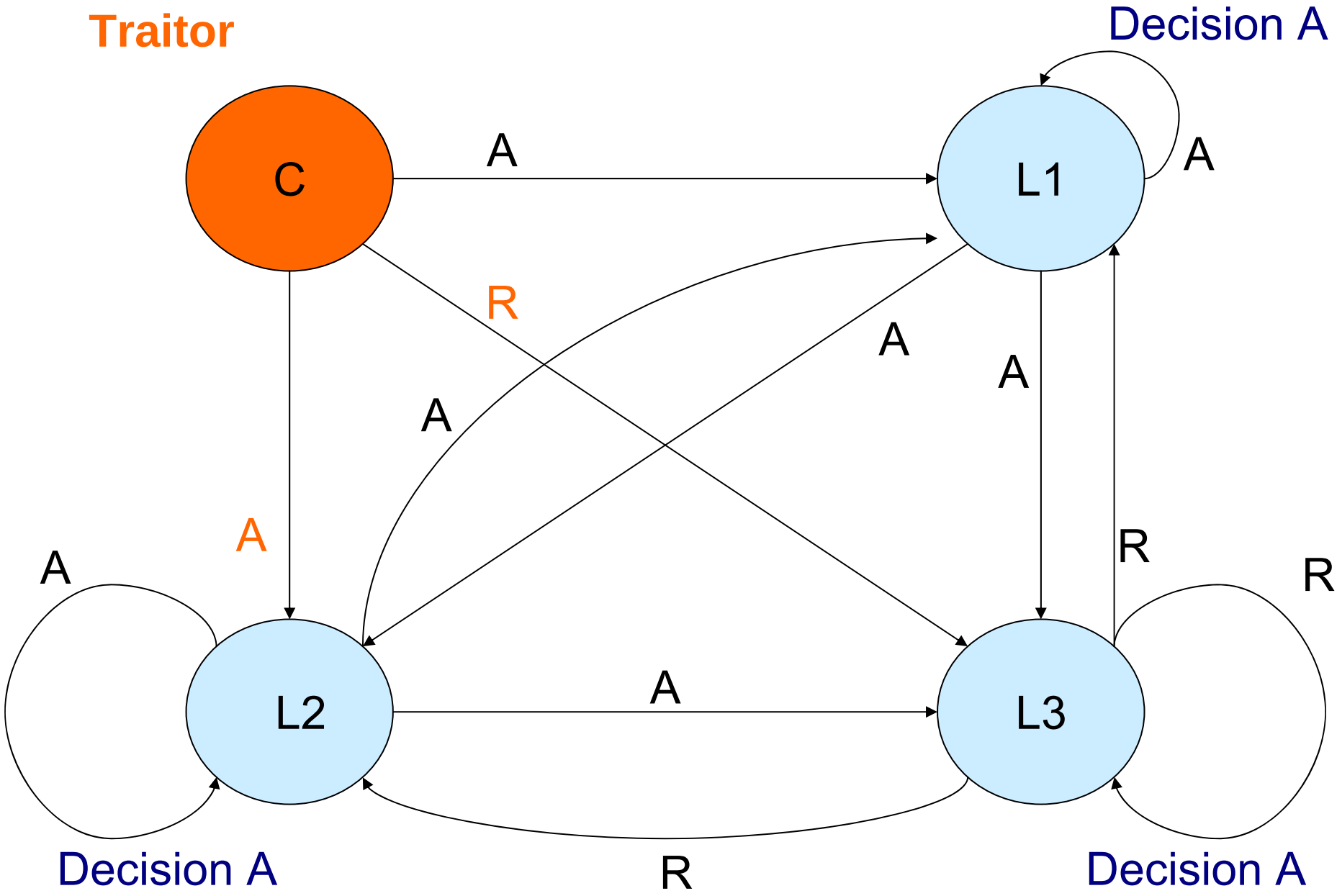
Algorithm A(0):

- Commander sends value (=order) to every lieutenant.

Algorithm A(*m*), *m*>0:

- Commander sends value to every lieutenant.
- Each lieutenant forwards value to all other lieutenants using algorithm A(*m*-1).
- Lieutenant *i* uses majority value of received values to determine result.


Traitor
Decision A
C
L1
L2
L3
A
A
A
A
A
A
A
A
A
A
A
R
Decision A
Decision A


Traitor
Decision A
C
L1
L2
L3
A
A
R
A
A
A
A
A
R
R
A
A
R
Decision A
Decision A

Lieutenants reach consensus (Case 1 traitor)

Case 1 : Commander is loyal
→ one lieutenant is the traitor
→ sets of forwarded messages differ by at most 1 value
→ same majority vote

Case 2 : Commander is traitor
→ lieutenants are loyal
→ sets of forwarded messages are identical
→ same majority vote at every lieutenant.

Lemma:

- Let there be more than $2k+m$ generals and at most k traitors. If the commander is loyal, then algorithm A(m) guarantees that all loyal lieutenants agree on the commander's order.

Induction on m:

- A(0): Commander is loyal ⇒ correct result received.

- $m-1 \rightarrow m$:
  - Loyal commander sends v to all lieutenants.
  - Each loyal lieutenant forwards v using A($m-1$) with $> 2k+m-1$ generals.
  - $2k+m-1 = 2k+(m-1)$
    ⇒ By ind. hyp. every loyal lieutenant receives v.

- Since there are at most k traitors
  → majority is loyal
- ⇒ majority vote is V.

Theorem

- Let there be more than 3*m* generals and at most *m* traitors. Then algorithm A(*m*) guarantees that the loyal lieutenants reach a consensus. If the commander is loyal, then the consensus is the commander‘s order.

Induction on m:

m=0 ⇒ A(0) ✓

m−1 → m:

Case: Commander is loyal
→ Apply lemma with k=m.

Case: Commander is traitor
- At most m−1 lieutenants are traitors
- There are ≥ 3m−1 lieutenants

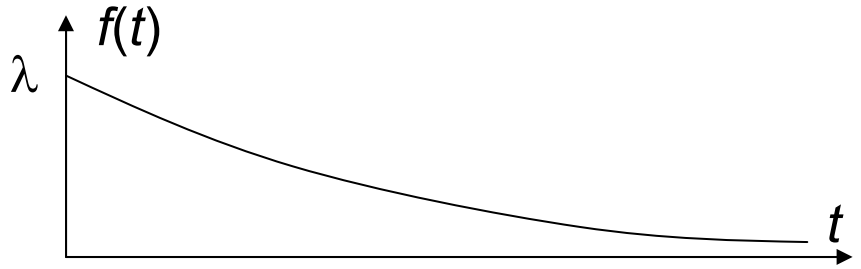
- $3n-1 > 3(n-1)$
- By ind. hyp. $A(n-1)$ ensures correct forwarding between any pair of loyal lieutenants and ensures among loyal recipients of msgs from traitors.
- $\Rightarrow$ The set of forwarded msgs is the same at each lieutenant
  $\Rightarrow$ Hence, the majority vote is the same at every lieutenant.

Reliability: f(t), F(t)

- Let T: time until first failure, T is a random variable
- Let $f(t)$ be the density function of T

Example: Exponential distribution

$f(t)=\lambda e^{-\lambda t}$

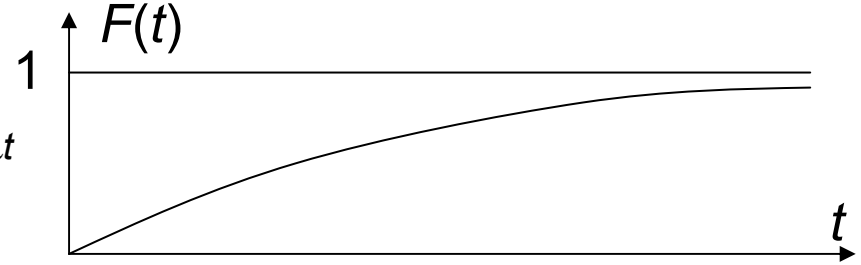


- $F(t)$ = probability of the system being faulty at time t:

$F(t) = \Pr(T \leq t)$ $\qquad F(t) = \int_0^t f(x)\,dx$

Example: Exponential distribution

$$F(t) = \int_0^t \lambda e^{-\lambda x} dx = -[e^{-\lambda x}]_0^t = 1 - e^{-\lambda t}$$



Reliability: *R*(*t*)

- **Reliability** *R*(*t*) = probability that the time until the first failure is larger than some time *t*:

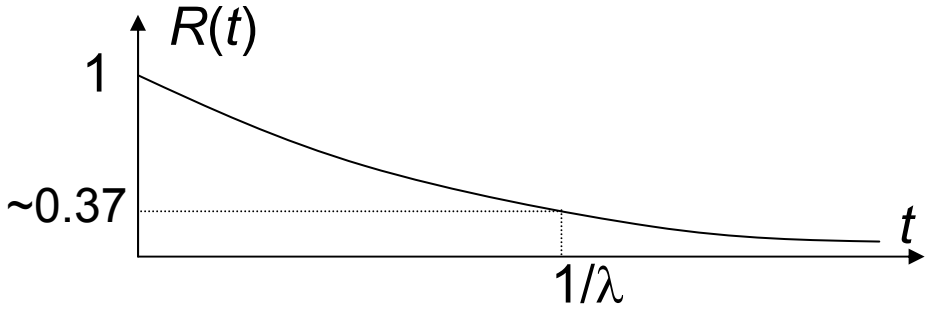
$R(t)=\Pr(T>t),\ t\geq 0 \qquad R(t)=\int_t^{\infty} f(x)dx$

$$F(t)+R(t)=\int_0^t f(x)dx+\int_t^{\infty} f(x)dx=1$$

$$R(t)=1-F(t)$$

Example: Exponential distribution

$R(t)=e^{-\lambda t};$



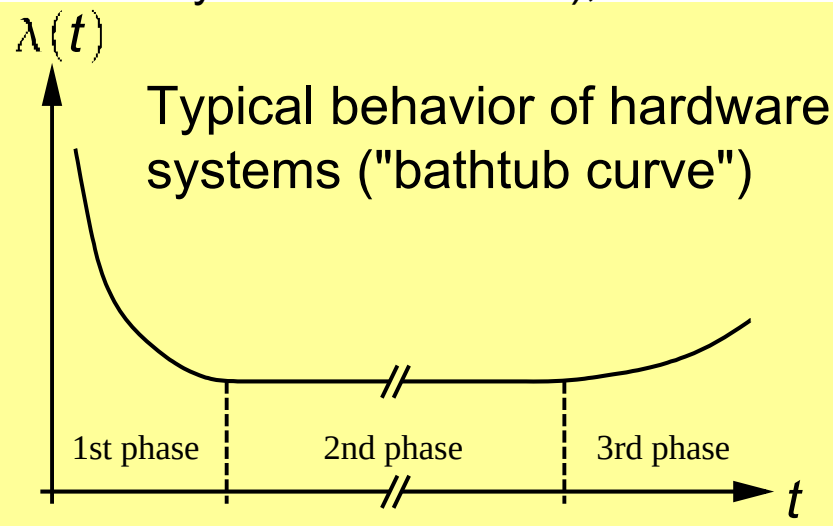
Failure rate

The failure rate at time *t* is the probability of the system failing between time *t* and time *t*+*Δt:*

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\Pr(t < T \le t + \Delta t \mid T > t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{F(t+\Delta t) - F(t)}{\Delta t R(t)} = \frac{f(t)}{R(t)}$$

Conditional probability ("provided that the system works at *t* ");

P(A|B)=P(AB)/P(B)



Typical behavior of hardware systems ("bathtub curve")

For exponential distribution:

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

FIT = expected number of failures in 10^9 hrs.

MTTF = *E*{*T* }, the *statistical mean* value of *T*

$$\mathrm{MTTF} = E\{T\} = \int_0^\infty t \cdot f(t)\, dt$$

According to the definition of the statistical mean value

Example: Exponential distribution

$$\mathrm{MTTF}_{exp} = \int_0^\infty t \cdot \lambda e^{-\lambda t} dt = -\left[t \cdot e^{-\lambda t}\right]_0^\infty + \int_0^\infty e^{-\lambda t} dt$$

$$\int u \cdot v' = u \cdot v - \int u' \cdot v$$

$$\mathrm{MTTF}_{exp} = -\frac{1}{\lambda}\left[e^{-\lambda t}\right]_0^\infty = -\frac{1}{\lambda}[0-1] = \frac{1}{\lambda}$$

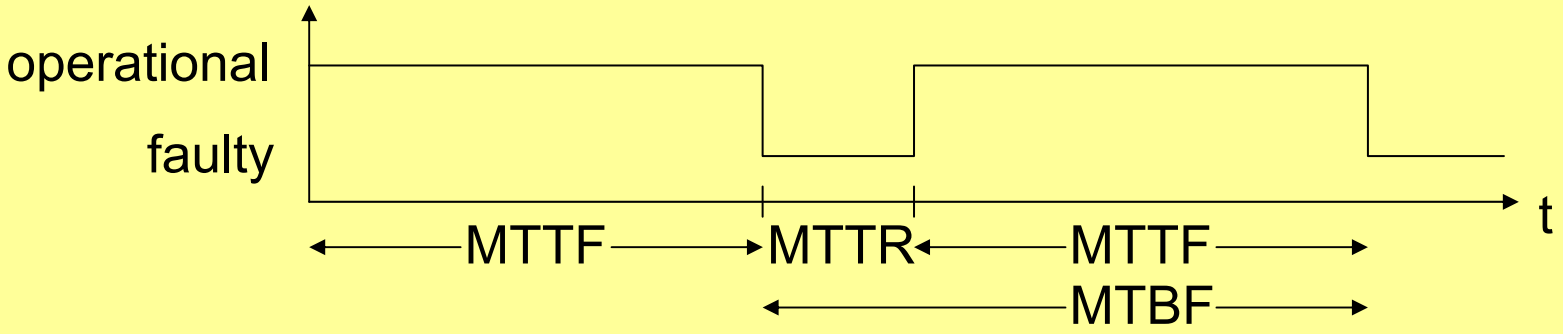
MTTF is the reciprocal value of failure rate.

MTTF, MTTR and MTBF

MTTR = mean time to repair
(average over repair times using distribution $M(d)$)
MTBF* = mean time between failures = MTTF + MTTR

$$\text{Availability } A = \lim_{t \to \infty} A(t) = \frac{\text{MTTF}}{\text{MTBF}}$$

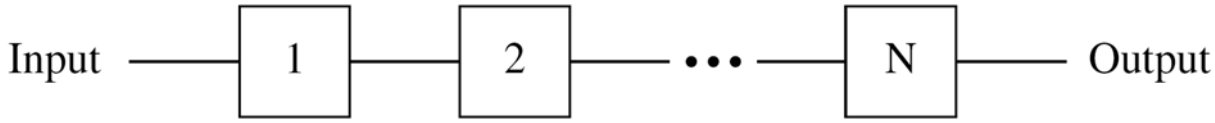
- Ignoring the statistical nature of faults …



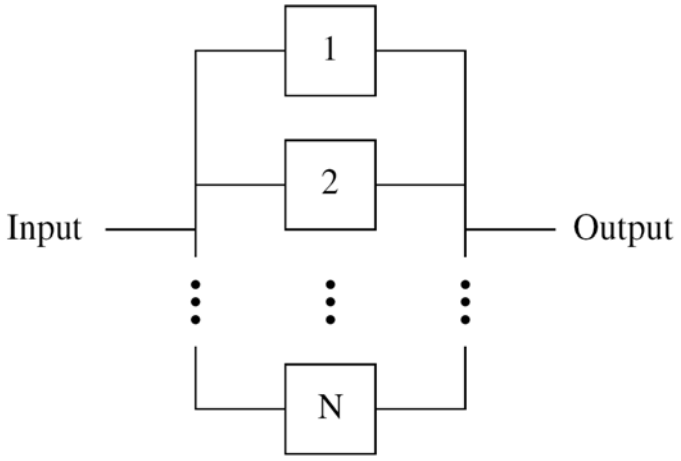
* Mixed up with MTTF, if starting in operational state is implicitly assumed

Reliability block analysis

- **Goal:** compute reliability of a system from the reliability of its components
- **Serial composition**



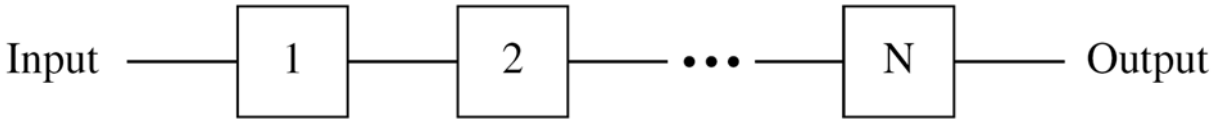
- **Parallel composition**



Inductive computation of reliability

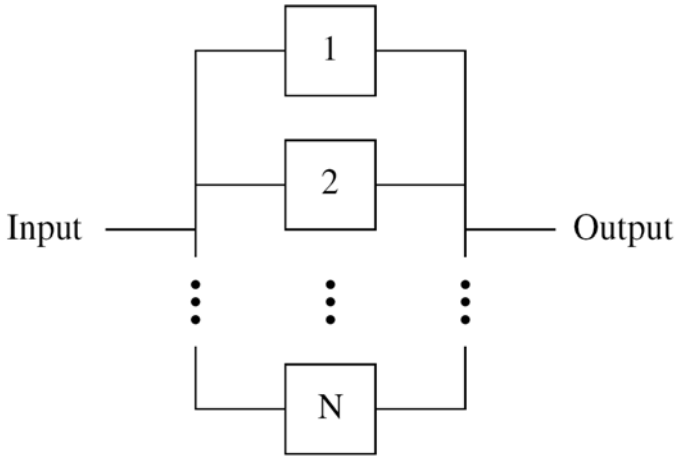
- **Assumption:** failures of the individual components are independent
- **Serial composition**

$$\prod_{i=1}^{N} R_i(t)$$

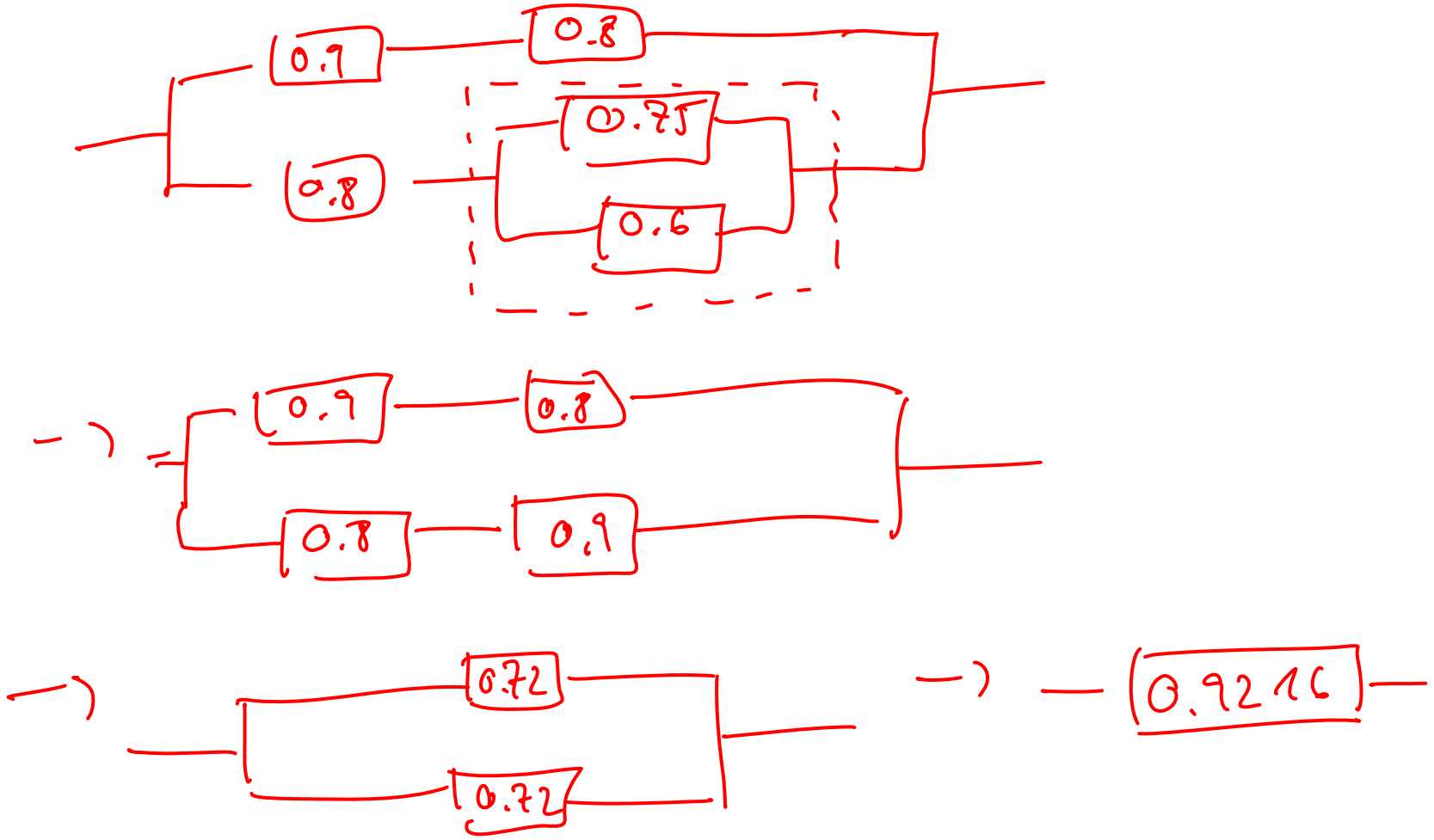


- **Parallel composition**

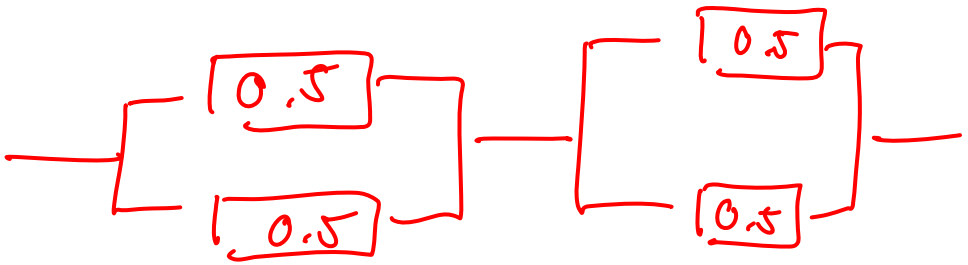
$$1 - \prod^{N} (1 - R_i(t))$$

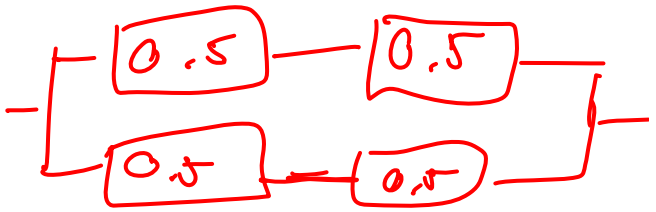


Example



Example





$R = (1 - (1 - \frac{1}{2})(1 - \frac{1}{2}))^2$

$= \frac{9}{16}$

$R = 1 - (1 - (\frac{1}{2})^2)^2$

$= 1 - \frac{7}{16} = \frac{7}{16}$

✓

"Redundancy more efficient at component level"

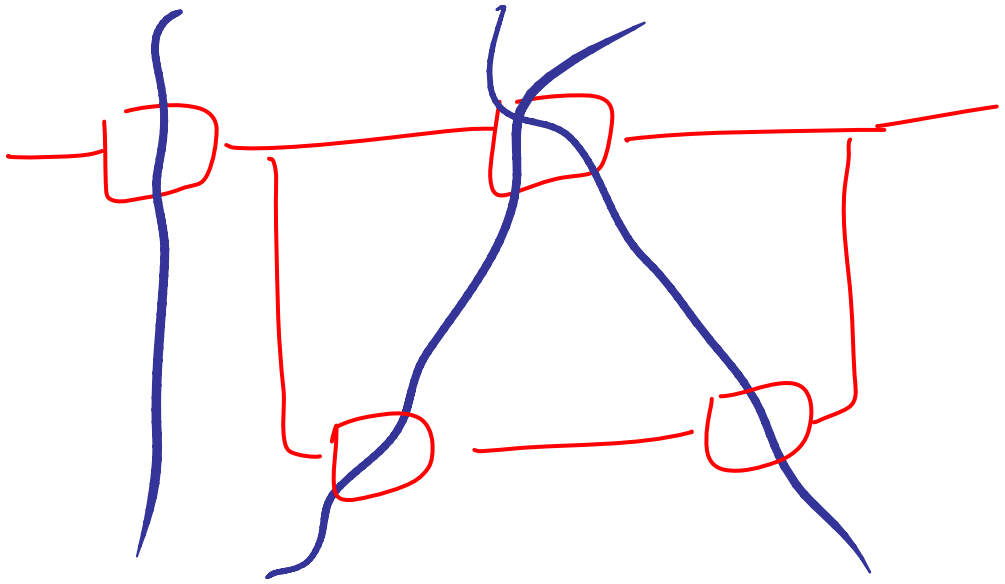
Approximation: Minimal Cuts

- A **minimal cut** is a minimal set of components such that their simultaneous failure causes a system failure

- $$1 - \sum_{j \in MinimalCuts} \prod_{i \in j} [1 - R_i(t)]$$

  is a lower bound for the reliability R(t) of the full system.

- Minimal cuts with a single component are called *single point failures*.



Minimal cuts