System Reliability Analysis

CS6423

Scalable Systems

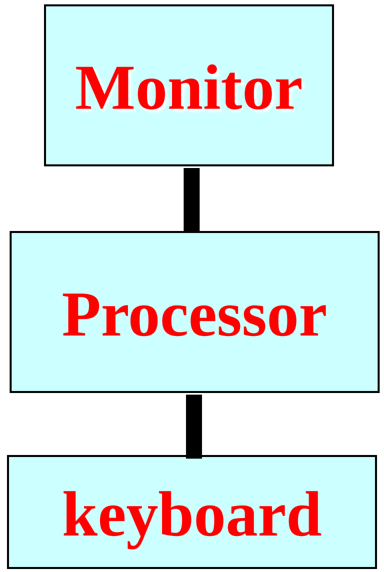
Topics

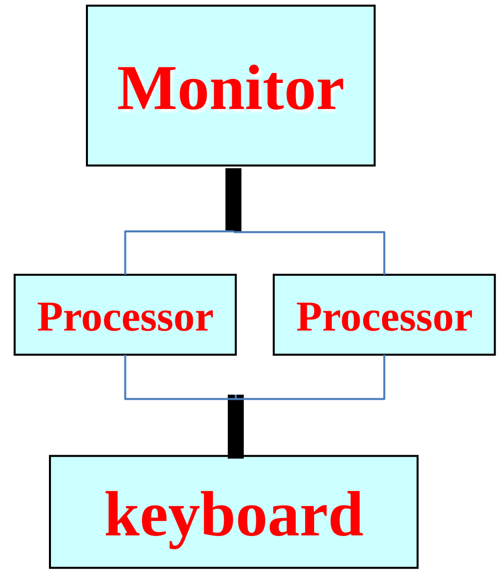
- Combinatorial Models for reliability
- Topology-based (structured) methods for
  - Series Systems
  - Parallel Systems
- Reliability analysis for arbitrary networks

Cork Complex Systems Lab

Computing System Reliability

- Depends on System Topology





- Assume each component fails randomly
  - How do we compute system reliability?

Combinatorial Approach (series topology)

If a system consisting of *n* components, and every component is either working or failed, then we can simply enumerate all the possible combinations and calculate the probability for each combination.

a	b	c	System	Prob.
w	w	w		p^3
w	w	f		$p^2(1-p)$
w	f	w		$p^2(1-p)$
f	w	w		$p^2(1-p)$
w	f	f		$p\,(1-p)^2$
f	f	w		$p\,(1-p)^2$
f	w	f		$p\,(1-p)^2$
f	f	f		$(1-p)^3$

Cork Complex Systems Lab

Combinatorial Method

- Use probabilistic techniques to enumerate the different ways in which a system can remain operational
- The reliability of a system is derived in terms of the reliabilities of the individual components of the system (thus the term combinatorial)

Complexity Concerns

- How many possible combinations of the status of these *n* components?
- What can be done to manage the complexity?
  - During model construction:
    - Need a more intelligent way to describe the system's failure behavior
    - Series and parallel RBD (Reliability Block Diagram) approach
  - During model solution:
    - Need more efficient approach than counting individual probabilities

Cork Complex Systems Lab

“Structured” Combinatorial Approach

- **Reliability block diagrams**
  - Integrate certain probability events into a module, which contains the info:
    - A probability of failure
    - A failure rate
    - A distribution of time to failure
    - Steady-state and instantaneous unavailability
  - Organize the modules in a “structured” way, according to the effects of each module’s failure
- **Statistical independence Assumption**
  - Failures independence
  - Repairs independence

Cork Complex Systems Lab

"Structured" Combinatorial models

- **Reliability block diagrams**, Fault trees and Reliability graphs
  - Integrate certain probability events into a module
  - Organize the modules in a "structured" way, according to the effects of each module's failure
  - Commonly used in reliability, availability, or safety assessment
  - These model types are similar in that they **capture conditions that make a system fail** in terms of the structural relationships between the system components.

Cork Complex Systems Lab

RBD Features

- Easy to use
- Assuming **statistical independence**
  - Failures independence
  - Repairs independence
- Each component can have attached to it
  - A probability of failure
  - A failure rate
  - A distribution of time to failure
  - Steady-state and instantaneous unavailability

RBD Features

- Easy specification,
- Fast computation
  - Relatively good algorithms are available for solving such models so that 100 component systems can be handled computationally
    - consider the case where you need to handle 2^{100} probability events

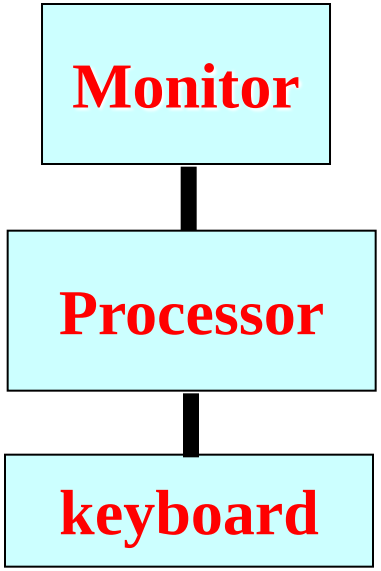
Example: Series System

- No redundancy
- Each component is needed to make the system work
- If any one of the components fails, the system fails
- Example:

Cork Complex Systems Lab

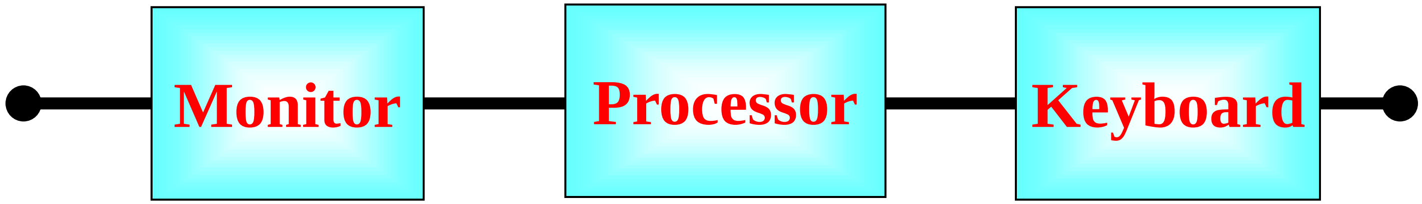
RDB Example for a Series System

- System Block Diagram for Example



Reliability Block Diagram Model & Reliability Calculation

RBD for Example



Let $\lambda 1$ be the failure rate for Monitor

Assume *exponential distribution* for the failures, then

$R_{monitor}(t) = e^{-\lambda 1 \cdot t}$

Similarly, $R_{processor}(t) = e^{-\lambda 2 \cdot t}$ and $R_{keyboardv}(t) = e^{-\lambda 3 \cdot t}$

$R_{system}(t) = R_{monitor}(t) \cdot R_{processor}(t) \cdot R_{keyboard}(t)$

$= e^{-\lambda 1 \cdot t} \cdot e^{-\lambda 2 \cdot t} \cdot e^{-\lambda 3 \cdot t} = e^{-(\lambda 1 \cdot t + \lambda 2 \cdot t + \lambda 3 \cdot t)} = e^{-(\lambda 1 + \lambda 2 + \lambda 3) \cdot t}$

When exponential failure distribution is assumed, the failure rate of a series system is the sum of individual components' failure rates

SS-Availability Calculation

Let λ1, λ2, λ3 be the failure rates and μ1, μ2, μ3 be the repair rates for the monitor, processor and keyboard. Then

$$A_{SS\text{-}Monitor} = \frac{\mu 1}{\lambda 1 + \mu 1}$$

$$A_{SS\text{-}processor} = \frac{\mu 2}{\lambda 2 + \mu 2}$$

$$A_{SS\text{-}keyboard} = \frac{\mu 3}{\lambda 3 + \mu 3}$$

$$A_{SS\text{-}system\text{-}series} = \frac{\mu 1}{\lambda 1 + \mu 1} \bullet \frac{\mu 2}{\lambda 2 + \mu 2} \bullet \frac{\mu 3}{\lambda 3 + \mu 3}$$

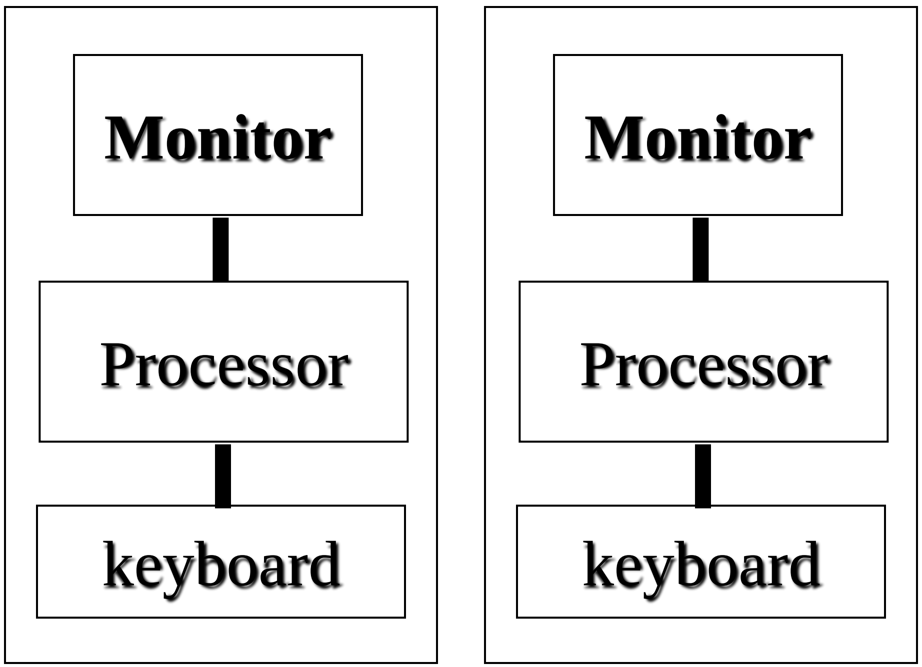
Parallel Systems

- A basic parallel system: only one of the ***N*** identical components is required for the system to function
- Example :

Cork Complex Systems Lab

Example : Basic Parallel System

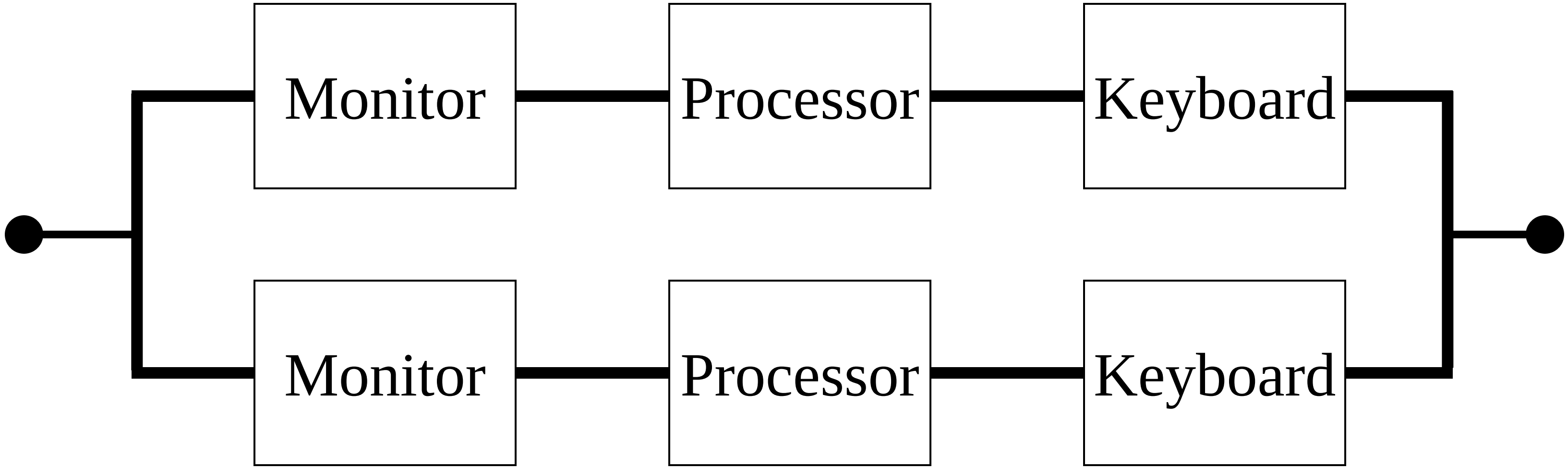
System Block Diagram



The purpose here is to show the parallel RBD and the corresponding reliability/availability calculations.

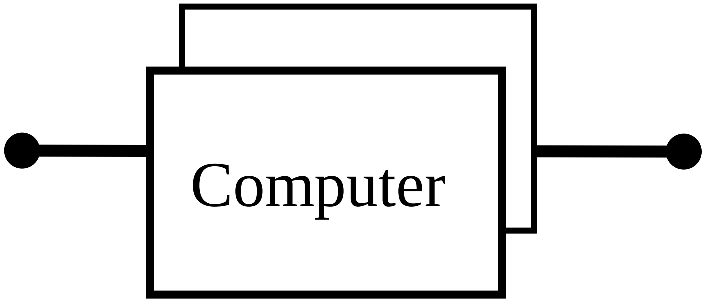
RDB example: Parallel System

- Reliability Block Diagram



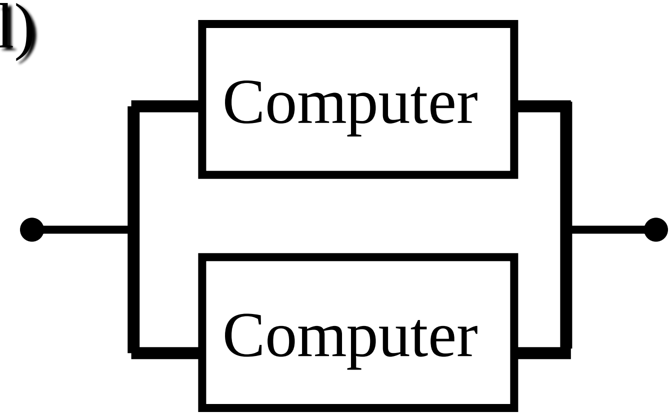
RDB using Hierarchical Composition/Decomposition

The Highest level (overall system level)



1 of 2

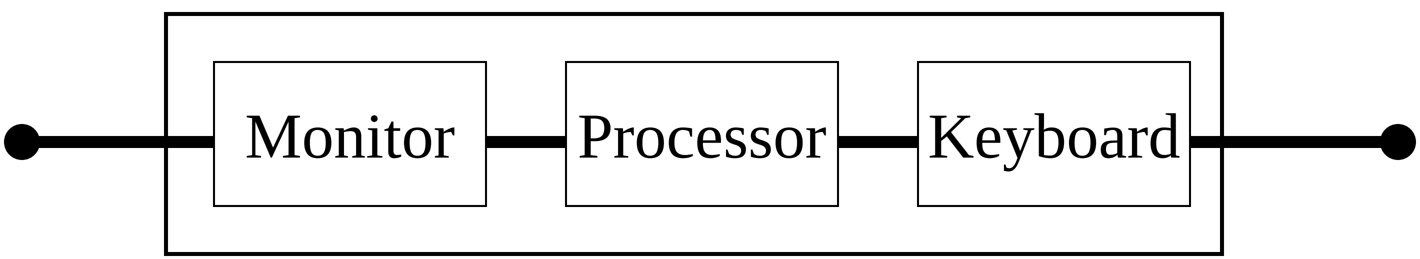
or



1 of 2

Usually indicate two different components

On the "Computer" level



Reliability Calculation

- The "Unreliability" of the parallel system can be computed as the probability that all ***N*** components fail.
- Assume all ***N*** components are having the same failure rate λ, and the probability that a component is failed at time ***t*** is $P_{fail}(t)$
- $R_{parallel}(t) = 1 - \prod_{i=1 \text{ to } N} P_{fail}(t)$
- If exponential distribution is used for $P_{fail}(t)$, derive the formula for $R_{parallel}(t)$

Independence Assumption

- Where in the above equation that the independence assumption is made?
- Just to remind you…

- Failure/Repair Dependencies are often assumed
- RBD usually does not handle the dependency such as
  - Event-dependent failure
  - Shared repair

Availability Calculation

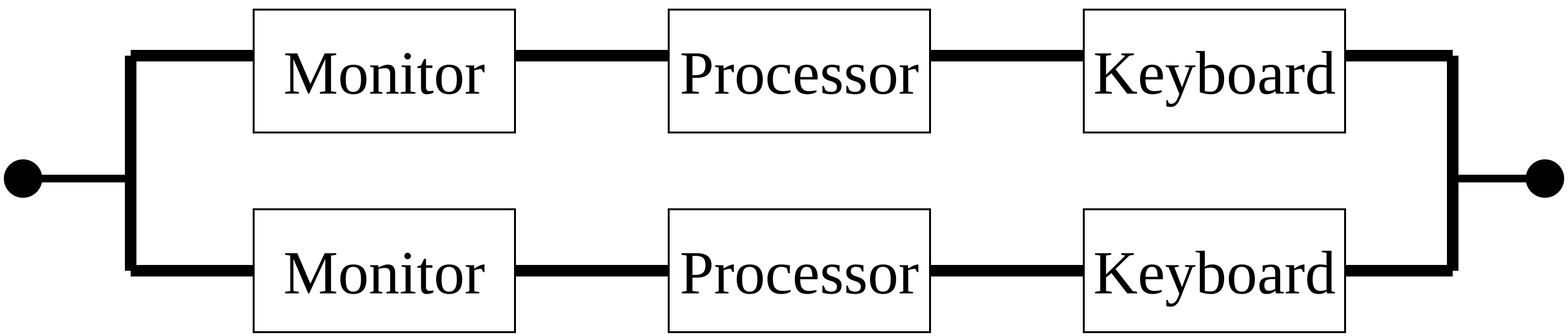
$$A_{SS\text{-}Monitor} = \frac{\mu 1}{\lambda 1 + \mu 1}$$

$$A_{SS\text{-}processor} = \frac{\mu 2}{\lambda 2 + \mu 2}$$

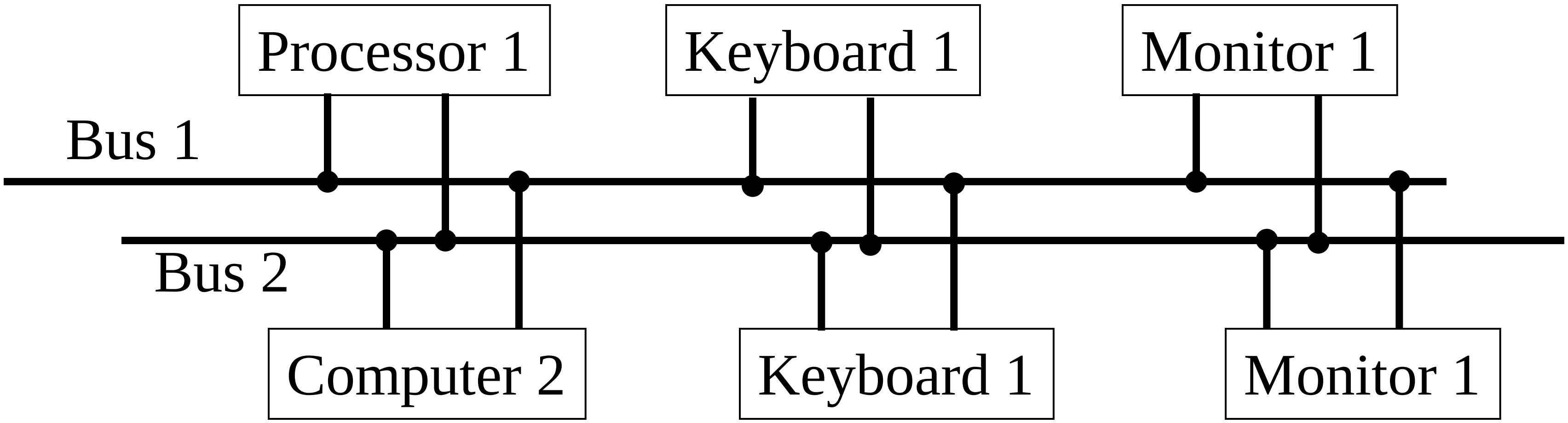
$$A_{SS\text{-}keyboard} = \frac{\mu 3}{\lambda 3 + \mu 3}$$

$$A_{SS\text{-}system\text{-}parallel} =$$

$$1 - \left(1 - \left(\frac{\mu 1}{\lambda 1 + \mu 1}\right)\left(\frac{\mu 2}{\lambda 2 + \mu 2}\right)\left(\frac{\mu 3}{\lambda 3 + \mu 3}\right)\right)^2$$



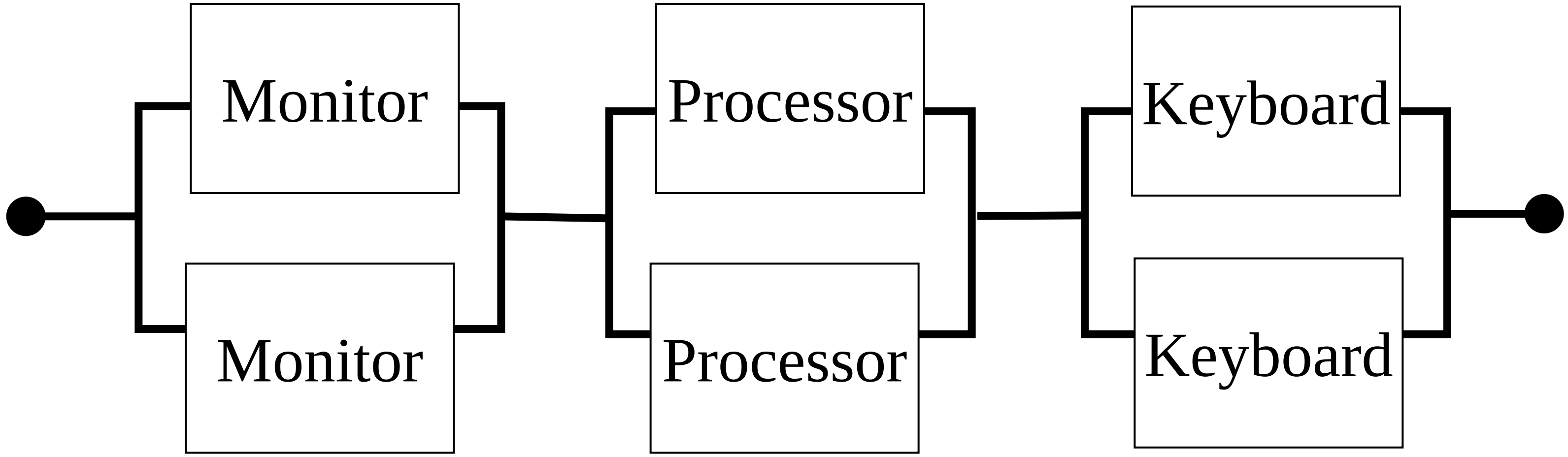
Parallel/Series System: Example



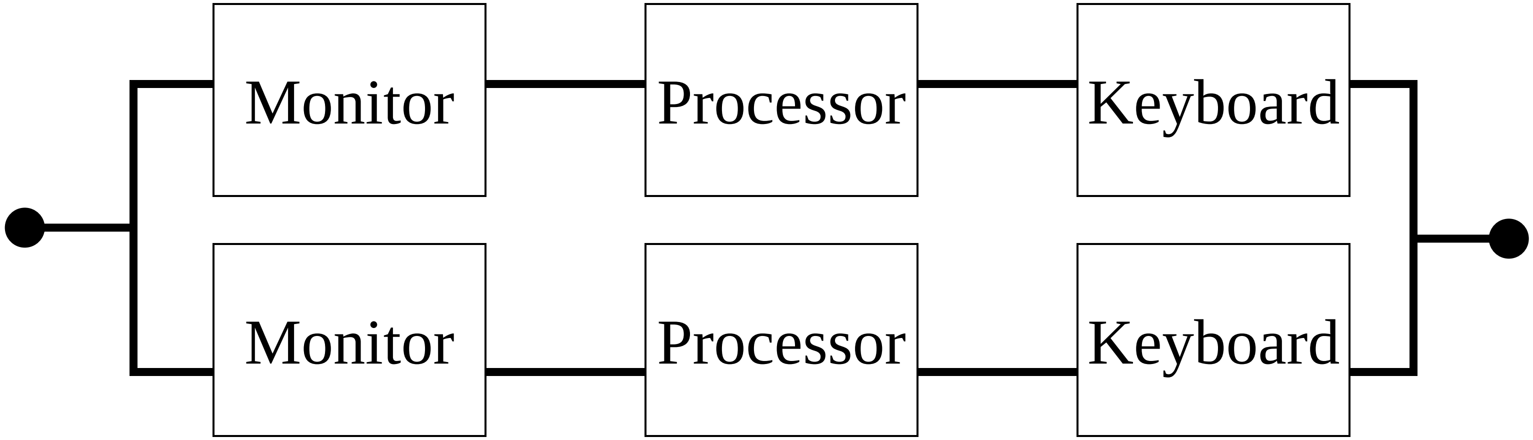
What is the corresponding RBD ?

Corresponding RBD

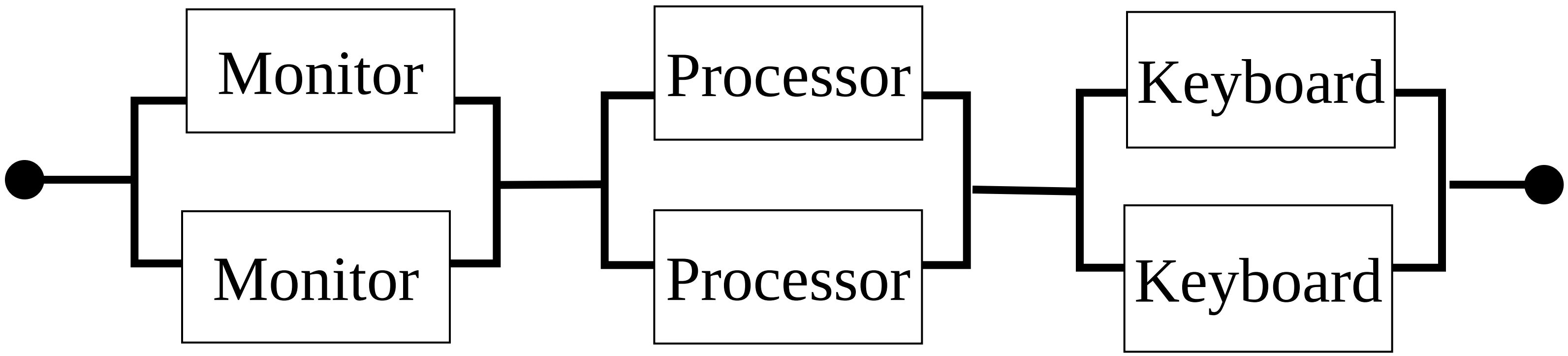
Assuming Buses are perfect



Compare to the RBD below, which one has better reliability?

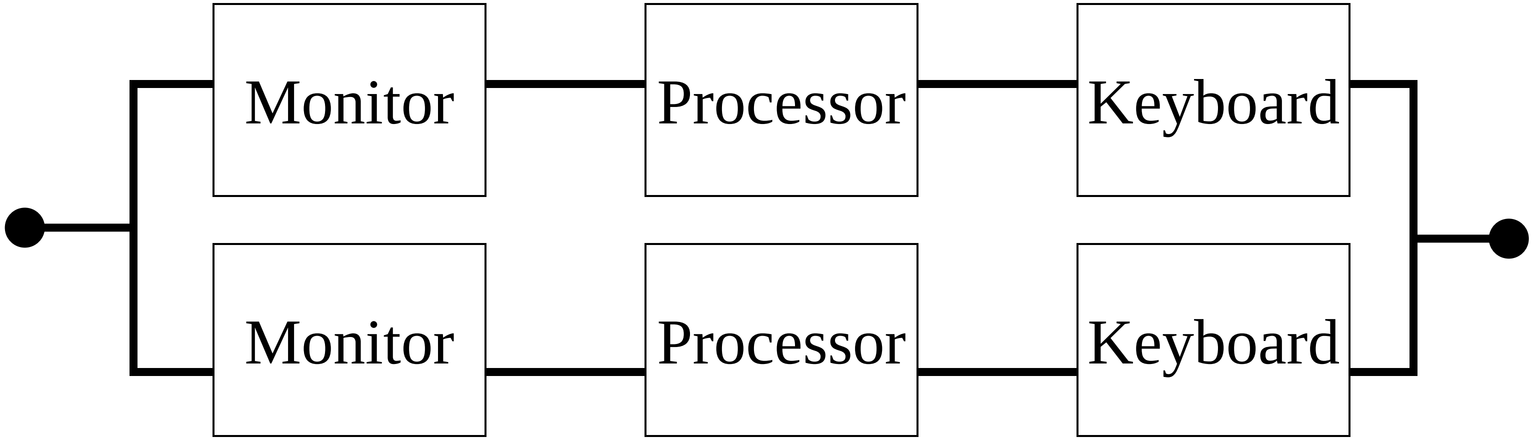


Numerical Comparison(1)



Component	Pw	Pf	Pw (1 of 2)
Monitor	0.99	0.01	0.9999
Keyboard	0.9	0.1	0.99
Processor	0.999	0.001	0.999999
			Psystem-w 0.98990001

Numerical Comparison (2)

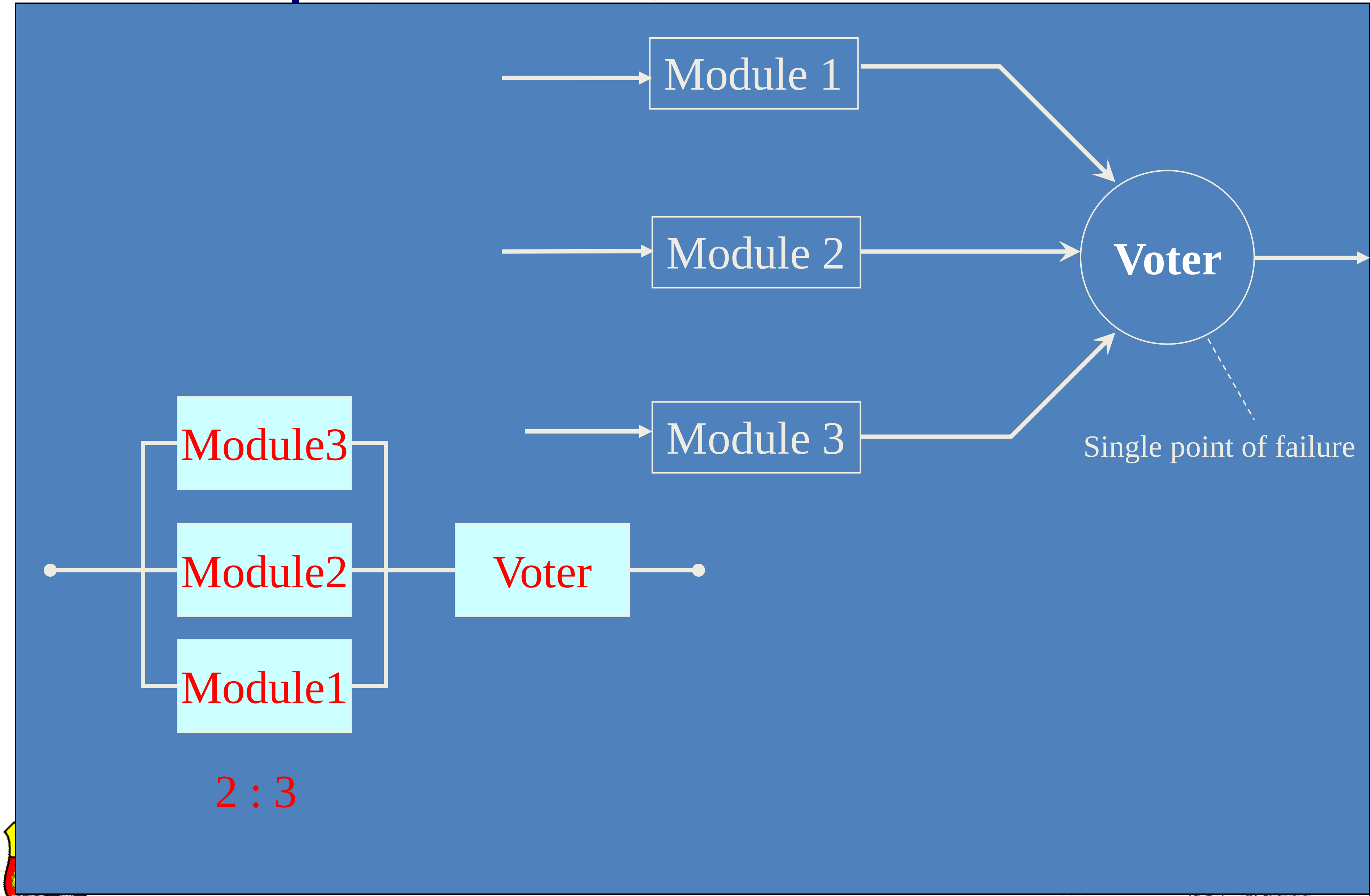


Component	Pw	Pf	Pw-single 0.890109 Psystem-w 0.987923968
Monitor	0.99	0.01	
Keyboard	0.9	0.1	
Processor	0.999	0.001	

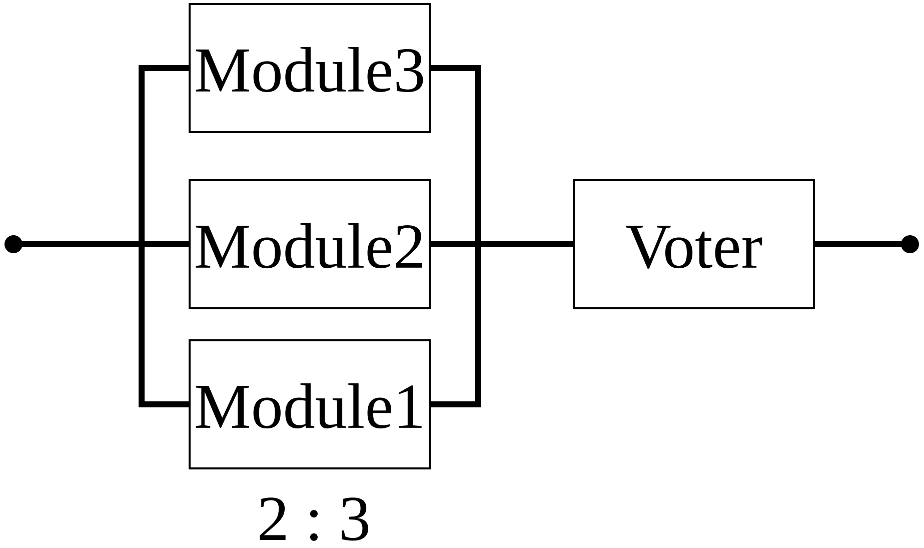
N Modular Redundancy

- ***M*** of ***N*** System
  - ***M*** of the total of ***N*** identical modules are required to function, $M \leq N$
  - **TMR** (Triple Modular Redundancy) is a famous example, where ***M*** is 2 and ***N*** is 3

Example: RBD for TMR



Reliability Calculation for TMR



Cases for the TMR to be working:

- all of the 3 modules are working
- any 2 modules are working, and 1 module is failed

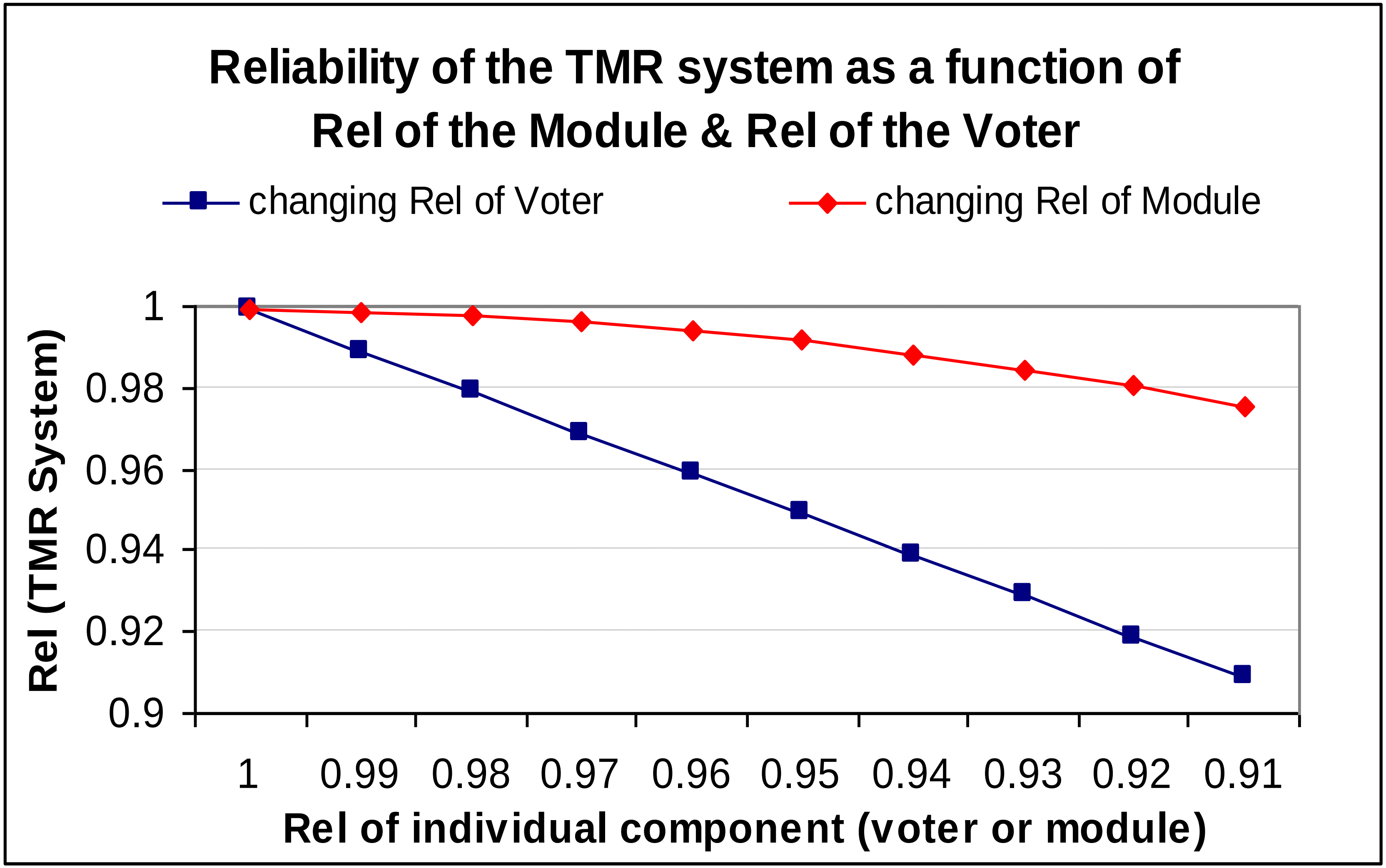
Look at it from another way:

Cases for the TMR to be failed

- all 3 modules are failed
- any one module is working, however, the rest 2 are not working

Remember, the voter is a Single-Point-Of-Failure

Module	voter	TMR	System Pw
0.999	0.999	0.999997	0.998997005



From this chart, you can see the effect that a single point of failure made is much more significant than that of a component with redundancy

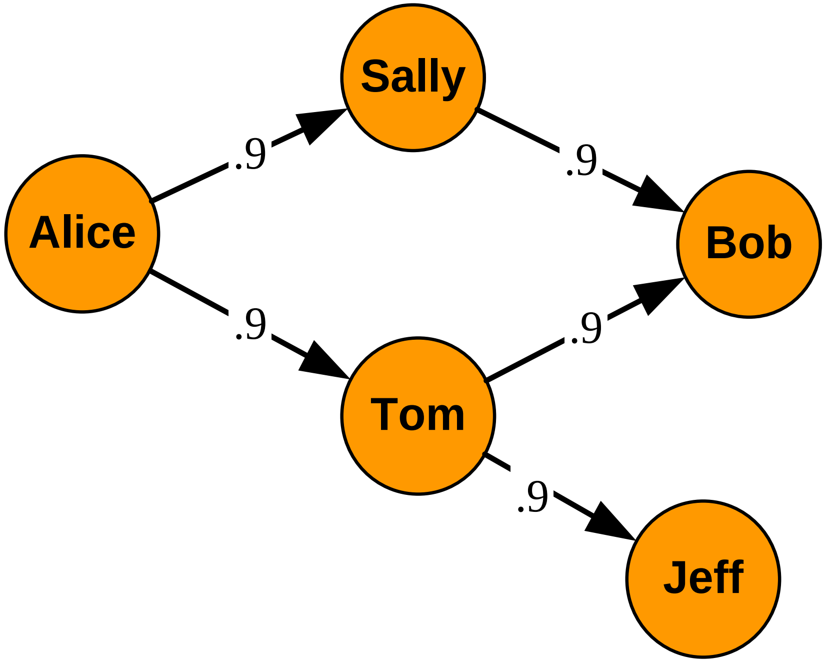
Bottom Line

- RBD provides the vehicles for analysts to construct models easier than the combinatorial approach
- The fundamental math is the same
- The reliability/availability calculation methods are provided by the methodology

Network Reliability

- Can compute the reliability of any network
- Use series and parallel analysis already described
- Method: series-parallel reduction
  - Use graph-theoretic (logical) reduction of system topology
  - Insert failure rates into equations

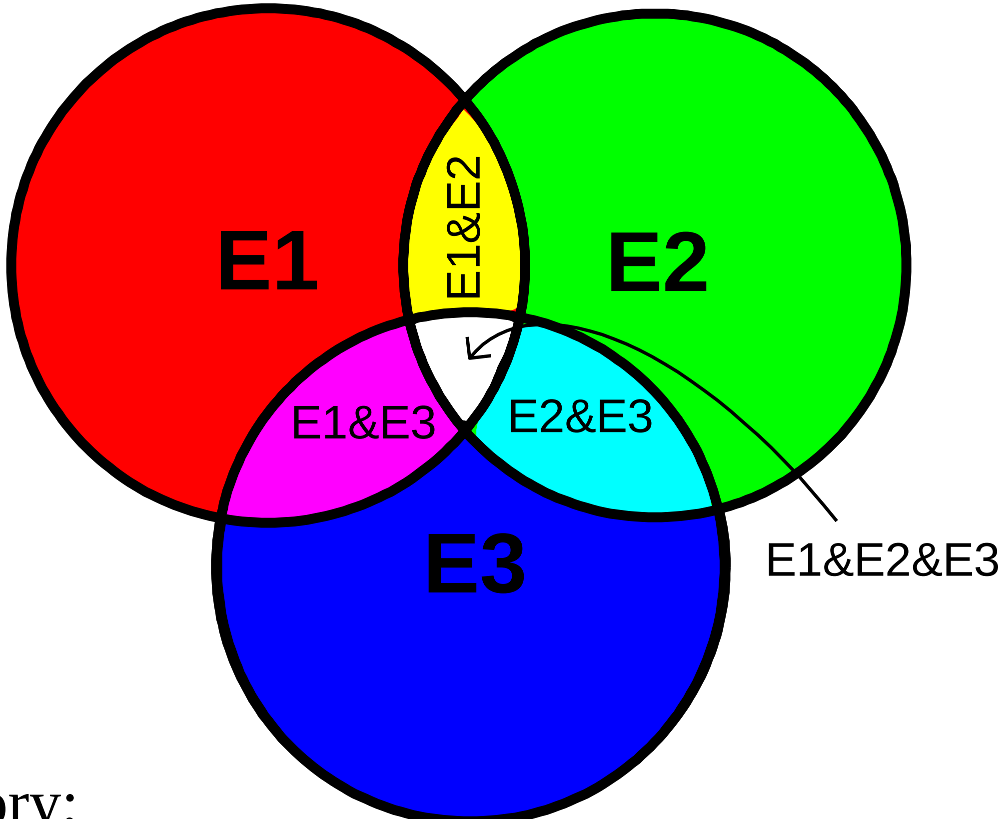
An example of 2-terminal reliability



Compute the probability of a communication channel between Alice and Bob existing.

$$
\begin{aligned}
Rel_{Alice,Bob} &= \text{Prob(any path from Alice to Bob)} \\
&= \text{1-Prob(all paths failed)} \\
&= 1 - (1 - .81)(1 - .81) \\
&= .9639
\end{aligned}
$$

General Computation: Use Inclusion-Exclusion



From set theory;

$$
\begin{aligned}
|E1 \cup E2 \cup E3| = \; & |E1| + |E2| + |E3| \\
& - |E1 \cap E2| - |E1 \cap E3| - |E2 \cap E3| \\
& + |E1 \cap E2 \cap E3|
\end{aligned}
$$

Inclusion-Exclusion applied to operational probabilities

Another way to derive the inclusion-exclusion algorithm (for 3 components)

-p_i is Prob(path *i* fails)

$$
\begin{aligned}
P(\text{any path}) &= 1 - P(\text{all paths failed}) \\
&= 1 - (1 - p_1)(1 - p_2)(1 - p_3) \\
&= 1 - (1 - p_1 - p_2 + p_1p_2)(1 - p_3) \\
&= 1 - (1 - p_1 - p_2 + p_1p_2 - p_3 + p_1p_3 + p_2p_3 - p_1p_2p_3) \\
&= p_1 + p_2 + p_3 - p_1p_2 - p_1p_3 - p_2p_3 + p_1p_2p_3
\end{aligned}
$$

General Network Reliability

"measure of the ability of a network to carry out a desired network operation."[colbourn87]

$$Rel(G) = \sum_{\substack{S \subseteq E(G); \\ S \text{ is operational}}} \prod_{e \in S} p_e \cdot \prod_{\substack{e \in \\ E(G)-S}} (1 - p_e)$$

Operational Criterion is the distinguishing feature of different metrics

Probability of "operation" of the arc *e*.

Primary Graph Reductions

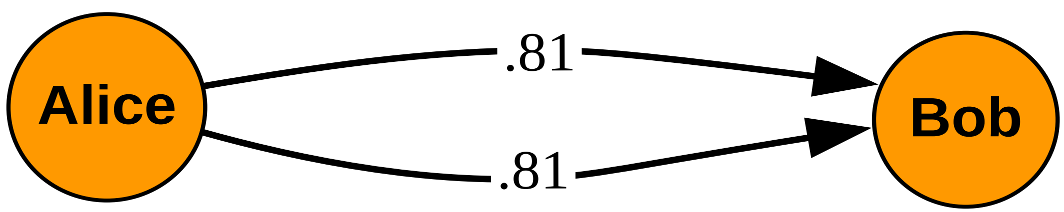
- Irrelevant – do not contribute to any operational state; remove
- Series – sequence of edges are required simultaneously; combine with axiom of probability:

$$P(A \cap B) = P(A)P(B)$$

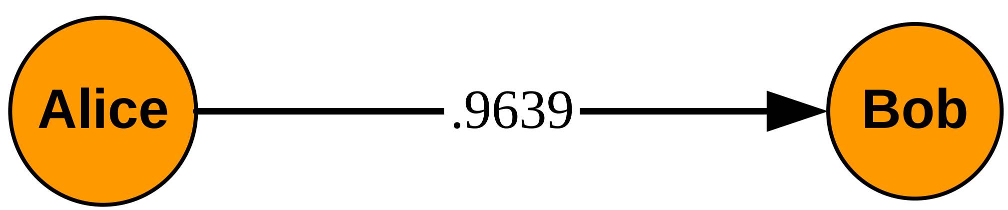
- Parallel – network is operational if any of these edges are operational; combine with axiom of probability:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Sequential reduction



Parallel reduction



Hierarchical Composition Method

- Given a detailed description of a system, too many components are displayed, which makes the modeling task difficult which creates unnecessary complexity
- Abstract the detailed description into a higher level description – **hierarchical composition method**