EE5903 RTS
Chapter 3
RTS – Reliability & Fault-Tolerance

Bharadwaj Veeravalli

elebv@nus.edu.sg

Contents

- Reliability in RTS
- Failure rate functions – modeling
- Failure modes
- Failure rate distributions
- State dependent systems – Quantifying reliability in software/hardware configurations
- Reliability-Aware Systems design – Different configurations
- Redundancy
- K out of N redundancy
- Availability & related concepts

Reliability in RTS

- Failure can occur at any part of a RTS − from sensors to actuators, including computing components

  -- Failures due to - Stress/pressure / heat / current and voltage surges / software bugs / unexpected critical inputs / …

- Failure can occur at any part of a RTS, either independently or in a dependent way – from sensors to actuators, including computing components

Reliability ··· (Cont'd)

- *Time-to-Failure* – An important parameter of interest to RTS;

  -- The ***Time-to-Failure*** **T**, of a component (or system) is random variable taking values on the range [0, infinity); Also called "life-time" and "failure-time".

- Parameter T may not represent time always – Ex: distance traveled by a car, # of actuations of a switch, etc;
- A particular value of T is denoted by 't'

Reliability ⋯ (Cont'd)

- Some meaningful questions:

  -What is the probability that a component fails at t, i.e., P(T=t)?

  - P(a < T <b) = ?

  - What is the expected value of T?

  - What is the spread of T?

  - How is T distributed?

In a multiple/multi-core CPU system – failure of one core affects the performance in a cooperating computing environment

Reliability ⋯ (Cont'd)

Example: Assume that a multi-core CPU system involved in a cooperative computing task. Suppose T is the time-to-failure of a core expressed in hours. Determine the probability that the CPU system will fail in the second hour assuming T has the following pdf:

$$f(t) = 0.2e^{-0.2t} \quad ; t \geq 0$$

$$P(\text{Failure in second hour}) = P(1 < T < 2)$$

$$= 0.2\int_{1}^{2} \exp(-0.2t)dt$$

$$= e^{-0.2} - e^{-0.4} = 0.148$$

We assume a single core failure would terminate the cooperative computing task.

Reliability ⋯ (Cont'd)

MTTF – Mean time to failure is the expectation of T, given by,

$$\begin{aligned}\text{MTTF} &= \mu_\text{T}\\ &= \text{E[T]}\\ &= \int_0^\infty \text{t f(t) dt}\end{aligned}$$

Continuing our example, we have MTTF as,

$$\begin{aligned}\text{MTTF} &= \text{E[T]}\\ &= \int_0^\infty 0.2\text{te}^{-0.2\text{t}}\,\text{dt}\\ &= 5\ \text{hrs}\end{aligned}$$

The variance of T, denoted $\sigma_\text{T}^{\,2}$, is the expectation of $(\text{T} - \mu_\text{T})^2$

Reliability ··· (Cont'd)

Variance:

$$\sigma_T{}^2 = \mathrm{E}[(\mathrm{T} - \mu_T)^2]$$
$$= \mathrm{E}(\mathrm{T}^2 - 2\mu_T T + \mu_T{}^2$$
$$= \mathrm{E}(\mathrm{T}^2) - 2\mu_T \mathrm{E}(\mathrm{T}) + \mu_T{}^2$$
$$= \mathrm{E}(\mathrm{T}^2) - \mu_T{}^2$$

Continuing our example, we have,

$$\sigma_T{}^2 = \mathrm{E}[\mathrm{T}^2] - \mu_T{}^2$$
$$= \int_0^\infty t^2 f(t)dt - \mu_T{}^2$$
$$= \int_0^\infty 0.2t^2 e^{-0.2t} dt - 25$$
$$= 25\ \mathrm{hr}^2$$
$$\sigma_T = 5\ \mathrm{hrs}$$

Reliability ··· (Cont'd)

- **RELIABILITY R(t)** is defined as the probability that a system/component will function over some time period t. Let T be a continuous RV denoting the time to failure of the system. Then:

$$R(t) = P(T \geq t)$$

$$= \int_{t}^{\infty} f(t)\, dt$$

- The **FALLIBILITY/failure F(t)** is the probability that a failure occurs before time t, i.e., in the time interval [0, t].

$$F(t) = P(0 \leq T \leq t)$$

Hence:

$$= \int_{0}^{t} f(t)\, dt$$

$$= 1 - R(t)$$

Reliability ··· (Cont'd)

- Clearly, F(t) is the complement of R(t) and it is also the *cumulative distribution function* (CDF), of T.
- Some important points to note:
- R(t) and F(t) are probabilities and hence they are bounded by 0 and 1.
- R(t) monotonically decreases from 1 to 0. $R(0) = 1 \quad R(\infty) = 0$
- F(t) monotonically increases from 0 to 1. $F(0) = 0 \quad F(\infty) = 1$
- $f(t) = \frac{dF(t)}{dt} = -\frac{dR(t)}{dt}$
- $P(a < T < b) = F(b) - F(a) = R(a) - R(b)$
- Alternatively, MTTF can be obtained directly from R(t): $MTTF = \int_0^\infty R(t)\, dt$

Reliability ⋯ (Cont'd)

Continuing our example, we have:

$$R(t) = e^{-0.2t} \qquad ; t \geq 0 \ (\textit{Verify!})$$

$$F(t) = 1 - e^{-0.2t} \quad ; t \geq 0$$

Question – How can we model faults and fault-tolerance?

- What is the underlying factor to observe / measure to quantify fault-tolerance?

Failure Rate / Hazard Rate Function

- The **FAILURE / Hazard RATE** $\lambda(t)$ of a component at time t, is the instantaneous rate of failure at time t given that it has survived to time t.
- If the component fails during the interval $(t, t+\delta)$ then it must be working up to time t. Hence, we consider the conditional probability:

$$P(t < T < t+\delta \mid T > t) = \frac{P(t < T < t+\delta)}{P(T > t)}$$

$$= \frac{f(t)\delta}{R(t)}$$

Dividing by δ to obtain a rate:

$$\frac{P(t < T < t+\delta \mid T > t)}{\delta} = \frac{f(t)}{R(t)}$$

Failure Rate Function (Cont'd)

- Taking the limit as δ tends to zero: $$\lambda(t) = \lim_{\delta\to 0} \frac{P(t < T < t + T \mid T > t)}{\delta}$$ $$= \lim_{\delta\to 0} \frac{f(t)}{R(t)}$$ $$= \frac{f(t)}{R(t)} \quad \text{OR} \quad -\frac{dR(t)}{dt} \cdot \frac{1}{R(t)}$$

- Example: Consider the following pdf:

$$f(t) = 0.2\, e^{-0.2t} \quad ; \ t \geq 0$$

- The reliability function is: $R(t) = e^{-0.2t} \quad ; \ t \geq 0$

- Hence, the failure rate is: $\lambda(t) = 0.2 \quad ; \ t \geq 0$

- Therefore in this case, the failure rate is constant throughout the life of the component. As an exercise, plot f(t), R(t), F(t) and $\lambda(t)$

Failure Rate Function (Cont’d)

- Often it is more useful to obtain R(t) from $\lambda(t)$ rather than $\lambda(t)$ from f(t) and R(t)

Starting from: $\lambda(t) = -\frac{\mathrm{dR(t)}}{\mathrm{dt}} . \frac{1}{\mathrm{R(t)}}$

We can rearrange to obtain: $-\lambda(t)\mathrm{dt} = \frac{\mathrm{dR(t)}}{\mathrm{R(t)}}$

Integrating on both sides, $-\int_0^t \lambda(t)\mathrm{dt} = \int_{\mathrm{R(0)}}^{\mathrm{R(t)}} \frac{\mathrm{dR(t)}}{\mathrm{R(t)}}$

Hence,

$$-\int_0^t \lambda(t)\mathrm{dt} = \ln\,(\mathrm{R(t)} - \ln(\mathrm{R}(0))$$

Failure Rate Function (Cont’d)

- But R(0) is equal to unity:

$$-\int_0^t \lambda(t)\mathrm{dt} = \ln(\mathrm{R(t)})$$

- Finally:

$$\mathrm{R(t)} = \exp\left[-\int_0^t \lambda(t)\mathrm{dt}\right]$$

Remarks:
Often failures of CPU cores are quantified as failure rate functions, which obey certain distributions. Hence, reliability-aware designs are enforced for embedded RTS.

- Example: If $\lambda(t)$ is equal to 0.1t then:

$$\mathrm{R(t)} = \exp\left[-\int_0^t 0.1\mathrm{tdt}\right]$$

$$= \mathrm{e}^{-0.05\mathrm{t}^2}$$

Memoryless Property of CFR Model

- CFR Model (exponential distribution) has a special property – Memoryless property!
- The time to failure of a component is **NOT** dependent on how long it has been operating!

  This means that the probability that a component will operate for the next 1000 hrs is same regardless of whether the component is brand new or has been operating several hours earlier!
- How do we prove this property?

  Assumption: Completely random and independent nature of the failure process.

  $R(t/T_0) = \mathbf{Pr[(T > T_0 + t) \mid (T > T_0)]} = Pr[T > T_0 + t] / Pr[T > T_0] = R(t+T_0) / R(T_0)$.

  Now, use CFR equation (exponential distribution) and simplify and show that $R(t/T_0) = R(t)$.

Some interesting points to ponder…

- What is **design life time** of a component? This is defined as the time to failure t_R that corresponds to a specified reliability R. It takes into account the built-in reliability of a component! Thus, for a given reliability R, for a CFR,

$$R(t_R) = e^{-\lambda t}{}_R = R \text{ from which we obtain:}$$

$$t_R = \mathbf{-(1/\lambda).ln(R)}$$

CFR Model for independent systems

Classroom discussions - Consider a system with two independent and redundant cores, each having the same failure rate, say λ. For each core we assume that we follow a CFR model and a system failure will occur when both components have failed.

In this case, ***what can we say about the behavior of the failure rate or hazard rate of the entire 2-core system?***

Failure Modes

- Often there are many different ways in which a component can fail. Each failure mechanism is called a failure mode.
  - Usually, each failure mode is independent of other failure modes.
  - Each failure mode has its own f(t) and hence R(t), etc.
  - The occurrence of any of the failure modes is deemed sufficient to cause the failure of the component (more in s/w than in h/w)

- Let R(t) be the reliability of the component.

Failure Modes (Cont'd)

$R_j(t)$ be the reliability of the j-th failure mode, and is the probability that this mode of failure does not occur before time t.

$R(t)$ = P({ Mode 1 does not occur before time t }
AND
{ Mode 2 does not occur before time t }
AND . . . AND
{ Mode n does not occur before time t })

$$= \prod_{j=1}^{n} \text{P(Mode j does not occur before time t)}$$

- Thus, $R(t) = R_1(t)\, R_2(t) \ldots R_n(t)$

Failure Modes (Cont'd)

- Continuing:

$$\mathrm{R(t)} = \exp\left[-\int_0^t \lambda(t)\mathrm{dt}\right]$$

and

$$\mathrm{R_j(t)} = \exp\left[-\int_0^t \lambda_\mathrm{j}(t)\mathrm{dt}\right]$$

where $\lambda_\mathrm{j}(t)$ is the failure rate of the j-th failure mode.

$$\exp\left[-\int_0^t \lambda(t)\mathrm{dt}\right] = \prod_{\mathrm{j=1}}^{\mathrm{n}} \exp\left[-\int_0^t \lambda_\mathrm{j}(t)\mathrm{dt}\right]$$

- Since:

$R(t) = R_1(t)\, R_2(t) \ldots R_n(t)$, we have,

Failure Modes (Cont'd)

$$\exp\left[-\int_0^t \lambda(t)\mathrm{dt}\right] = \exp\left[-\sum_{j=1}^{n}\int_0^t \lambda_j(t)\mathrm{dt}\right]$$

$$= \exp\left[-\int_0^T \sum_{j=1}^{n} \lambda_j(\mathrm{t})\mathrm{dt}\right]$$

$$\lambda(t) = \lambda_1(t) + \lambda_2(t) + \ldots\ldots.\lambda_n(t)$$

That is, the overall component failure rate is the sum of the failure rates of each of the independent failure modes.

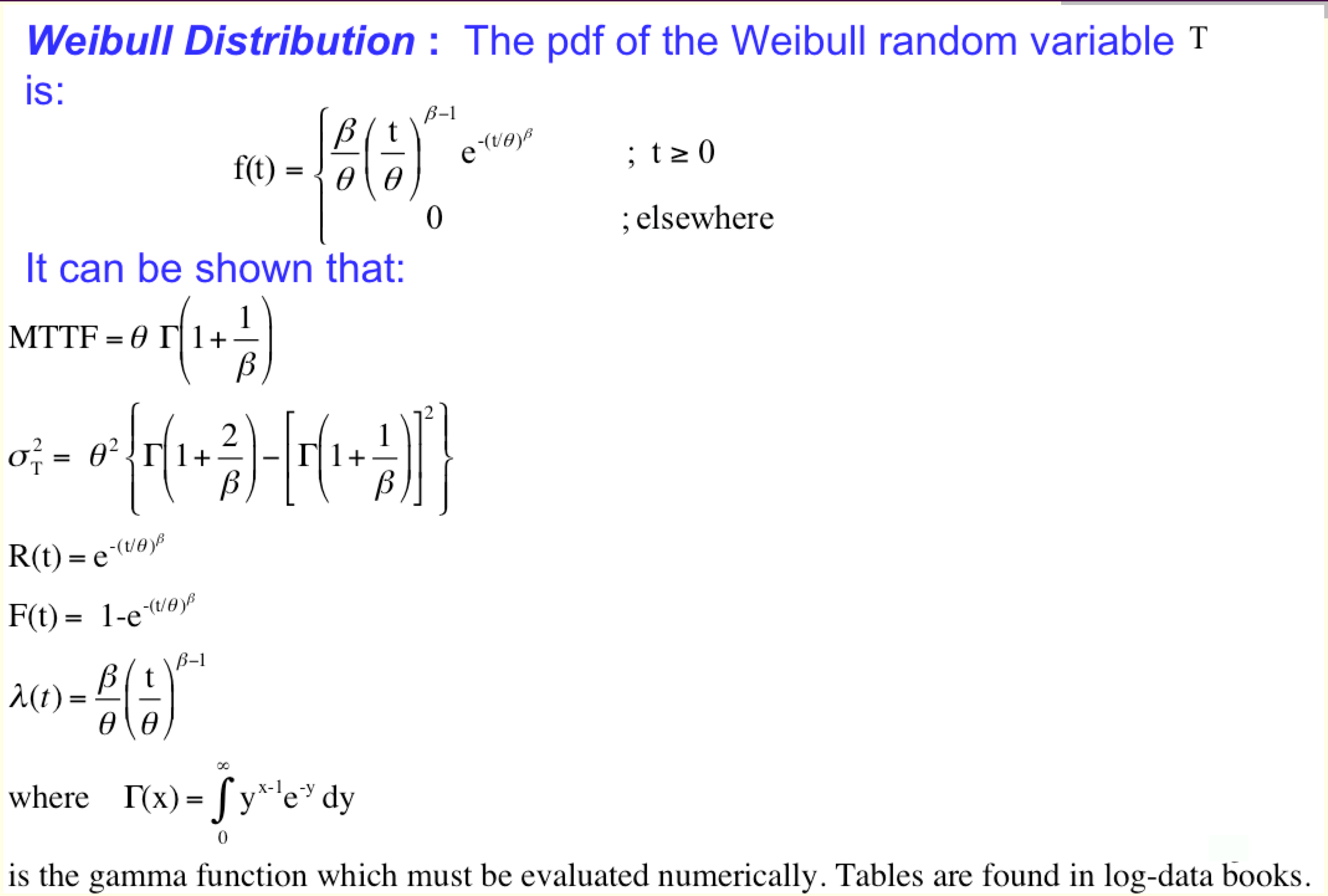
Failure distributions

- Any known distributions like, exponential, normal, log-normal can be assumed depending on the application/component failure historical data.
- Generic distribution – Weibull Distribution

Generates most commonly known distributions with respective parameter choices.

How does it look like?

Failure distributions ···(Cont’d)



Weibull Distribution **:** The pdf of the Weibull random variable T is:

$$f(t) = \begin{cases} \dfrac{\beta}{\theta}\left(\dfrac{t}{\theta}\right)^{\beta-1} e^{-(t/\theta)^{\beta}} & ;\ t \geq 0 \\ 0 & ;\ \text{elsewhere} \end{cases}$$

It can be shown that:

$$\mathrm{MTTF} = \theta\ \Gamma\left(1+\frac{1}{\beta}\right)$$

$$\sigma_{\mathrm{T}}^{2} = \theta^{2}\left\{\Gamma\left(1+\frac{2}{\beta}\right)-\left[\Gamma\left(1+\frac{1}{\beta}\right)\right]^{2}\right\}$$

$$R(t) = e^{-(t/\theta)^{\beta}}$$

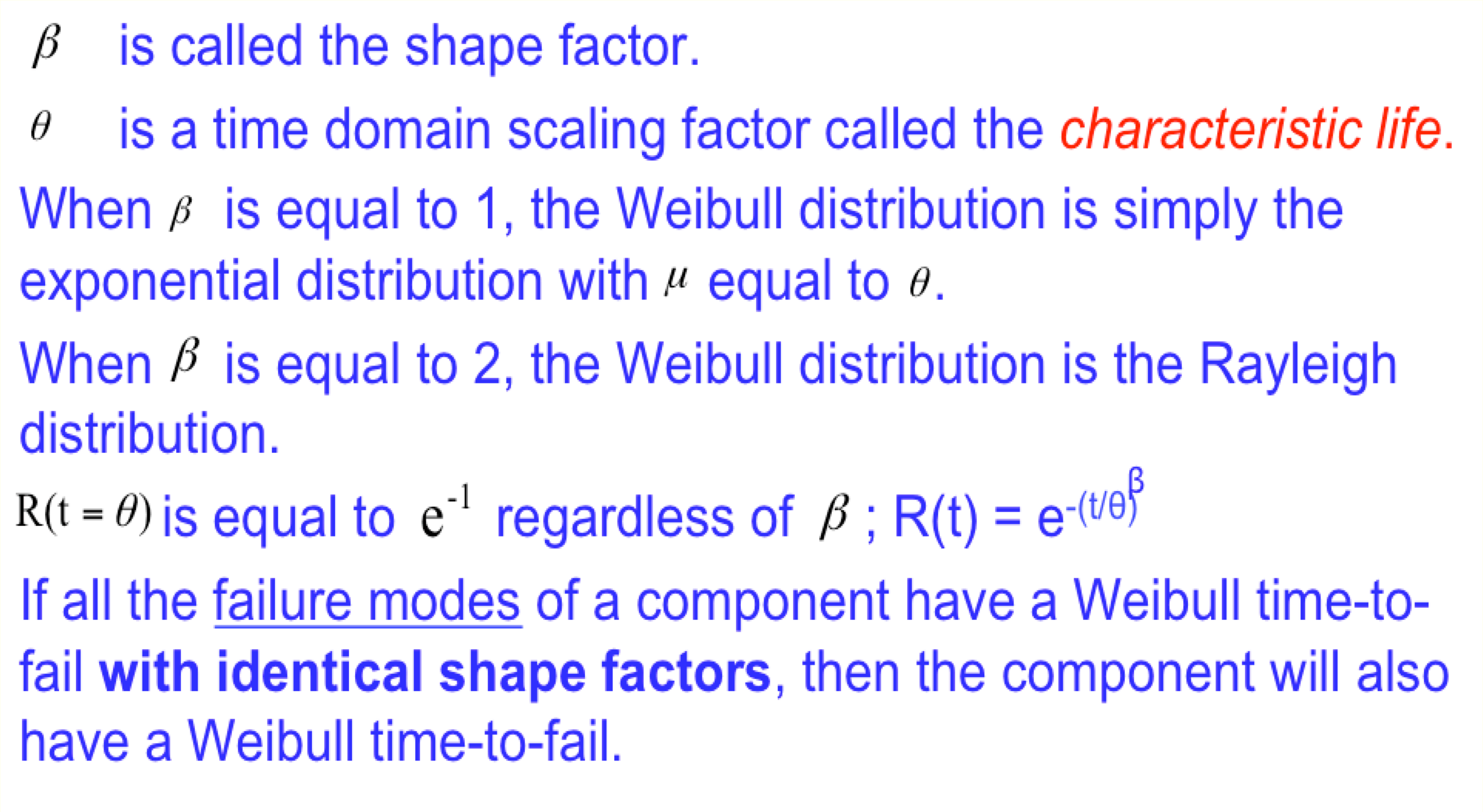
$$F(t) = 1-e^{-(t/\theta)^{\beta}}$$

$$\lambda(t) = \frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1}$$

where $\Gamma(x) = \int_{0}^{\infty} y^{x-1} e^{-y}\, dy$

is the gamma function which must be evaluated numerically. Tables are found in log-data books.

Failure distributions ···(Cont'd)



β is called the shape factor.

θ is a time domain scaling factor called the *characteristic life*.

When β is equal to 1, the Weibull distribution is simply the exponential distribution with μ equal to θ.

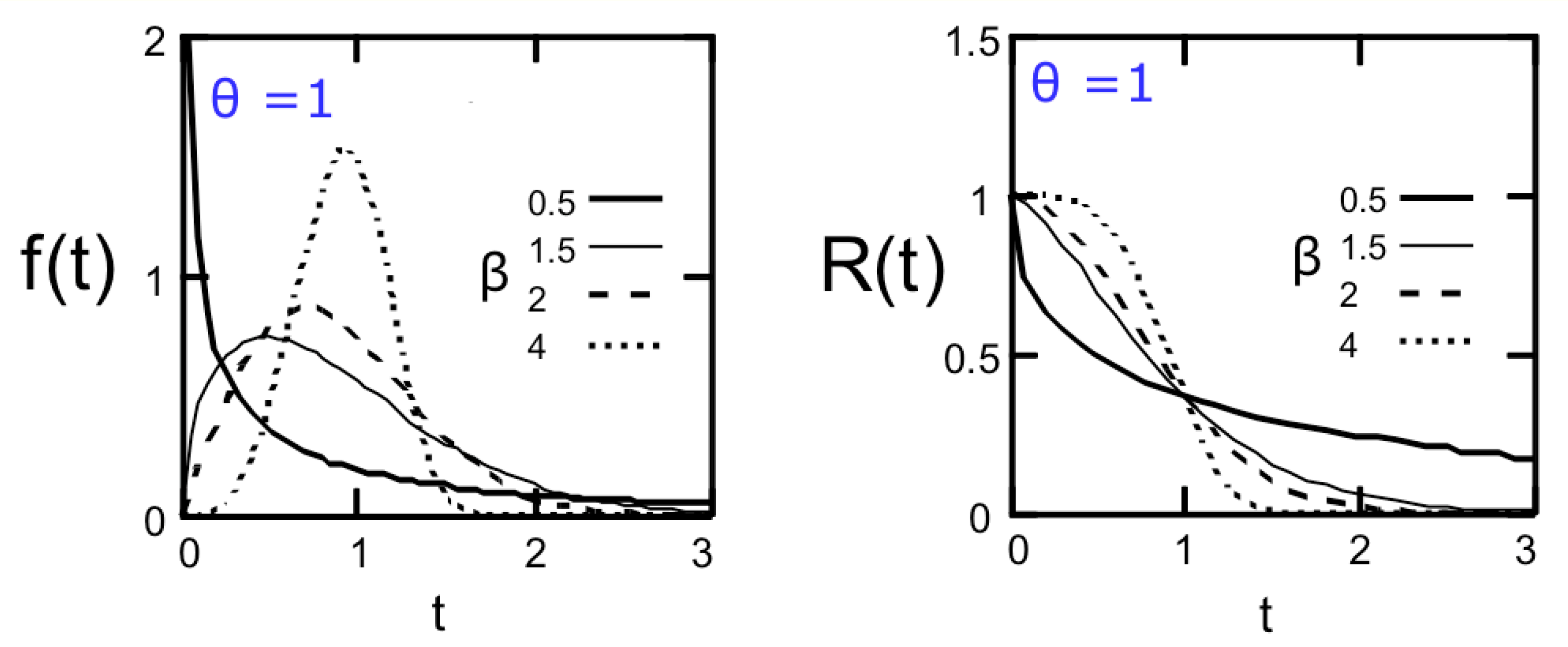
When β is equal to 2, the Weibull distribution is the Rayleigh distribution.

$R(t = \theta)$ is equal to e^{-1} regardless of β ; $R(t) = e^{-(t/\theta)^{\beta}}$

If all the failure modes of a component have a Weibull time-to-fail **with identical shape factors**, then the component will also have a Weibull time-to-fail.

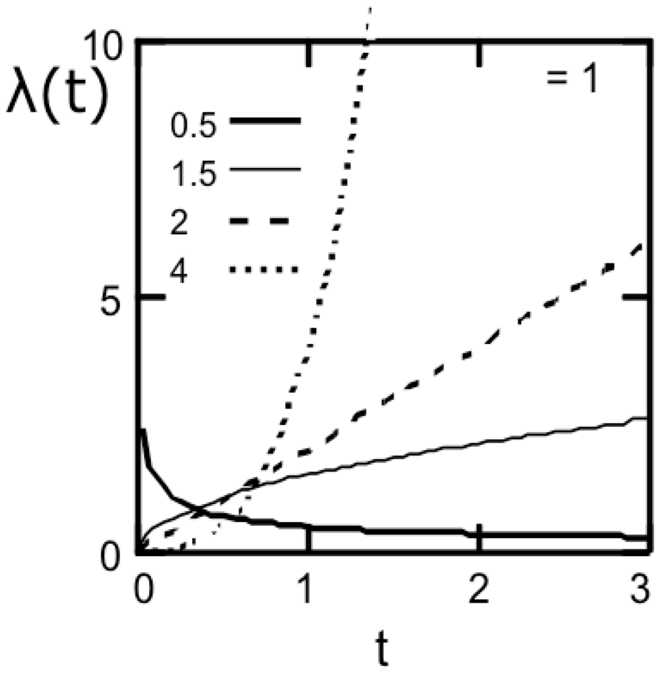
Failure distributions ···(Cont'd)

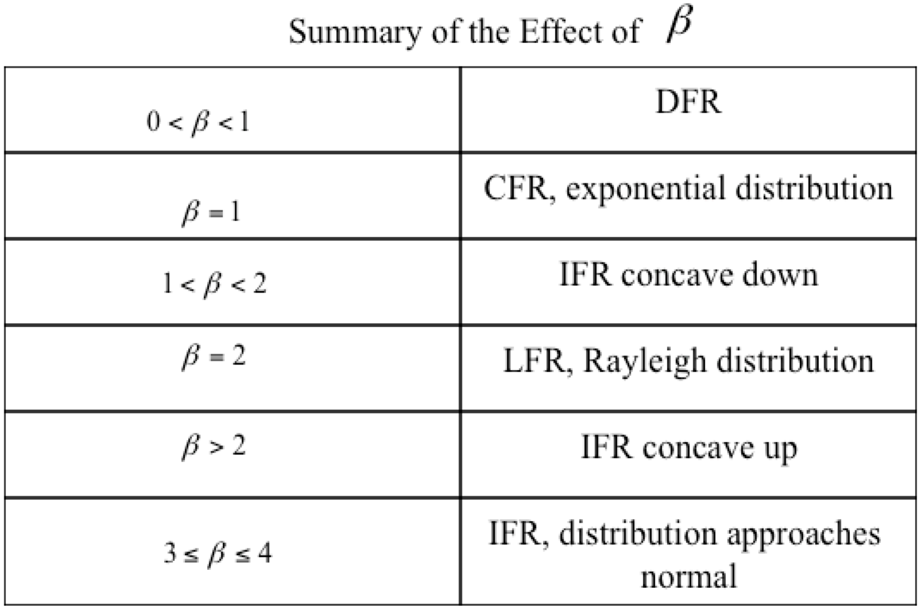
Weibull distributions (f(t) and R(t)) for certain β values:



Failure distributions ···(Cont’d)

Failure rate distribution (λ(t)) for certain β values:




Summary of the Effect of β

$0 < \beta < 1$	DFR
$\beta = 1$	CFR, exponential distribution
$1 < \beta < 2$	IFR concave down
$\beta = 2$	LFR, Rayleigh distribution
$\beta > 2$	IFR concave up
$3 \le \beta \le 4$	IFR, distribution approaches normal

State dependent systems – Real-Time Issues

A fundamental computation in Embedded RTS is the determination of system reliability from individual component reliabilities and the *system configuration*.

Component – Hardware / functional blocks / modules

In the so far analysis we always assumed independent failures of the components – this leads to a simple derivation of failure probabilities and reliabilities;

Q: What if the component failures are some way dependent on each other? (one s/w issue could trigger other failures in other components)

Q: What kind of analysis will help us to derive the reliabilities or probabilities of failure of the overall system?

State dependent systems ··· (Cont'd)

- **Markov Analysis** – Looks at a system as being in one of several states; Example of a state – all components are operating, only one component is operating, etc.
- Assumption – A fundamental assumption in using Markov analysis is that the probability that a RTS will undergo a transition from one state to another depends on the current state of the system and does not depend on how the system arrived at the current system! Thus, transition is not dependent on the past history;
- We will express the transition from one state to another state as an instantaneous failure rate;
- Assumption: Process is stationary – This means the transition probabilities do not change over time.

State dependent systems ··· (Cont'd)

Analysis for a two-core system

- Conditions – Each core will be one of the two states – operating (O) or failed (F);
- Thus for a n-core system the system can dwell in one of the 2^n states;
- For a 2 core system we will have 4 states as follows;

State	C1	C2	Remarks
1	O	O	O
2	F	O	F (for series)
3	O	F	F (for series)
4	F	F	F (for parallel & series)

One core is sufficient for task execution, however at the cost of missing the deadline or increase in response time.

State dependent systems ··· (Cont'd)

- If the two cores operate in parallel (shared memory) then only state 4 results in failure of the system;
- If the two cores operate in series (pipeline) then states 2, 3 and 4 would each constitute a failure state;
- *What is our objective*? Our objective is to find the probability of the overall 2- core system being in each state as a function of time.

We derive the probability of being in state **i** at time t as $P_i(t)$.

State dependent systems ··· (Cont'd)

(A) For a two-core series system: $R_s(t) = P_1(t)$

(B) For a two-core parallel system: $R_p(t) = P_1(t) + P_2(t) + P_3(t)$

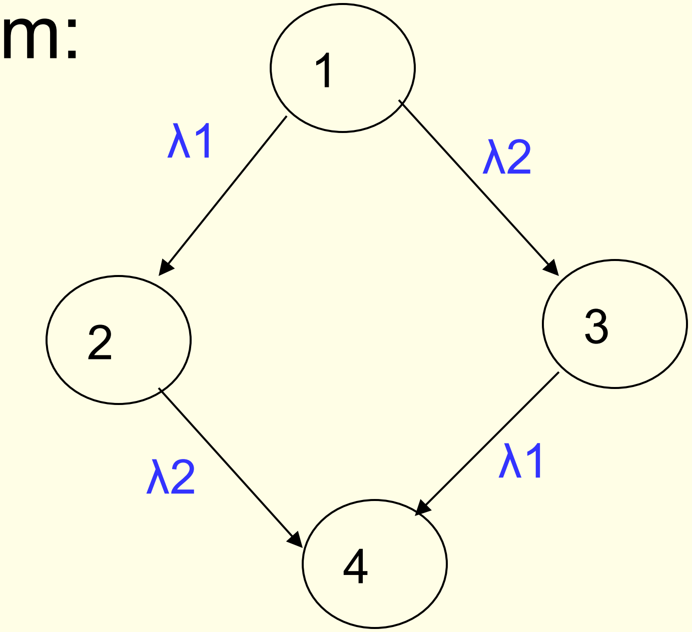
Observe that the system must be in one of the states at any given time; This implies:

$$P_1(t) + P_2(t) + P_3(t) + P_4(t) = 1 \qquad (1)$$

We need to determine each of these $P_i(t)$, i=1,2,3,4

State dependent systems ··· (Cont'd)

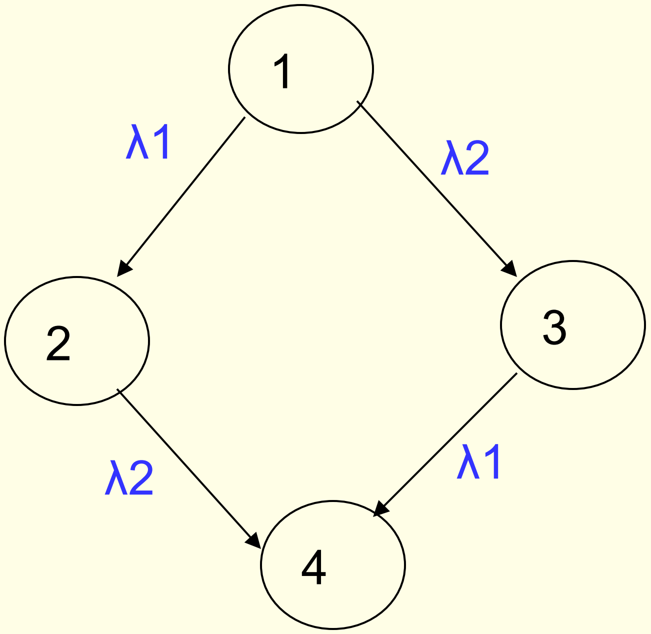
Assume that the individual cores have constant failure rates λ, we can represent the transitions possible for this two core system using the following state diagram:



Assumption : Transition from state 1 to 4 has very low probability or 0;

State dependent systems ··· (Cont'd)

From the above state diagram we derive the
following equations:



(2) $P_1(t + \Delta t) = P_1(t) - \lambda_1 \Delta t\, P_1(t) - \lambda_2 \Delta t\, P_1(t)$

$\lambda_1 \Delta t$: conditional probability of a transition to state 2 occurring during Δt given that the state of the system is currently in state 1;

$\lambda_1 \Delta t.\ P_1(t)$: is the joint probability of the system being in state 1 at time t and making a transition to state 2 during Δt.

We can similarly interpret other component too.

State dependent systems ⋯ (Cont'd)

Thus, from the rate diagram, we have:

(3) $P_2(t + \Delta t) = P_2(t) + \lambda_1 \Delta t\ P_1(t) – \lambda_2 \Delta t\ P_2(t)$

(4) $P3(t + \Delta t) = P3(t) + \lambda 2 \Delta t\ P1(t) – \lambda 1 \Delta t\ P3(t)$

(5) $P4(t + \Delta t) = P4(t) + \lambda 2 \Delta t\ P2(t) + \lambda 1 \Delta t\ P3(t)$

From (2), dividing the difference $P1(t + \Delta t) – P1(t)$ by Δt on both sides and taking the limit $\Delta t \rightarrow 0$ we obtain the following set of differential equations

State dependent systems ··· (Cont'd)

(6) $dP1(t)/dt = -(\lambda 1 + \lambda 1)P1(t)$

(7) $dP2(t)/dt = \lambda 1P1(t) - \lambda 2P2(t)$

(8) $dP3(t)/dt = \lambda 2P1(t) - \lambda 1P3(t)$

(6) – (8) can be solved easily. The solutions for these sets of differential equations are as follows:

$P1(t) = e^{-(\lambda 1 + \lambda 2)t}$

$P2(t) = e^{-\lambda}2^{t} - e^{-(\lambda 1 + \lambda 2)t}$

$P3(t) = e^{-\lambda}1^{t} - e^{-(\lambda 1 + \lambda 2)t}$

State dependent systems … (Cont'd)

State dependent systems with repair - Typical assumption in embedded systems / software architectures

Assume that repair is possible for a component. Typically true with large equipment. Let the repair and failure rates be constant (exponential distribution).

Let us suppose that a repair may be completed for a failed component before the other unit has failed. This implies that there will be no system failures. How do we model this system reliability and behavior?

The state diagram for a two component system is shown below.

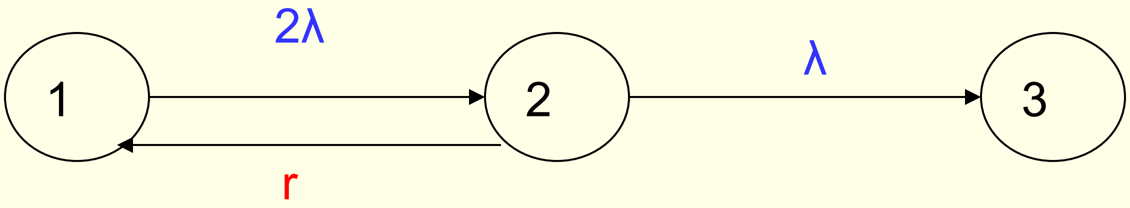
State 1 – no failures;

State – 2 – one of the components is operating;

State 3- both have failed;

State dependent systems ··· (Cont'd)

Let the failure rate be λ and the repair rate be r. Then,



(1) $dP_1(t)/dt = -(2\lambda)P_1(t) + rP_1(t)$

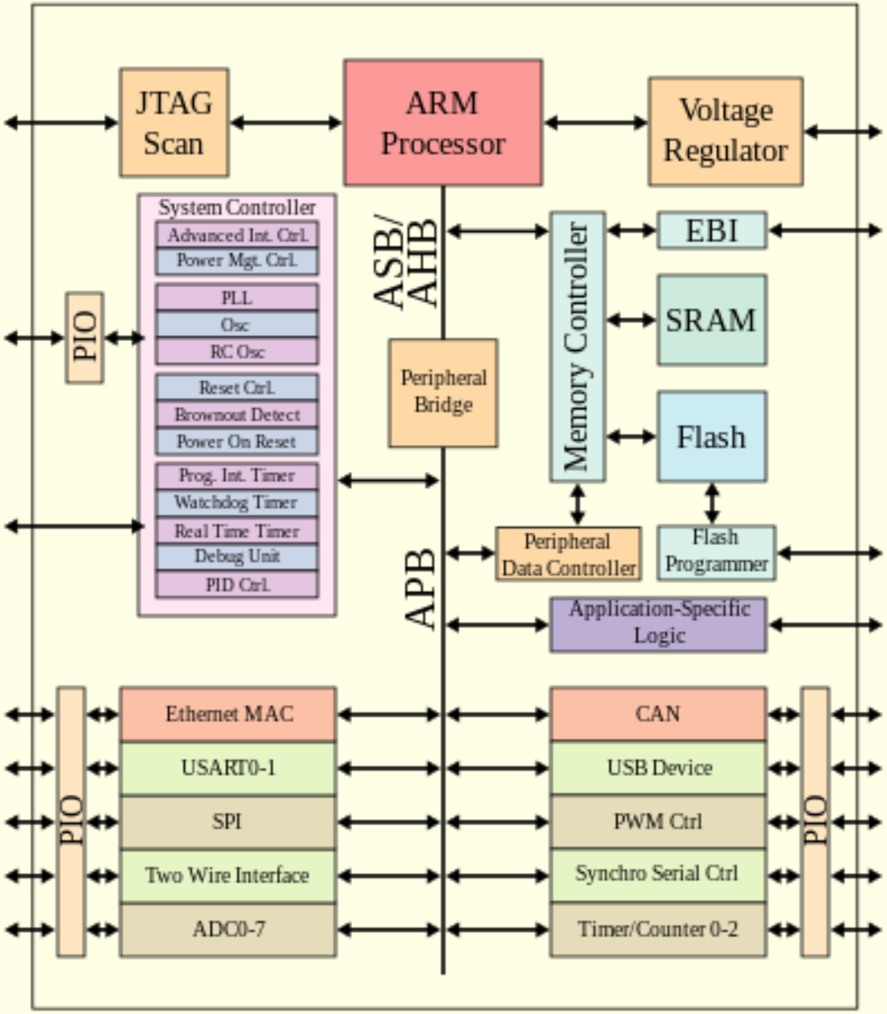
(2) $dP_2(t)/dt = 2\lambda P_1(t) - (\lambda+r)P_2(t)$

(3) $dP_3(t)/dt = \lambda P_2(t)$

(1) – (3) can be solved to obtain the individual $P_i(t)$, i=1,2,3 and hence the reliability can be computed using: $R(t) = 1 - P_3(t)$.

Obviously, if r=0 then we should be able to generate the same set of solutions derived for without repair case earlier.

SoC Architecture – Embedded System – Identifying Independent & Dependent components

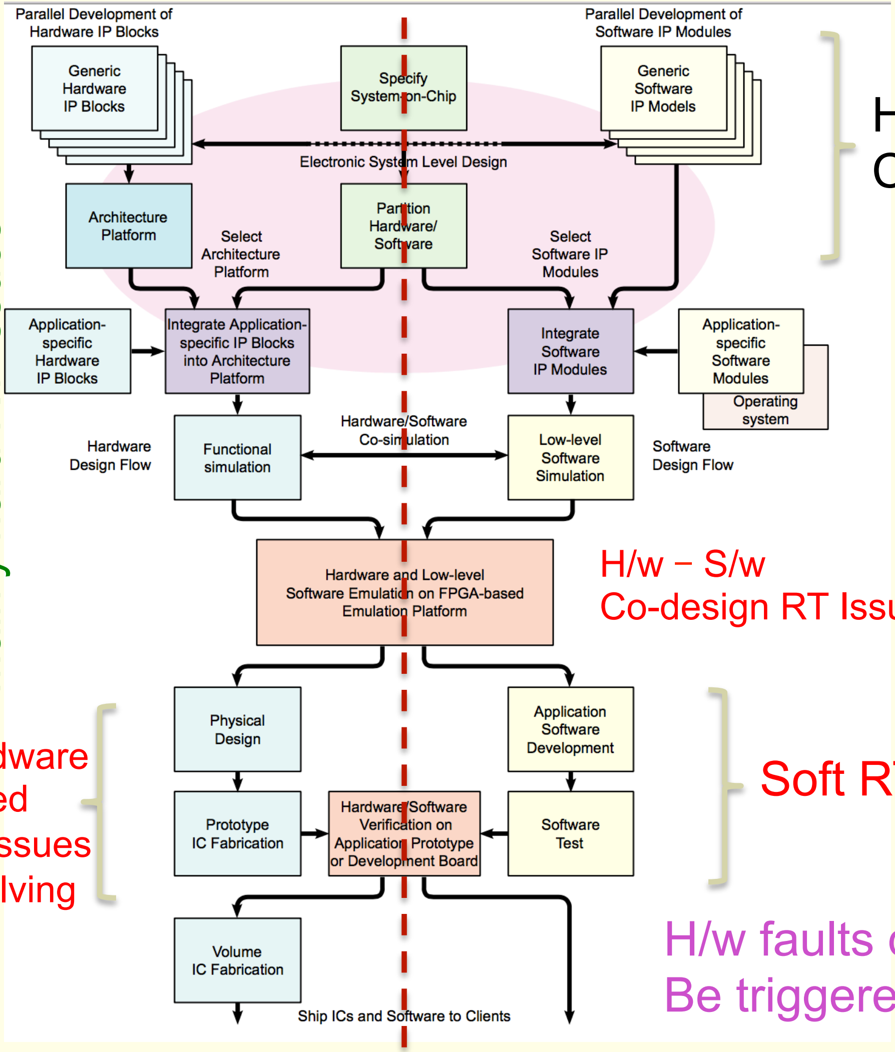


Independent and dependent components

Independent components – Can be replaced – redundancy measures; (Bridge, SRAM, etc)

Dependent components – Failure / fault /Error propagates to other components; (System controller, Voltage reg, etc)

Identifying RT issues SoC Architecture



Parallel Development of Hardware IP Blocks
Parallel Development of Software IP Modules
Generic Hardware IP Blocks
Specify System-on-Chip
Generic Software IP Models
Electronic System Level Design
Architecture Platform
Select Architecture Platform
Partition Hardware/ Software
Select Software IP Modules
Application-specific Hardware IP Blocks
Integrate Application-specific IP Blocks into Architecture Platform
Integrate Software IP Modules
Application-specific Software Modules
Operating system
Hardware Design Flow
Functional simulation
Hardware/Software Co-simulation
Low-level Software Simulation
Software Design Flow
Hardware and Low-level Software Emulation on FPGA-based Emulation Platform
Physical Design
Application Software Development
Prototype IC Fabrication
Hardware/Software Verification on Application Prototype or Development Board
Software Test
Volume IC Fabrication
Ship ICs and Software to Clients

Mainly hard RT Issues

H/w – S/w
Co-design RT Issues

End-to-end RT issues
are of major concern here

Mainly soft RT Issues

H/w – S/w
Co-design RT Issues

Hardware
based
RT issues
resolving

Soft RT Issues

Close to physical
H/W

H/w faults dominant; Faults can also
Be triggered by S/W from higher layers

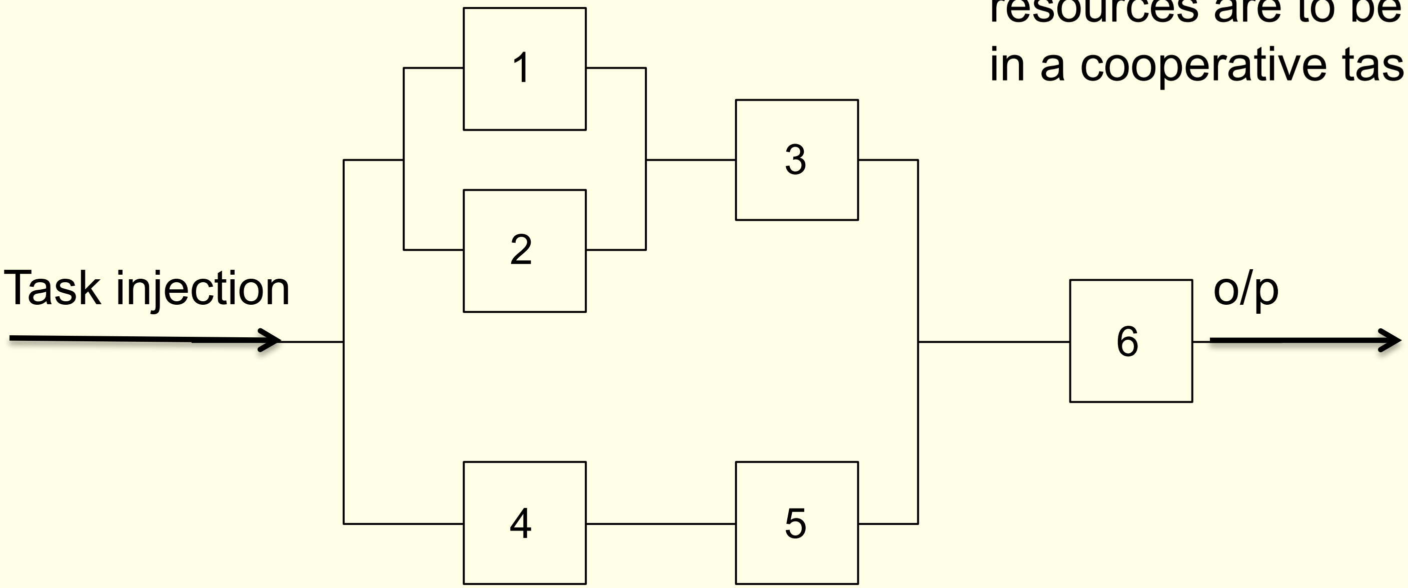
(c) Bharadwaj V 2016

RTS – Design & Analysis of Reliable System Configurations

Complex flow of control and data paths

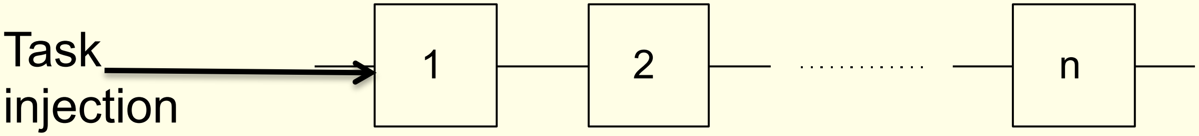
Possible execution flow in a software architecture -
Control Flow could trigger execution of {1,3,6} or {1,2,3,6} or {4,5,6}

This could also reflect the way resources are to be scheduled in a cooperative task execution



RTS – Design & Analysis of Reliable Hardware System Configurations – Building Redundancy

- If a system contains *n* execution components with a series reliability configuration:



- The overall reliability R(t), over a time t, of the system is:

R(t) = P(Component 1 functions over [0, t]
AND
Component 2 functions over [0, t]
AND ⋯ AND
Component *n* functions over [0, t])

Thus, $R(t) = R_1(t)\, R_2(t) \ldots R_n(t)$

where, Rj(t) is the reliability of the j-th component.

RTS – Design & Analysis of Reliable System Configurations

- **Question 3.1** - Consider a system of 10 computing components with a series reliability configuration. If the reliability of each component is 0.99, what is the overall reliability of the computing system ?

- The example above elicits the inherent weakness of a computing system with a series reliability configuration.
- Note that the result for a series reliability configuration is the same for failure modes of a component (Slide #21)

- Therefore in a series reliability configuration, the components are equivalent to the failure modes of a system.

RTS – Design & Analysis of Reliable System Configurations

Hence the failure rate $\lambda(t)$ of the system is: $\lambda(t) = \lambda_1(t) + \lambda_2(t) + ...\lambda_n(t)$ where, $\lambda_j(t)$ is the failure rate of the j-th component.

Clearly R(t) decreases and $\lambda(t)$ increases with increasing number of execution components in a series configuration.

Question 3.2 - If a system with a series reliability configuration contains three components with constant failure rates of 0.4, 0.5, and 0.6 failures/year respectively, determine the overall MTTF of the system. Comment on the failure rate of the overall system.

RTS – Design & Analysis of Reliable System Configurations

Theorem - If all the execution components of a series reliability config. have a Weibull distributed time-to-fail with identical shape factor β, then the overall system will have a Weibull distributed time-to-fail with shape factor β.

Proof: The failure rate of the j-th component will be:

$$\lambda_j(t) = \frac{\beta}{\theta_j}\left(\frac{\mathrm{t}}{\theta_j}\right)^{\beta-1}$$

where θ_j is the characteristic life-time of the j-th component. The system failure rate will then be:

$$\lambda(t) = \frac{\beta}{\theta_1}\left(\frac{\mathrm{t}}{\theta_1}\right)^{\beta-1} + \ldots\ldots\ldots \frac{\beta}{\theta_n}\left(\frac{\mathrm{t}}{\theta_n}\right)^{\beta-1}$$

$$= \beta\ \mathrm{t}^{\beta\text{-}1}\left(\frac{1}{\theta_1{}^{\beta}} + \frac{1}{\theta_2{}^{\beta}} + \ldots\ldots\frac{1}{\theta_n{}^{\beta}}\right)$$

RTS − Design & Analysis of Reliable System Configurations

- Now the system reliability is related to the failure rate by:

$$\mathrm{R(t)} = \exp\left[-\int_0^t \lambda(t)\mathrm{dt}\right]$$

$$= \exp\left[-\int_0^t \beta\, \mathrm{t}^{\beta\text{-}1}\left(\frac{1}{\theta_1{}^{\beta}} + \frac{1}{\theta_2{}^{\beta}} + \ldots\ldots\ldots \frac{1}{\theta_n{}^{\beta}}\right)\mathrm{dt}\right]$$

$$= \exp\left[-\mathrm{t}^{\beta}\left(\frac{1}{\theta_1{}^{\beta}} + \frac{1}{\theta_2{}^{\beta}} + \ldots\ldots\ldots \frac{1}{\theta_n{}^{\beta}}\right)\right]$$

- And hence R(t) can be written in the form: $\mathrm{R(t)} = \mathrm{e}^{-(\mathrm{t}/\theta)^{\beta}}$

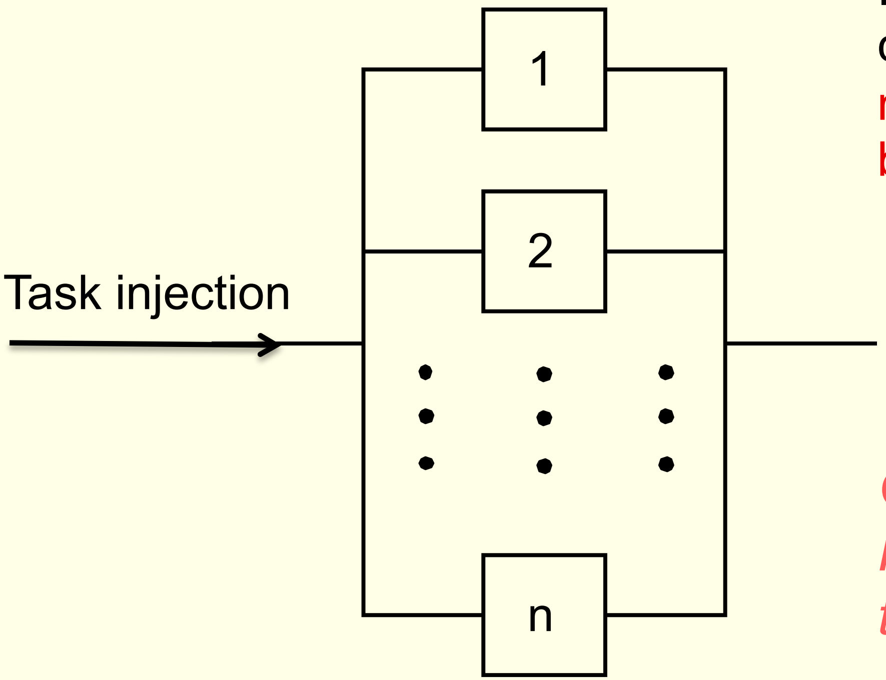
where θ is the characteristic life-time of the system and is given by:

$$\theta = \left(\frac{1}{\theta_1{}^{\beta}} + \frac{1}{\theta_2{}^{\beta}} + \ldots\ldots\ldots \frac{1}{\theta_n{}^{\beta}}\right)^{-\frac{1}{\beta}}$$

Clearly, R(t) corresponds to a Weibull time-to-fail with shape factor β and characteristic life-time θ. Hence the proof.

RTS – Design & Analysis of Reliable System Configurations

Parallel Computing Configuration



In a s/w architecture, this may correspond to more than one module / functional block being kept active at a time;

Q: What can we say about Failure rate function for this case?

RTS – Design & Analysis of Reliable System Configurations

- Then the **fallibility** F(t) of the system, over a time t, is:

F(t) = P (Component 1 fails over [0, t]
AND
Component 2 fails over [0, t]
AND ⋯ AND
Component *n* fails over [0, t])

= $F_1(t).F_2(t) \ldots F_n(t)$, where $F_j(t)$ is the fallibility of the j-th component.

- This is clearly the complimentary case of the series reliability configuration.

RTS – Design & Analysis of Reliable System Configurations

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

  $= 1 - F_1(t).F_2(t) \ldots F_n(t)$

  $= 1 - (1 - R_1(t))(1 - R_2(t)) \ldots (1 - R_n(t))$

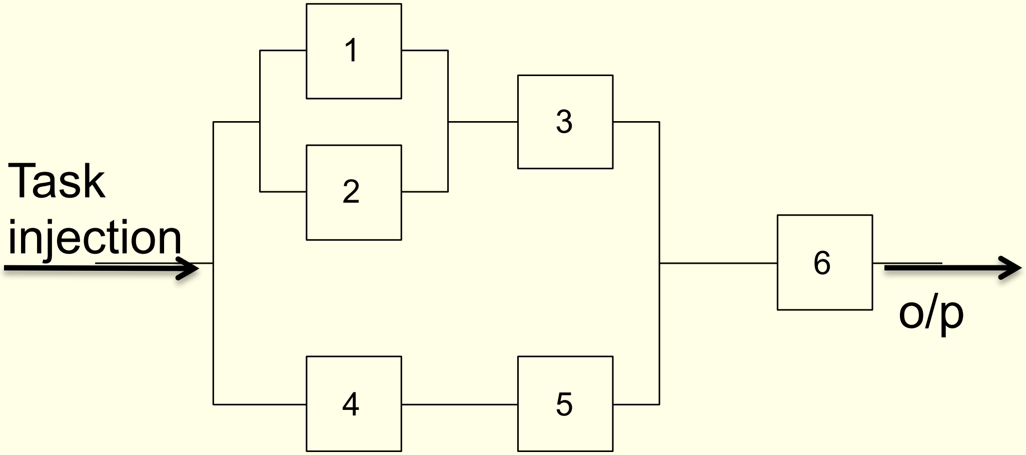
- Note that we have assumed that the reliability of a component in the system is **independent** of the failure status of other components.

Question 3.3 - Consider a system of 4 components with a parallel hardware configuration. If the reliability of each component is 0.8, what is the reliability of this overall parallel computing system ?

RTS – Design & Analysis of Reliable System Configurations

Question 3.4 - If a system with a parallel reliability configuration contains three components with constant failure rates of 0.4, 0.5, and 0.6 failures/months, respectively, determine the MTTF and failure rate of the overall system. Comment on the failure rate of the entire system.

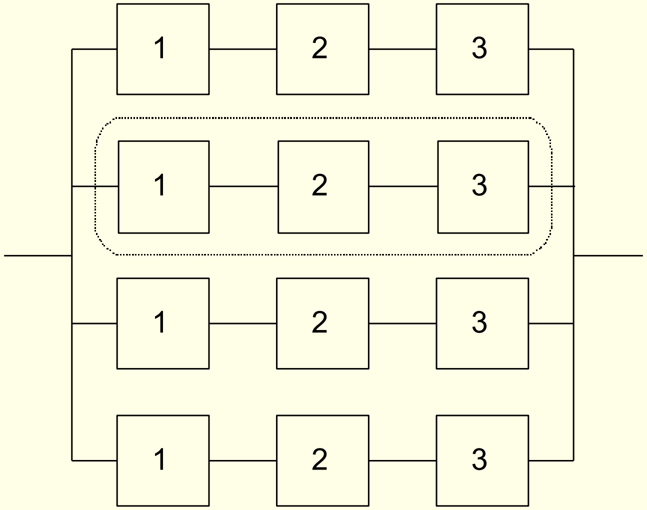
Question 3.5 – Consider a complex computing system with the following hardware configuration. Compute the overall system reliability.



RTS – Redundancy

Two levels at which redundancy can be applied to improve system reliability - high-level & low-level redundancy.

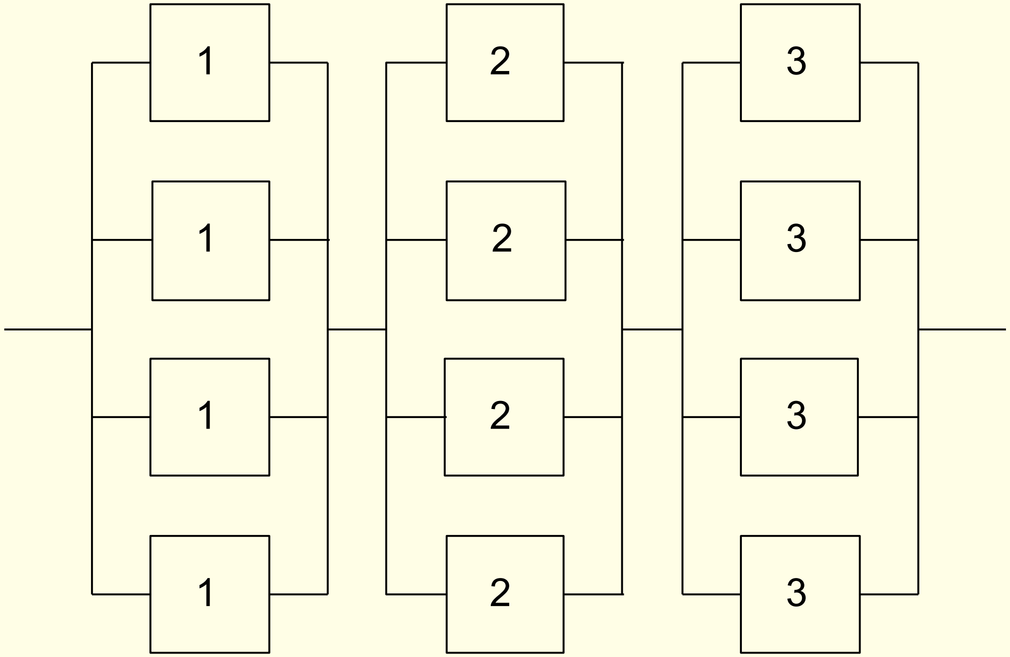
High Level Redundancy: In this case, identical components (with assumed series reliability configuration) are configured to form a parallel redundant system.



In simplest terms this implies the entire series configuration must be Replicated.

RTS – Redundancy

Low Level Redundancy: In this case, parallel redundancy is applied at the component level.



(c) Bharadwaj V 2016

RTS – Redundancy

- **Load Sharing Systems:** In this case, the load (stress) is shared - usually equally - among the operating components in the parallel redundancy configuration.
- If a component fails, the share of its load is then re-distributed among the remaining operating components (**Ex**: SoC/NoC)
- Hence the stress on the remaining operating components will increase thereby causing their reliability functions to change.

- **Standby Systems:** In this case, only one component of the parallel redundancy configuration carries the load - the remaining components are waiting in standby (off / idle) ready to take over (powered-up and connected in) from a failed component.

- The components on standby will only endure operating stress when engaged. They may endure a reduced environmental stress when idle.

K out of N Redundancy

- We can generalise and consider the case where k functional components are necessary for the system to function.

- **Example** - Consider a four core machine that can continue computing with only two cores. This means that at most, only two cores can fail. If R_j, is the reliability of the j-th core, and R is the reliability of the system:

R = R1 R2 R3 R4

+

F1 R2 R3 R4 + R1 F2 R3 R4 + R1 R2 F3 R4 + R1 R2 R3 F4

+

F1 F2 R3 R4 + F1 R2 F3 R4 + F1 R2 R3 F4 +

R1 F2 F3 R4 + R1 F2 R3 F4 +

R1 R2 F3 F4

- The first term is the probability that all cores work, the first group of terms are the probabilities for all combinations of single core failure, the second group of terms are the probabilities for all combinations of two-core failure.

K out of N Redundancy

- The example above illustrates the binomial nature of k out of n redundancy.
- If we have *n* components with identical and independent reliability *r*, then:

$$\mathrm{P(k\ operate\ AND\ n - k\ fail)} = \binom{n}{k} r^{k} (1 - r)^{n-k}$$

where $$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

- **Question 3.6** - Compute the probability for a four core system for which at least two cores must operate for the system to continue computing a certain task. Assuming that all cores have equal reliability *r* :

Software Fault Tolerant Techniques

- Software Fault-tolerance can also be influenced by HW faults and their tolerance

- S/W fault detection is a bigger challenge!

  -- Many S/W faults are of latent and may show up at random times - Software testing phase is crucial;

  -- Use of watchdogs to figure out if the program has crashed!

- Change of specification to provide any low-level service;

S/W FT Techniques

- Incorporating new versions of software

  - Isolate or remove original version

  - N-version Programming (NVP, 1977)

    Still used in Airbus 320 and related design;

- Use an on-line acceptance test to determine which version to accept:

  - Recovery Block (RB) schemes

What are the recovery choices available?

Software FT Design Techniques

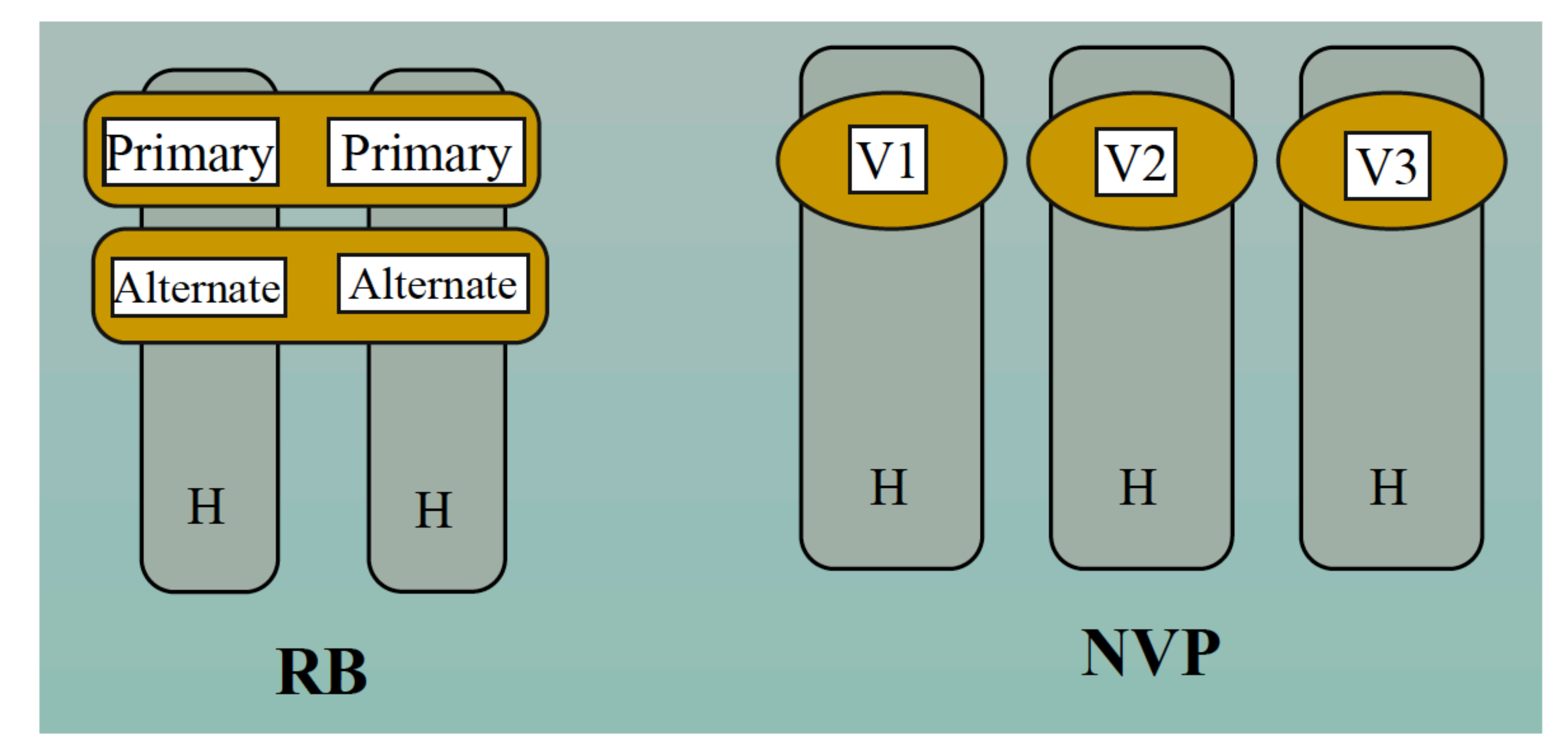
- Recovery Block Scheme (RB) - Dynamic redundancy

- N-version programming scheme (NVP)

Many versions operate on same specs and based on a voting scheme output will be accepted; *How to decide "N" is a challenge in S/w engineering!*

- Depending on H/W redundancy schemes - Hardware redundancy support is needed to implement the above S/W FT techniques

S/W FT Design Techniques



RB Scheme

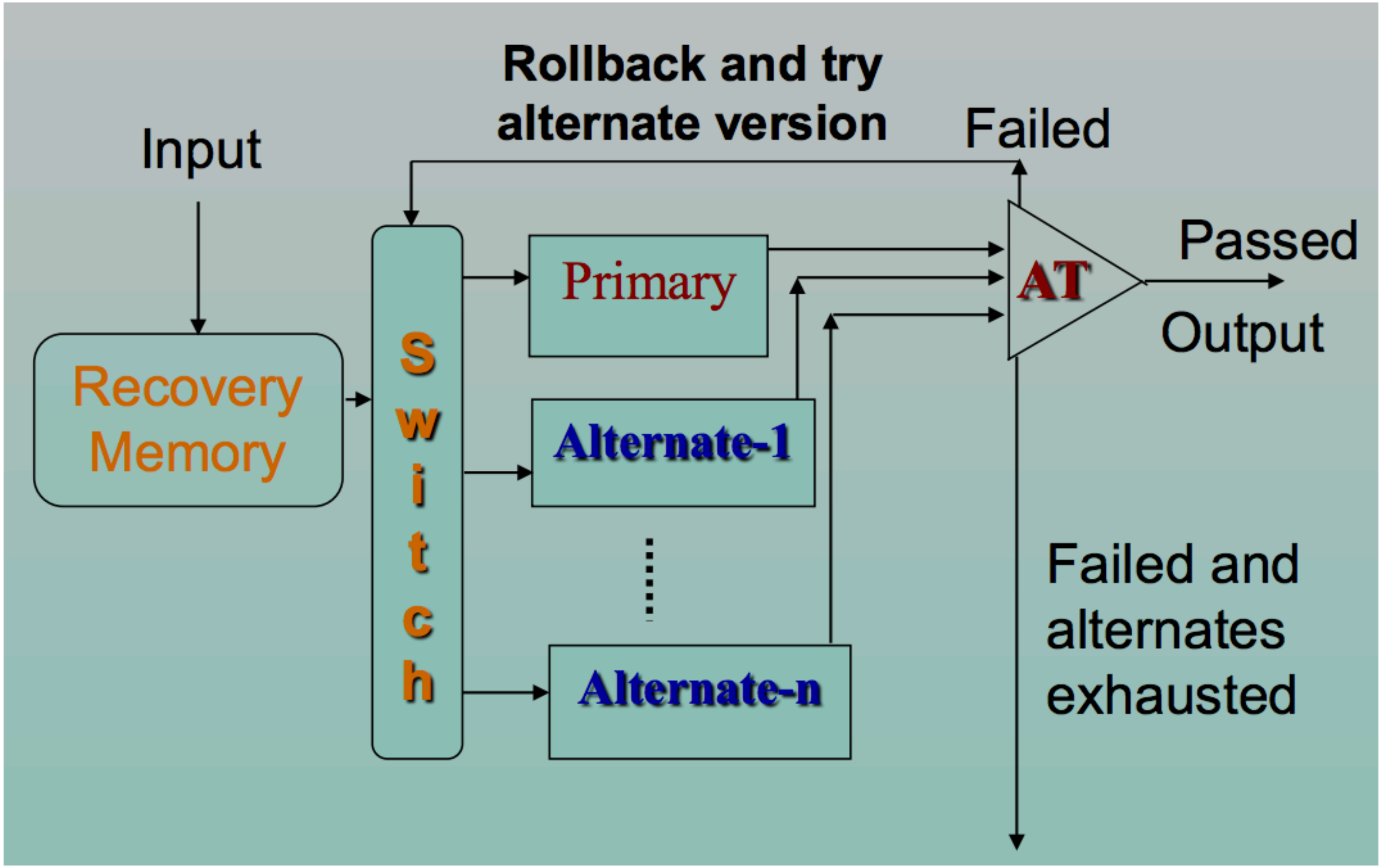
- RB Scheme - Comprises 3 elements

  - A Primary Module - To execute critical software functions

  - Acceptance Test for the output of the PM

  - Alternate modules to perform the same functions as of primary

RB Scheme



RB uses diverse versions.
Attempt to prevent residual software faults

NVP Scheme

- N-independent program variants execute in parallel on the identical input;
- Results are obtained by voting upon the output of the individual programs
- Prudent monitoring of the versions is needed and this is another level of challenge; Deciding on “N” is another challenge!

Fault Recovery Schemes

- Fault recovery technique's success depends on the detection of faults accurately and as early as possible;
- Check to be made to verify if reliability-aware design techniques are built-in
- Three classes of recovery procedures:

  - Full recovery (Needs all aspects of FTC)
  - Degraded recovery (referred to as graceful degradation)
  - Safe shutdown

Fault Recovery Schemes – 2 approaches

- Forward recovery

  - Produces correct results through continuation of normal processing; Each fwd step is validated;

  - Highly application dependent

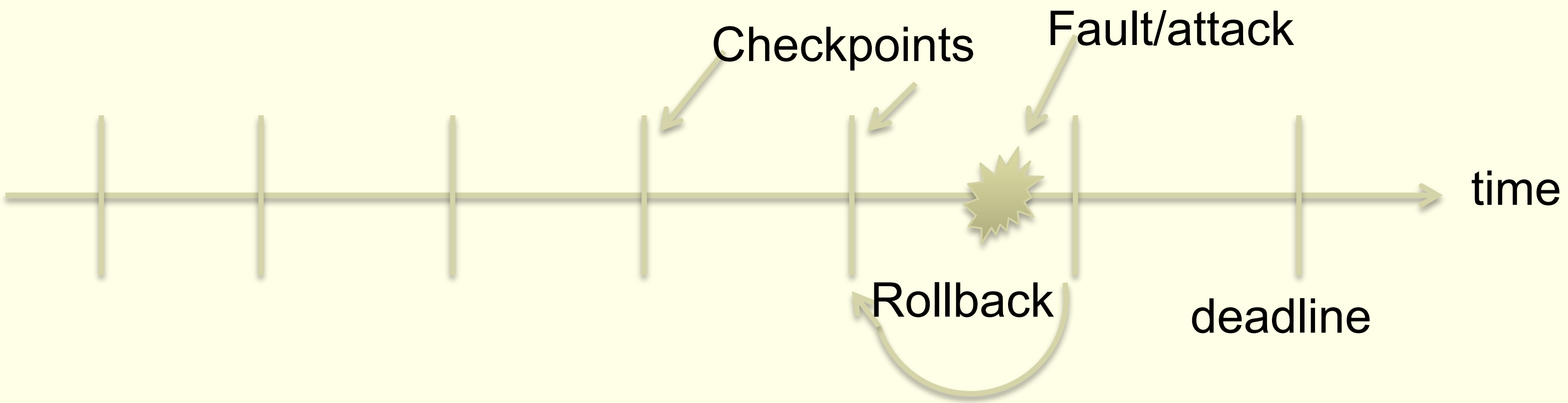
- Backward recovery

  - Some redundant process and state information is recorded continuously with the progress of computation;

  - Rollback the interrupted process to a point for which the correct information is available (Retry, checkpointing, etc)

Fault-recovery via check pointing technique (Ex: Task scheduling)

- Consider a DVFS enabled processor scheduling independent tasks;
- Check points are time epochs at which a check is made to see the correctness, consistency, etc, such that if a rollback is triggered the system enters the most recent verified checkpoint.
- Since RT tasks have deadlines a complete rollback is not a viable option as we may miss the deadline;



(c) Bharadwaj V 2016

Check pointing… (Cont'd)

- Within a checkpoint, a consistent state of the system is saved onto a stable storage.
- Moreover, before saving the task state at the instant of checkpointing, an acceptance test is run to discover the possible occurrences of transient faults.
- Therefore, if a fault occurs, its detection is postponed until the next checkpointing interval.
- If a fault occurrence is detected, rollback to the last checkpoint is done and the tasks remaining portion is executed by the maximum processor speed to catch the deadline! So, to some extent timing constraints need to be verified before guaranteeing deadline!

Availability

When we have repair capability, an alternative measure of system performance is ***Availability***.

To predict the system availability, both failure and repair probability distribution must be considered.

Availability = Uptime / (Uptime + Downtime)

Definition: *Availability is the probability that a system or component is performing its required function at a given point in time or over a stated period of time when operated and maintained in a prescribed manner.*

With that definition, let A(t) be the availability at time t, referred to as the point availability. Then,

Availability

Q: What is the average availability over an interval [0,T]?

Q: What is the average availability over an interval of time t_1 to t_2?

Inherent Availability – is defined as:

$$A_{inh} = (MTBF) / (MTBF + MTTR), \text{ where}$$

MTBF – Mean time between failures and MTTR is the mean time to repair;

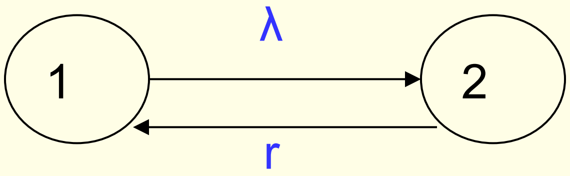
A_{inh} is also the steady-state availability found by taking limits as $t \rightarrow$ infinity;

We will see a simple example that captures the average and inherent availabilities using an exponential availability model.

Exponential Availability Model

Assumptions:

- We consider a single component;
- Constant failure rate λ;
- Constant repair rate r;
- System will be in one of the two possible states - operating or repair;



The transition rate diagram is shown above; Treating this as a Markov Process, the corresponding flow equations are:

$$dP_1(t)/dt = -\lambda P_1(t) + rP_2(t)$$

$$P_1(t) + P_2(t) = 1$$

Exponential Availability Model

Solving the above two equations we obtain:

$$P_1(t) = [r/(r+ \lambda)] + [\lambda/(r+ \lambda)].e^{-(\lambda+r)t}$$

Using this solution, we can compute the following:

(1) Since State 1 is the available state $A(t) = P_1(t)$ is the point availability of the component and provides the probability that the component is operating at time t.

(2) The interval availability is given by integrating the above point Availability over an interval say, t_1 to t_2;

(3) Steady-state availability =

$$A_{inh} = r/(r+ \lambda) = MTBF / (MTBF+MTTR)$$

System Availability

Using similar assumptions made to derive reliability,

- For n independent components in series, each having a component Availability $A_i(t)$, the system availability is given by:

$$A_{ser\_sys}(t) = A_1(t).A_2(t)....A_n(t)$$

- For n independent components in parallel, each having a component Availability $A_i(t)$, the system availability is given by:

$$A_{par\_sys}(t) = 1 – [(1-A_1(t)).(1-A_2(t))....(1-A_n(t))]$$

Maintainability

Maintainability is the measure of the ability of a system or item to be retained or restored to a specified condition when maintenance is performed by qualified personnel using specified procedure and resources.

Maintainability can be measured using Mean Time To Repair (MTTR), MTTR is average repair time and is given by

$$\text{MTTR} = \frac{(\text{Total maintenance downtime})}{(\text{Total number of maintenance actions})}$$

MTBMA is Mean Time Between Maintenance Actions including preventive and corrective maintenance tasks

THANK YOU!